ASFL: Adaptive Semi-asynchronous Federated Learning for Balancing Model Accuracy and Total Latency in Mobile Edge Networks

Jieling Yu
Wuhan University
Wuhan, China
yjling@whu.edu.cn

Ruiting Zhou*
Wuhan University
Southeast University
Nanjing, China
ruitingzhou@seu.edu.cn

Chen Chen
Shanghai Jiao Tong University
Shanghai, China
chen-chen@sjtu.edu.cn

Bo Li
Hong Kong University of Science and Technology
HK, China
bli@cse.ust.hk

Fang Dong
Southeast University
Nanjin, China
Fdong@seu.edu.cn

ABSTRACT

Federated learning (FL) is a new paradigm for privacy-preserving learning. This is particularly appealing in the mobile edge network (MEN), in which devices collectively train a global model with their own set of data. It is, however, routinely difficult for FL algorithms to satisfy different training task preferences in terms of the total latency and model accuracy due to a number of factors including the straggler effect, data heterogeneity, communication bottleneck and device mobility. To this end, we propose an **A**daptive **S**emi-asynchronous **F**ederated **L**earning (ASFL) framework, which adaptively balances the total latency and model accuracy according to the task preferences in MEN. Specifically, ASFL conducts a two-stage operation: i) *Device selection stage.* Each global round selects a set of devices that can maximize the model accuracy to eliminate data heterogeneity and communication bottlenecks; ii) *Training stage.* We first define a latency-accuracy objective value to model the balance between the latency and accuracy. Then in each global round, we use a deep reinforcement learning (DRL) algorithm based on soft actor-critic with discrete actions to intelligently derive the number of picked devices (*i.e.*, participants in the current global aggregation) and the lag tolerance at each global round to maximize the latency-accuracy objective value. Extensive experiments show that ASFL can improve the latency-accuracy objective value by up to 94% compared with three state-of-the-art FL frameworks.

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Learning settings**;

**Corresponding author

ICPP 2023, August 7–10, 2023, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-0843-5/23/08…$15.00
https://doi.org/10.1145/3605573.3605582

KEYWORDS

Federated learning, Semi-asynchronous, Deep reinforcement learning, Lag tolerance

ACM Reference Format:
Jieling Yu, Ruiting Zhou, Chen Chen, Bo Li, and Fang Dong. 2023. ASFL: Adaptive Semi-asynchronous Federated Learning for Balancing Model Accuracy and Total Latency in Mobile Edge Networks. In *52nd International Conference on Parallel Processing (ICPP 2023), August 7–10, 2023, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3605573.3605582

1 INTRODUCTION

Federated learning (FL) trains a global model by a large number of devices without exposing raw data, which has been widely applied in the mobile edge network (MEN) [10]. In MEN, different training tasks have different preferences in terms of the total latency and accuracy. For example, personalized recommendation services require fast training to timely adapt to the rapid interest change [5], while speech recognition services require more accurate models [21]. However, under MEN environment, it is challenging for existing FL frameworks to satisfy different training task preferences.

Challenges. There are four main factors in MEN that affect the total latency and accuracy of FL: i) **Straggler effect.** The latency per round is always [6] bounded by the *straggler*, i.e., the slowest device which is mainly caused by edge device heterogeneity in computing and communication capabilities. ii) **Data heterogeneity.** The local dataset on different edge devices are imbalanced and biased, which leads to accuracy degradation [4]. iii) **Mobility of devices.** The mobile devices may fall behind or disconnect, which affects the communication latency and slow down the training process. iv) **Communication bottleneck.** A large number of mobile devices concurrently communicate with the cloud server, causing communication congestion and increased communication latency.

Related Work. Existing FL methods mainly fall into two categories: synchronous FL and asynchronous FL. The synchronous FL methods [15], [4], [16], [1] mainly overcome the data heterogeneity and communication bottleneck through device selection, but the total latency is always slowed down by stragglers. Asynchronous FL eliminates stragglers by asynchronous aggregation [27], [12], [6].

But the local models trained based on unfashionable global model (*i.e.*, stale updates) would poison the global aggregation, resulting in accuracy degradation.

The emerging semi-asynchronous FL combines the benefits of both synchronous and asynchronous FL, which allows a part of devices to upload outdated models for global aggregation. Some semi-asynchronous FL methods [9], [28], [29] eliminate the stragglers by setting a deadline for a global round or adjusting the number of devices for global aggregation. The staleness problem is mainly solved by modifying the aggregated weight related to the staleness. In addition, some works [26], [14], [11] set the lag tolerance τ[1] to prevent the accuracy degradation caused by excessive staleness. However, existing FL frameworks are static (*e.g.*, set a fixed latency per round or a fixed lag tolerance) for maximizing accuracy or minimizing latency, which cannot satisfy the heterogenous training task preferences in MEN. In the Sec. 2, we will discuss in details.

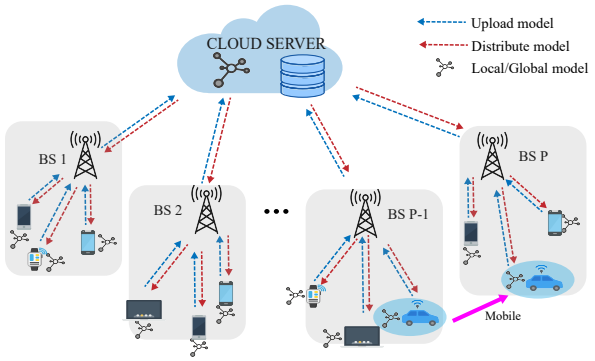

Figure 1: Illustration of ASFL process in MEN.

Our Solution. Considering different preferences of training tasks in MEN, we propose **A**daptive **S**emi-asynchronous **F**ederated **L**earning (ASFL). Given the straggler effect, data heterogeneity, mobility, and communication bottlenecks, ASFL adaptively adjusts the number of picked devices (*i.e.,* participants in the current global aggregation) and the lag tolerance to achieve a trade-off between total latency and accuracy. As shown in Fig. 1, devices are connected to the cloud server by the nearest base station (BS). ASFL takes a two-stage operation at k-th global round: i) Device selection stage. To mitigate the communication overhead and the data heterogeneity, the cloud server selects S devices based on the data quality without exposing privacy. The goal of selection is to maximize accuracy and establish an upper bound on the accuracy for the next stage. After selecting the devices, the cloud server distributes the global model to the selected devices through the BS. ii) Training stage. The cloud server waits for M_k devices to perform global aggregation. Then devices whose staleness exceeds the lag tolerance τ_k are forced to synchronize, *i.e.*, the local model is overwritten and the model version is updated. The number of picked devices M_k and the lag tolerance τ_k are adaptively determined by a deep reinforcement learning (DRL) method. By leveraging semi-asynchronous FL and DRL, ASFL can mitigate the effect of stragglers and adapt to the mobility of devices.

[1]The lag tolerance τ is the maximum tolerable staleness, and the staleness of device i is defined as $sl_i = v_G - v_i$, where v_G is the global model version and v_i is the local model version of device i. A model's version indicates the number of global rounds at which that model was updated.

Contributions. We summarize our main contributions as:

- To satisfy the heterogeneous training task preferences in MEN, we model the training process as a two-stage operation: i) We first model a device selection problem to overcome the communication bottleneck and handle data heterogeneity without compromising the data privacy. This stage aims to maximize the accuracy and can provide an upper bound of accuracy for the next stage. ii) We then design an accuracy-latency objective value to capture the preferences of accuracy and latency for heterogeneous edge tasks. We formulate an accuracy-latency objective value maximization problem. The solution of this problem can address the tradeoff between accuracy and latency by adjusting the number of picked devices and the lag tolerance.
- We propose an adaptive semi-asynchronous FL framework, ASFL. To perform device selection, ASFL defines a device utility which indicates device's contribution to the model accuracy. Then ASFL utilizes it as a metric to select the first S devices with the largest utilities. Next, to maximize the accuracy-latency objective value, we employ a DRL method to adaptively determine the number of picked devices and the lag tolerance at each FL global round. Specially, to handle two discrete decision variables, we propose a reinforcement learning agent based on an actor-critic architecture with discrete actions, and carefully design the reward in order to aggressively reduce the total latency.
- We conduct experiments on three tasks with different perferences: i) ASFL can achieve up to 80% , 59% and 94% higher objective value than SAFA [26], Oort [8], and FedCS [18]. ii) ASFL can greatly reduce the total latency, being up to 9×, 6×, and 19× faster than that of SAFA, Oort, and FedCS. iii) The accuracy of ASFL is higher than the other three benchmarks in tasks I and II. In Task III[2], which prefers to reduce latency, the accuracy is only 0.2% lower than one benchmark.

2 RELATED WORK AND MOTIVATION

Synchronous vs. Asynchronous FL. Since McMahan *et al.* [15] propose the classic Fedavg framework, there are many synchronous FL frameworks with different focuses. To alleviate the data imbalance, Duan *et al.* [4] present a self-balancing FL framework named Astraea, which creates the mediator to reschedule the client training for averaging the local imbalance. Wolfrath *et al.* [25] propose HACCS, a Heterogeneity-Aware Clustered Client Selection system that identifies and exploits the statistical heterogeneity. To incentivize the participation of heterogeneous clients, Pang *et al.* [19] design an incentive mechanism in FL through a procurement auction for social cost minimization. In order to maintain the long-term participation of high-quality data owners, Ng *et al.* [16] design an effective incentive scheme FedGame based on multiplayer games. To reduce the training latency, Luo *et al.* [13] devise an algorithm for client sampling that is adaptive to both system and statistical heterogeneity, in order to minimize the wall-clock convergence time. Pilla *et al.* [20] investigate the issue of reducing the duration of FL rounds by controlling the amount of data used for training by each device. Furthermore, some works have been proposed to achieve a balance between the training latency and accuracy. Zhou *et al.* [31] propose a new FL framework, Hca, to strike a balance between

[2]The details of task I-III is presented in Sec. 5

the job completion time and model accuracy. Cho *et al.* [1] present a communication- and computation-efficient biased client selection framework called Power-of-Choice that flexibly straddles the trade-off between convergence speed and accuracy. However, the straggler effect can significantly slow down the training progress in synchronous FL.

Asynchronous FL eliminates stragglers by asynchronous aggregation. To improve the flexibility and scalability, Xie *et al.* [27] propose an asynchronous federated optimization algorithm. In order to reduce the impact of non-IID datasets, Zhu *et al.* [32] propose STAFL, which adjusts the aggregation parameters based on each user's network weight and staleness on asynchronous updates. To deal with non-IID data, Nguyen *et al.* [17] present FedDRL, a new FL approach that leverages deep reinforcement learning to dynamically determine the aggregation weights for each client during the aggregation process. Taking the time efficiency into consideration, Zhou *et al.* [30] introduce a time-efficient protocol for asynchronous FL, TEA-Fed, which controls parameters to limit the number of devices simultaneously involved in training the same model. To tackle the straggler problem, Zhu *et al.* [33] adopt an asynchronous FL framework that minimizes the training latency through client selection while considering both client availability and long-term fairness. However, the staleness effect will poison the model convergence in asynchronous FL, leading to a low model accuracy.

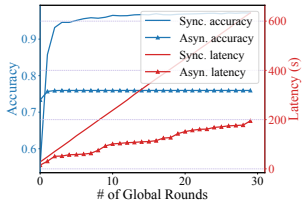

Figure 2: Accuracy/latency with synchronous FL and asynchronous FL.

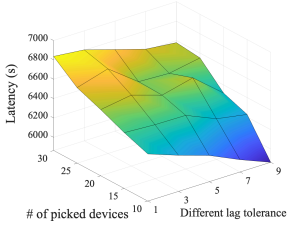

Figure 3: Latency under different M and τ.

To fully illustrate the strengths and weaknesses of both synchronous [15] and asynchronous [27] FL, a comparison of latency and accuracy is presented in Fig. 2. The comparison demonstrates that synchronous FL performs exceptionally well in terms of accuracy, while asynchronous FL exhibits lower latency. Therefore, we aim to develop a novel framework that strikes a balance between the two mechanisms, taking into account both accuracy and latency.

Semi-asynchronous FL. To date, many semi-asynchronous FL methods have been proposed. To solve the problem of stragglers in MEN, You *et al.* [28] present a Semi-Synchronous Personalized Federated Averaging (PerFedS2) algorithm that mitigates the straggler problem and ensures good convergence performance. Liang *et al.* [9] propose a semi-synchronous FL (semi-SynFed) protocol that dynamically adjusts the server waiting time of each global round under internet of vehicles to improve the time efficiency. Ma *et al.* [14] focus on three key challenges: edge heterogeneity, non-IID data and communication resource constraints, and propose a semi-asynchronous FL mechanism (FedSA) to determine the optimal number of participating workers to minimize the training completion time. There are also some semi-asynchronous FL works that directly address the balance between accuracy and latency. Yu *et al.* [28] design the Latency awarE Semi-synchronous client Selection and mOdel aggregation for federated learNing (LESSON) method, which sets a deadline and adjusts the trade-off between model accuracy and convergence speed by changing the deadline. Chen *et al.* [2] propose a novel Semi-asynchronous Hierarchical FL (SHFL) framework that derives the tradeoff between training accuracy and transmission latency. However, the above semi-asynchronous methods only consider the waiting latency for each global round (*i.e.*, the deadline or the number of participated devices), and does not consider the issue of lag tolerance. Only the SAFA designed by Wu *et al.* [26] considers both, but these two parameters are fixed values, which cannot adapt to real-time changes in MEN. We propose ASFL, which adaptively adjusts the number of picked devices and the lag tolerance to balance between latency and accuracy.

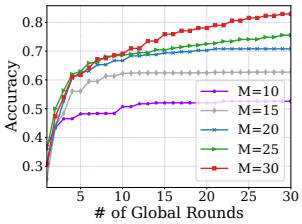

Figure 4: Accuracy with different M.

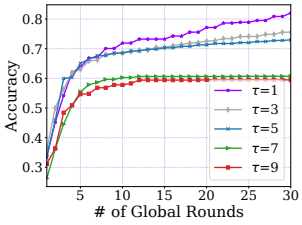

Figure 5: Accuracy with different τ.

We identify two significant factors in semi-asynchronous FL: the number of picked devices (M) and the lag tolerance (τ). We further assess their effects on accuracy and latency. We vary the values of M and τ in a semi-asynchronous FL structure [26], and plot the results in Fig. 3-Fig. 5. We have the following observations: i) *The effect of M*. The fluctuation of latency with difference numbers of picked devices (M) reveals that choosing more devices increases the impact of the straggler effect, leading to a slower convergence speed. Conversely, the change of accuracy demonstrates that a larger value of M leads to a higher accuracy. ii) *The effect of τ*. Increased lag tolerance results in greater device training utilization, which in turn reduces latency. However, a higher lag tolerance permits more stale local models to partake in global aggregation, which disrupts convergence and leads to decreased accuracy. Given these findings, our focus is to adaptively determine the optimal values of M and τ at every global round to address the balance between accuracy and latency in heterogeneous edge tasks.

3 FRAMEWORK OF ASFL AND SYSTEM MODEL

3.1 Adaptive Semi-asynchronous FL

Let $\mathcal{X}$ denote the integer set $\{1, 2, ..., X\}$. As illustrated in Fig. 1, in the mobile edge network, there are a set of mobile devices $\mathcal{I} = \{1, 2, ..., I\}$, P BSs and a cloud server. The mobile devices will move randomly whose communication with the cloud server needs to be forwarded through the nearest BS. We propose an adaptive semi-asynchronous FL framework, ASFL, to balance the total latency and accuracy adaptively according to the preference of the training task. The training progress of ASFL lasts for K global rounds. At k-th global round, ASFL takes a two-stage operation: i) **Device selection stage**. To overcome the communication bottleneck and handle data heterogeneity, the cloud server first selects a set of devices $\mathcal{S}_k$ ($|\mathcal{S}_k| = S < I$), and distributes the latest global model

to them to start their local training. This stage aims to maximize the accuracy by selecting the devices that can contribute the most to the next global round of training, and establish an upper bound of accuracy for the next stage. **ii) Training stage**. Each selected device $i \in \mathcal{S}_k$ performs E epochs of local training. At each epoch, device i trains a model on a local dataset $\mathcal{D}_i$, with its size D_i. To balance latency and accuracy adaptively, the server waits for the first M_k devices (which form the set $\mathcal{P}_k$, and $|\mathcal{P}_k| = M_k$) to perform global aggregation. More specifically, to update the global model at global round k, the cloud server performs global aggregation by:

$$\omega_G^{k+1} = \sum_{i \in \mathcal{P}_k} \frac{D_i}{D} \omega_i^*, \tag{1}$$

where ω_i^* is the transformed local model of device i which is defined as $\omega_i^* = \varsigma^{sl_i}\omega_i + (1 - \varsigma^{sl_i})\omega_G$. ω_i is the original local model uploaded by device i, ς[3] is the decay coefficient due to the staleness, and $0 < \varsigma < 1$. Next, according to the **lag tolerance** τ_k[4], devices exceeding the lag tolerance are enforced synchronization and the model versions are updated. The convergence analysis of ASFL is similar to [14], so we won't repeat it here.

Based on the training process of ASFL, there are three types of devices: i) **Picked devices.** Devices that participate in the current global aggregation (*i.e.*, $\mathcal{P}_k$). Note that $\mathcal{P}_k \neq \mathcal{S}_k$. ii) **Asynchronous devices.** Devices that do not perform their local training on the latest global models, but the staleness can be tolerated. iii) **Deprecated devices.** Devices whose version of initial models for local training are too stale (*i.e.*, exceeding τ_k).

3.2 Devices Selection Stage

Decision Variables. At this stage, high-quality (*i.e.*, datasets can efficiently improve accuracy) devices are selected to maximize the accuracy. The cloud server makes the decision $x_i^k \in \{0, 1\}$, which denotes whether or not to select device i at global round k.

Device Utility. To maximize the model accuracy, we hope to derive a device utility that can improve model accuracy by efficiently capturing the volume and distribution of the device dataset while respecting privacy. The design leverages the important conclusion in the literature [7] [8]: devices with larger accumulated loss can make more contributions to improving accuracy in future rounds. The utility of device i is calculated as $U_i = |\mathcal{D}_i|\sqrt{\frac{1}{|\mathcal{D}_i|}\sum_{d_i \in \mathcal{D}_i} Loss(d_i)}$. The utility is proportional to the importance of improving the model accuracy, where the training loss $Loss(d_i)$ of sample d_i is automatically generated during training with negligible collection overhead.

Selection Criteria. At each global round, the cloud server greedily selects the first S devices with the largest utilities to achieve the goal of maximizing accuracy. Important notations are listed in Table 1 for easy reference.

3.3 Training Stage

Decision Variables. The cloud server needs to make the following decisions to trade off the total latency and the accuracy: i) $M_k \in \mathbb{N}^+$: the number of picked devices at global round k; ii) $\tau_k \in \mathbb{N}^+$: the lag tolerance at global round k.

[3]The value of the parameters ς is described in Sec. 5.

[4]The definition of lag tolerance τ_k is introduced in Sec. 1.

Table 1: Notations and Descriptions

Symbol	Description
$\mathcal{I}$	the set of mobile devices in mobile edge network
K	the number of global rounds
$\mathcal{S}_k$	the set of selected devices at global round k
$\mathcal{P}_k$	the set of picked devices at global round k
$\mathcal{D}_i$	the local dataset of device i
m_{dist}^k	the number of distributed models at global round k
m_{syn}^k	the number of deprecated devices at global round k
U_i	the statistical utility of device i
r_i^k	the arrival latency for device i at global round k
r_{max}^k	the maximum arrival latency of picked devices at global round k
g_i	the training capability of device i
T_i^{cmp}	the computation latency of device i for local training
T_i^U	the upload latency of device i for upload local model to server
T_{dist}^k	the distribute latency at global round k
T^k	the latency at global round k
Variables	**Description**
x_i^k	whether device i is selected at global round k
τ_k	the lag tolerance at global round k
M_k	the number of picked devices at global round k

Next, we model the latency of k-th global round (denoted as T^k) , consisting of the distributed latency T_{dist}^k and the maximum arrival latency r_{max}^k at global round k. More details of these two components are described below.

Distributed Latency. T_{dist}^k denotes the server-side latency for distributing the global model to the devices. Since the bandwidth of the server can support sending the models to the devices in parallel [23], the distribution latency is only affected by the communication bandwidth of the cloud server (denoted as b_s) and the number of distributed global model copies (denoted as m_{dist}^k):

$$T_{dist}^k = \frac{m_{dist}^k \cdot G}{b_s}. \tag{2}$$

At global round k , m_{dist}^k is calculated as $m_{dist}^k = S + m_{syn}^k$, where S , m_{syn}^k indicates the number of selected devices and the number of deprecated devices at global round k respectively. G is the size of global model. Similar to [24], we ignore the latency for BS to send the global model to the devices since the number of model copies forwarded by a single BS is much less than that of the cloud server.

Arrival Latency. The maximum arrival latency r_{max}^k at global round k is defined as $r_{max}^k = \max_{i \in \mathcal{P}_k}\{r_i^k\}$. r_i^k is the arrival latency of device i at global round k, which is defined as:

$$r_i^k = \begin{cases} \max\{0, r_i^{k-1} - r_{max}^{k-1} - T_{dist}^k\}, & i \text{ is asyn.} \\ T_i^{cmp} + T_i^U, & \text{otherwise} \end{cases} \tag{3}$$

The arrival latency of device i at global round k is divided into two cases: i) The device is a deprecated device or a newly selected device at the current round, which starts the training from the current round. Then the arrival latency of device i includes the computation latency T_i^{cmp} and the upload latency T_i^U. The computation latency is calculated as $T_i^{cmp} = \frac{\Delta_i E}{g_i}$, where Δ_i is the mini-batch size of device i, E is the number of local training epochs, g_i is the training capability of device i, indicating how many data samples can be processed per second. The upload latency is defined as $T_i^U = \frac{\Gamma}{b_i^k}$, where Γ is the model size and b_i^k is the bandwidth of the device i at

k-th global round, and b_i^k is inversely proportional to the distance from the device to the BS. Similarly, we ignore the latency for BS to send the local models to the cloud server [24]. ii) The device is an asynchronous device: the device continues the previous process. Its arrival latency at global round k is the arrival latency of last round r_i^{k-1} minus the maximum arrival latency of last global round r_{max}^{k-1} and the distributed latency of current round T_{dist}^k (because the asynchronous device does not need to wait for distributing model).

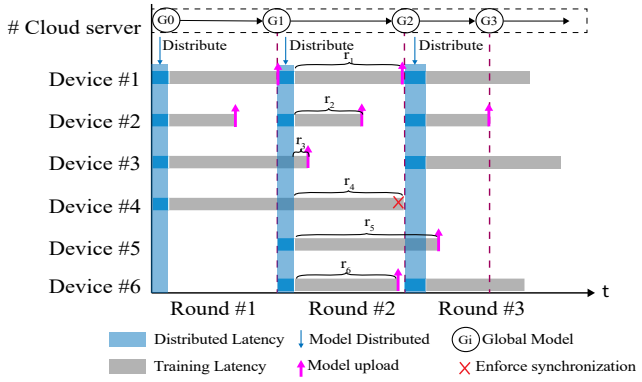

Figure 6: The process of global aggregation.

In Fig. 6 we show the arrival latency and global aggregation process of ASFL, which includes six mobile devices (devices #1 to #6) and performs three global rounds of training. For example, at global round 2, we assume that the number of selected devices $S = 4$, the number of picked devices $M_2 = 4$, the lag tolerance $\tau_2 = 1$. At the beginning of global round 2, the cloud server distributes the latest global model ω_G^1 to the selected devices #1, #2, #5, #6 and picks devices #1, #2, #3, #6 to perform the global aggregation. In addition, device #4 is deprecated as its staleness $sl_4 = 2 - 0 = 2$ is bigger than τ_2, which causes the distributed latency of the global round 3 to be longer (as $m_{syn}^3 = 5$). Note that the arrival latency r_i^k is a round-varying value, and r_i^2 is marked in Fig. 6.

Model for Total Latency and Accuracy Tradeoff. To model the heterogeneous preferences of different training tasks in terms of total latency and accuracy, we adopt a customized weighted product method to define a latency-accuracy objective function [22]:

$$\max_{M_k,\tau_k} \quad acc^K \times \left[\frac{T_{max}}{\sum_{k=1}^{K} T^k}\right]^{\vartheta}, \tag{4}$$

where ϑ is the weight factor defined as:

$$\vartheta = \begin{cases} \alpha, & T^K \le T_{max} \\ \beta, & \text{otherwise} \end{cases}, \tag{5}$$

and acc^K is the final accuracy (*e.g.*, the ratio of the number of positive samples predicted by the global model to the the number of total samples), and α and β are task-specific constants, representing the task's preference on latency before and after exceeding the latency limit T_{max} for total K rounds of training.

Fig. 7 illustrates the objective function with two typical values of (α, β). In the top figure with $(\alpha = 0, \beta = 1)$, the task prioritizes the accuracy and puts zero weight on the total latency when the total latency is within the latency limit T_{max}. But it sharply penalizes the objective value when the latency limit is passed. In the bottom figure with $(\alpha = 0.07, \beta = 0.07)$, the task is more tolerant to latency

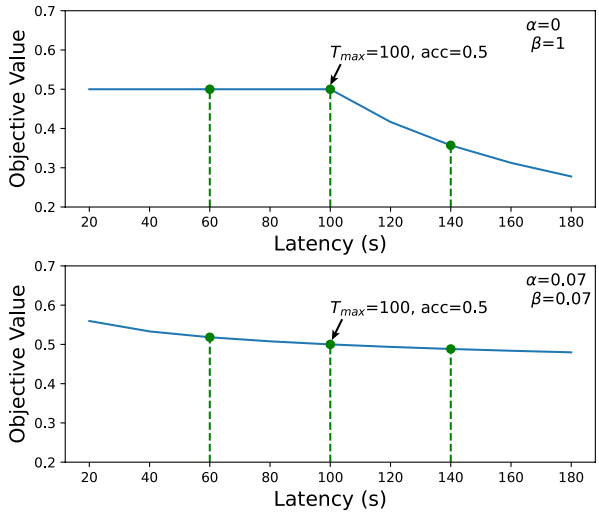

Figure 7: The trend of objective function.

(treats the latency limit T_{max} as a soft constraint), and pays more attention to accuracy while slowly penalizing the objective value according to the latency.

4 THE DESIGN OF ASFL

4.1 Design Overview

ASFL is designed with two stages to maximize the latency-accuracy objective value:

i) Device selection stage. This stage aims to maximize the accuracy and establish an upper bound on accuracy for the subsequent stage. We define the utility of the device to measure its importance of accuracy improvement. The cloud server selects the first S devices with the largest utilities. Refer to Sec. 3.2 for details.

ii) Training stage. After selecting the devices, the cloud server adopts the DRL method to determine M_k and τ_k at k-th global round to solve the problem (4). We first converted problem (4) into problem (6) in order to convert it into a Markov decision process (MDP). Specifically, the cloud server is the agent, and the semi-asynchronous FL system is the environment. At k-th global round, the agent observes the current state s^k from the environment, and inputs it into the policy π to get the actions a^k. Here, the policy π is parameterized with θ (*e.g.*, the weights of the neural network for the agent). After the agent takes the action, the environment feeds back a reward r^k and changes. When the next global round starts, the environment will feed back the new state s^{k+1} to the agent. The transitions $\{s^k, a^k, r^k, s^{k+1}\}$ are stored in an experienced pool for updating its actor and critic networks. The learning lasts until the convergence of the actor and critics. After that, the convergent model of DRL can be used to determine the number of picked devices M_k and the lag tolerance τ_k for k-th global round.

4.2 Design of DRL Algorithm in ASFL

To maximize the latency-accuracy objective value, we adopt the DRL approach based on soft actor-critic with discrete actions (SACD) [3] to solve the problem (4). After the DRL training is complete, DRL only incurs only milliseconds of inference to make action decisions, which is negligible and doesn't affect the FL training process.

Problem Transformation. The standard reinforcement learning objective is to maximize the expectation of the cumulative discounted rewards $\mathcal{R} = \sum_{k=1}^{K} \gamma^{k-1} r^k$. However, it is hard to convert from the latency-accuracy objective value to the reward as it is not similarly cumulative in form. Then we transform the original problem (4) into the cumulative form:

$$\max_{M_k, \tau_k} \quad \sum_{k=1}^{K} acc^k \times \left[\frac{T^k_{lim}}{T^k} \right]^{\vartheta}, \tag{6}$$

where acc^k is the accuracy of k-th global round and ϑ is the weight factor defined as:

$$\vartheta = \begin{cases} \alpha, & T^k \le T^k_{lim} \\ \beta, & \text{otherwise} \end{cases}, \tag{7}$$

and $T^k_{lim} = \frac{1}{(K-k)}(T_{max} - \sum_{k=1}^{K} T^k)$ is the average round latency limit which replaces the latency limit T_{max}.

Next, we convert problem (6) into a MDP, which is described by the tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$. Here $\mathcal{S}$ denotes the state of a time sequence and $\mathcal{A}$ denotes actions of a time sequence. The policy function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is used to compute the state transition probability $p(s^{k+1}|s^k, a^k)$, given current action a^k and state s^k. The $\mathcal{R}$ is a sequence of rewards that are discounted by the factor $\gamma \in [0, 1]$. And the reward function is defined as $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The details of the state, action, and reward are presented as follows:

1) **State** $\mathcal{S}$: At k-th global round, the state of the semi-asynchronous FL environment consists of: i) The global loss $F(\omega^k_G)$ which shows the current training accuracy state. ii) The latency ratio lo^k is the ratio of the total latency of the first $k-1$ global rounds to the latency limit T_{max}. It indicates the current latency state. iii) The objective value of the original problem O^k which represents the state of the objective value at the first $k-1$ global rounds. iv) The synchronous ratio so^k is the average number of deprecated devices at each global round (*i.e.*, $so^k = \frac{1}{(k-1)} \sum_{i=1}^{k-1} m^k_{syn}$). It measures the wasted process state at global round k. v) The version variance vv^k is the average version variance, reflecting the staleness state of the past $k-1$ rounds (*i.e.*, $vv^k = \frac{1}{(k-1)} \sum_{i=1}^{k-1} var(\mathcal{V}^i)$). Therefore, the state observed by the agent at global round t is represented by a vector $s^k = \{F(\omega^k_G), lo^k, O^k, so^k, vv^k\}$.

2) **Action** $\mathcal{A}$: Once the state s^k is observed, the cloud server (*i.e.*, the agent) determines the action a^k at global round k for FL training. The agent decides the number of picked devices M_k and the lag tolerance τ_k at each global round.

3) **Reward** $\mathcal{R}$: After the environment takes the action a^k, the agent will receive a reward $r^k \in \mathcal{R}$. Naturally, combining the classical reward function and the objective function of problem (6), the reward at global round k is defined as:

$$r^k = acc^k \times \left[\frac{T^k_{lim}}{T^k} \right]^{\vartheta}. \tag{8}$$

Intuitively, when the cumulative discounted rewards reaches the maximum, the problem (6) also reaches the maximum.

Design of SACD-based Algorithm. Considering the challenges that our high-dimensional continuous state and discrete actions are difficult to converge [3], we adopt SACD framework which is shown in Fig. 8. It is an off-policy algorithm based on the maximum entropy DRL framework, which is sample efficient and emphasizes exploration. SACD uses a special actor-critic architecture that

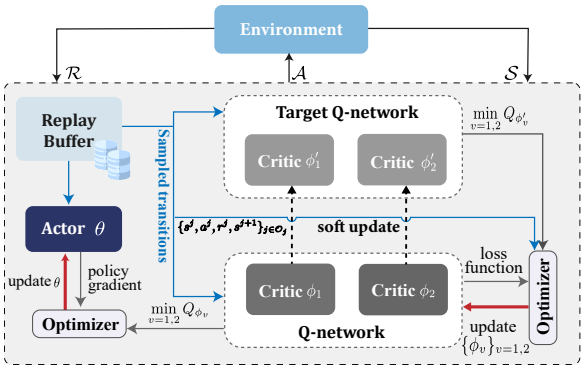

Figure 8: The SACD Framework.

employs an actor and two Q-networks to approximate both the policy π_θ and the Q-functions $\{Q_\phi, Q_{\phi'}\}$. Two of the Q-networks are named Q-network and target Q-network, and each of them has two critics. The main feature of SAC is entropy regularization, whose objective function adds an entropy term, $\mathcal{H}(\pi(\cdot|s^k)) = -\log \pi(\cdot|s^k)$, as a regularizer [34]. Then the modified formulation of the policy gradient objective is:

$$\pi^*_\theta = \arg\max_{\pi} \sum_{k=1}^{K} \mathbb{E}_{(s^k, a^k) \sim \kappa_\pi} [\gamma(r^k + \alpha \mathcal{H}(\pi(\cdot|s^k)))], \tag{9}$$

where α is an adjusting factor that indicates the importance of the entropy term over the reward. κ_π is the distribution of trajectories induced by policy π_θ. Then the discrete action is discretized by the output generated by the actor according to the current state and policy π_θ.

SACD Training Details. As the training of SACD leverages experience replay buffer O_b, the tuple of a state transition is stored into the experience pool as $\{s^k, a^k, r^k, s^{k+1}\}$. Accordingly, the agent randomly samples a mini-batch transitions $\{s^j, a^j, r^j, s^{j+1}\}_{j \in O^j}$ from the replay buffer and updates the parameters ϕ_v of Q-network:

$$J_Q(\phi_v) = \mathbb{E}_{(s^j, a^j) \sim O^j} [\frac{1}{2}(Q_{\phi_v}(s^j, a^j) - \hat{Q}^j)^2], v = 1, 2. \tag{10}$$

with the target Q-values:

$$\hat{Q}^j = r^j + \gamma(\min_{v=1,2} Q_{\phi'_v}(s^{j+1}) - \alpha \log \pi_\theta(s^{j+1})). \tag{11}$$

The parameters ϕ_v can be optimized with stochastic gradients (*i.e.*, $\nabla_\phi J_Q(\phi)$). In the implementation, SACD [3] uses the minimum of two delayed-update target critic network outputs to reduce overestimation. The target Q-network $\{\phi'_v\}_{v=1,2}$ performs a soft update periodically:

$$\phi'_v = \delta \phi_v + (1 - \delta)\phi'_v, v = 1, 2. \tag{12}$$

In addition, the parameters of the actor θ are optimized according to the Q-value of the Q-network $(Q_{\phi_1}(\cdot), Q_{\phi_2}(\cdot))$ by minimizing the objective function:

$$J_\pi(\theta) = \mathbb{E}_{s^j \sim O^j} [\pi^j(s^j)^T [\alpha \log(\pi_\theta(s^j)) - \min_{v=1,2} Q_{\phi_v}(s^j)]]. \tag{13}$$

Then optimizing the factor α called temperature parameter involves minimizing:

$$J(\alpha) = \pi^j(s^j)^T [-\alpha \log(\pi_\theta(s^{j+1})) - \hat{H}], \tag{14}$$

where $\hat{H}$ is the target entropy, and we set it as a constant.

Algorithm 1 Soft Actor-critic with Discrete Actions Based Algorithm (SAA)

1: Initialize the weights of the actor network θ, the Q-networks ϕ_1, ϕ_2 and the target Q-networks $\phi_1' \leftarrow \phi_1$, $\phi_2' \leftarrow \phi_2$, the replay buffer O_b, hyperparameters γ, α, δ, ℓ_Q, ℓ_π, ℓ_α, $\bar{H}$;
2: **for** $ep = 1$ to EP **do**
3: Initialize the semi-asynchronous FL environment, and observe the initial state s^1
4: **for** $k = 1$ to K **do**
5: Get the discrete actions a^k based on the current state;
6: Apply actions a^k to the semi-asynchronous FL environment, and obtain the reward r^k feed backs;
7: Observe the next state s^{k+1}, then store $\{s^k, a^k, r^k, s^{k+1}\}$ in the replay buffer O_b;
8: Randomly sample a mini-batch of O^j experiences $\{s^j, a^j, r^j, s^{j+1}\}_{j \in O^j}$ from replay buffer O_b;
9: Update the Q-network: $\phi_v \leftarrow \phi_v - \ell_Q \nabla \phi_v J_Q(\phi_v), v = 1, 2$;
10: Update the weights of actor network: $\theta \leftarrow \theta - \ell_\pi \nabla \pi J_\pi(\theta)$;
11: Update the temperature parameter: $\alpha \leftarrow \alpha - \ell_\alpha \nabla \alpha J(\alpha)$;
12: Soft update the target Q-network: $\phi_v' \leftarrow \delta \phi_v + (1 - \delta)\phi_v', v = 1, 2$.
13: **end for**
14: **end for**

Algorithm Details. The soft actor-critic with discrete actions algorithm, SAA, is presented in Alg. 1. At line 1, it initials some hyperparameters and the parameters of the actor, the Q-networks, and the target Q-networks, the replay buffer. Then it starts DRL for EP episodes. At every episode, line 3 initials the semi-asynchronous FL environment and observes the original state. At each global round, lines 5-7 first observe the current state $s^k = \{F(\omega_G^k), lo^k, O^k, so^k, vv^k\}$ to get the discrete actions $a^k = \{M_k, \tau_k\}$ based on current policy π^k. To get the discrete actions, we apply a softmax activation function in the final layer of the Q-network to ensure that it outputs a valid probability distribution, and sample discrete actions according the probability distribution. After the agent takes the actions and receives a reward r^k, the environment switches to the next state s^{k+1}. The transitions $\{s^k, a^k, r^k, s^{k+1}\}$ is stored into replay buffer O_b for the next DRL training. Lines 8-12 sample a mini-batch transitions O_j to update the weights of the Q-network ϕ, the actor θ, and the target Q-network ϕ', and adjust the temperature parameter α.

5 EXPERIMENTS

5.1 Experiment Setup

Setup for semi-asynchronous FL. In this section, we implement ASFL by PyTorch and verify its effectiveness with experiments under three training tasks and three data distributions. Task I trains a regression model on the public Boston Housin dataset[5], which prefers latency, but still maintains a certain accuracy requirement (*i.e.*, $\alpha = 0.1, \beta = 0.5$). Task II trains classical MNIST dataset using a CNN model, which is consists of two 5×5 convolution layers with 2×2 max pooling. The task prefers accuracy, so we set $\alpha = 0.07, \beta = 0.09$. Task III learns a SVM model for detecting network intrusion given the TCP dump data. And the trained data is a subset of the KDD Cup'99 dataset[6]. Task III is extremely latency sensitive with $\alpha = 0.5, \beta = 0.9$. The specific settings of the three tasks are shown in Table 2.

Table 2: Setup For Tasks

Parameter	Task I	Task II	Task III
Dataset	Boston	MNIST	KDDCup99
# of features	13	28×28	35
model	Reg	CNN	SVM
data size	506	80k	186k
# of devices	20	100	500
# of selected devices	6	10	25
# of local epochs	3	5	5
(α, β)	(0.1,0.5)	(0.07,0.09)	(0.5,0.9)
$T_{max}(s)$	2000	5000	8000
mini-batch size	5	64	64
learning rate	1e-4	1e-3	1e-2

As the training capability of devices $\{g_i\}_{i \in I}$, we assume they obey the exponential distribution with $\lambda = 1.0$, denoted as $Exp(\lambda)$, and the bandwidth of server is set to 1250, the bandwidth of devices is set in range $[0.125, 0.3]$ [26]. For clarity, we use Gaussian distribution to simulate the imbalanced and heterogeneous data distribution [26]: we assume the local data size follows the Gaussian distribution $N(\mu, a\mu)$, where μ is the ratio of the number of devices to the total amount of data, and a indicates the degree of imbalance: the larger a, the greater the degree of imbalance. In our experiments, we set $a = 0.2, 0.3, 0.4$, which is referred to as N2, N3, N4 respectively. The decay coefficient of staleness $\varsigma = 0.9$ is set according to [11].

Table 3: Hyperparameters of DRL

Parameter	Value
Discounting factor γ	0.99
soft update factor δ	0.01
Size of replay buffer O_b	5000
Temperature factor init.	1.
Learning rate ℓ_Q, ℓ_π, ℓ_α	1e-6
Optimizer	Adam
Activation function	ReLU

Setup for DRL. The actor consists of four layers, which are an input layer with output dimension 128, a middle layer with 128×128 and two output layers with input dimension 128. The critics also have three layers, but the middle layer is 128×64 and the output layer has an input dimension of 64. The total episodes of task I and task III is 1000, while the total episodes of task II is 600. Other important hyperparameters are listed in Table 3.

Benchmarks. To evaluate the performance of ASFL, the following three benchmarks are compared.

- SAFA [26]: SAFA is a semi-asynchronous FL protocol that picks M devices for aggregation in each global round, and stores devices with response time within the average round latency limit T_{lim}^k and staleness within the lag tolerance τ in cache for the next round of global aggregation. In this algorithm, T_{lim}^k is defined as $\max_{i \in I}\{T_i^{cmp} + T_i^U\} + T_{dist}^k$. We choose SAFA to compare with ASFL in terms of addressing the straggler effect in asynchronous FL.

[5]https://www.cs.toronto.edu/ delve/data/boston/bostonDetail.html

[6]https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

- FedCS [18]: FedCS is a refined FedAvg algorithm that has to drop devices with the latency per round exceeding average round latency limit T^k_{lim} at the stage of device selection, where $T^k_{lim} = T_{max}/K$. FedCS was selected to compare with ASFL in terms of handling data heterogeneity in FL.
- Oort [8]: Oort aims to balance the accuracy and latency, and greedily selects the device with the largest utility. Oort was chosen to compare with ASFL in terms of balancing the latency and accuracy in FL with similar priorities.

Note that M and τ are constants for the three benchmarks, and we set them as 5, 2 respectively.

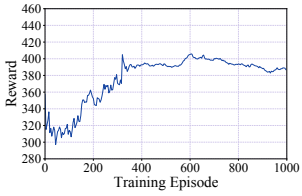

Figure 9: Convergence on Task I.

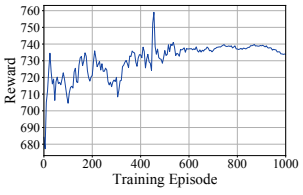

Figure 10: Convergence on Task II

Convergence of reward. Fig. 9-11 depict the convergence curves of the SAA's learning process for tasks I-III. We performed 1000 episodes of DRL on both tasks I-III. It can be seen that task I and III converge at about 400 episodes and task II converges at about 600 episodes. Each episode completes a whole FL training, which starts from initializing the FL environment and ends after training the model for 50 global rounds.

Figure 11: Convergence on Task III.

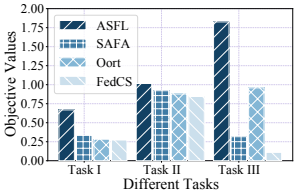

Figure 12: Objective values on different benchmarks.

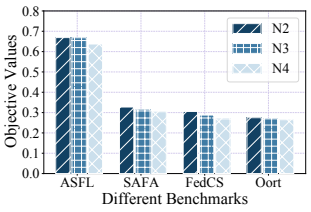

Figure 13: Objective values on different distribution.

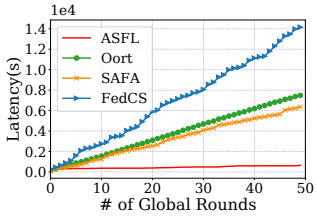

Figure 14: Latency of task I on different benchmarks.

Performance of latency-accuracy objective value. We first demonstrate the effectiveness of ASFL by comparing it with other benchmarks (shown in Fig. 12), and then study the influence of data imbalance degree on the objective value (shown in Fig. 13). Obviously, we can conclude that ASFL can always efficiently balance the latency and accuracy according to the preference of the task. In addition, the degree of imbalance is inversely proportional to the objective value. This is because the higher the degree of imbalance, the lower the upper bound of accuracy determined by the device selection stage, resulting in a lower overall objective value.

Task type	Distribution	ASFL	Oort	SAFA	FedCS
Task I	N2	0.668	-58.88%	-50.49%	-54.61%
	N3	0.669	-59.78%	-52.64%	-55.59%
	N4	0.637	-58.43%	-52.27%	-49.60%
Task II	N2	1.018	-13.28%	-9.57%	-23.39%
	N3	1.015	-20.40%	**-11.31%**	-23.48%
	N4	1.013	-11.91%	-11.54%	-23.08%
Task III	N2	1.853	-46.29%	-80.09%	-93.74%
	N3	1.825	-47.51%	-79.90%	-94.02%
	N4	1.761	-46.78%	-79.48%	**-94.09%**

Table 4: Latency-accuracy objective values on different distributions

Table 4 records the specific objective value results. On average, for task I, ASFL can achieve 51% , 59% and 53% higher than the objective value of SAFA, Oort, and FedCS. For task II, ASFL can improve the objective value by 10% , 15% and 23% compared with SAFA, Oort, and FedCS. For task III, ASFL can improve the objective value by 80% , 47% and 94% compared with SAFA, Oort, and FedCS. Moreover, the results show that more devices bring higher objective value. Since the number of asynchronous devices is proportional to the total number of devices, the efficiency of the training process can be improved (i.e., with a reduced deprecated rate), thereby achieving a larger reduction in latency.

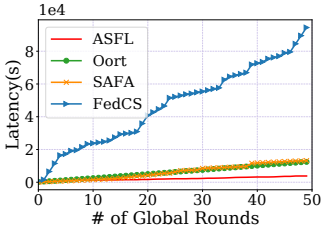

Figure 15: Latency of task II on different benchmarks.

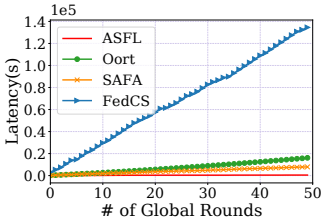

Figure 16: Latency of task III on different benchmarks.

Performance of latency. Fig. 14-16 show the comparison of latency under different task types. The latency of each benchmark is the average latency under different distributions. It can be clearly found that the reduction of latency by ASFL is significant, and it performs the best under different distributions. Compared to the classical Fedavg framework [15], ASFL can make more efficient use of the stale local models. Furthermore, ASFL adjusts the number of picked devices and lag tolerance to avoid unnecessary synchronous latency and unused stale local models. This approach can significantly reduce the training latency.

For task I, ASFL trains the task 9, 5, and 5 times faster than SAFA, FedCS and Oort. For task II, ASFL runs 3, 8, 3 times faster than SAFA, FedCS and Oort. For task III, ASFL runs 5, 19, 6 times faster than SAFA, FedCS and Oort. Since task II focuses more on accuracy, the latency reduction in task II is smaller than that of the other two tasks.

Performance of accuracy. Under different data distributions, the accuracy results of three benchmarks are shown in Table 5. Since task I pays more attention to latency with modest accuracy requirements and task II prefers accuracy, the final accuracy of ASFL is the best in tasks I and II. But task III extremely prefers to reduce latency, which will inevitably inhibit the accuracy, so the accuracy is 0.1% and 0.2% lower than that of Oort and FedCS on N3. In addition,

Task type	Distribution	ASFL	Oort	SAFA	FedCS
Task I	N2	**0.622**	0.618	0.621	0.615
	N3	**0.621**	0.617	0.621	0.614
	N4	**0.622**	0.616	0.621	0.616
Task II	N2	**0.985**	0.984	0.980	0.981
	N3	**0.987**	0.986	0.984	0.984
	N4	**0.986**	0.980	0.976	0.979
Task III	N2	**0.999**	**0.999**	0.996	**0.999**
	N3	0.997	0.998	0.996	**0.999**
	N4	**0.999**	**0.999**	0.996	**0.999**

Table 5: Accuracy results on different distributions

task I will be slower than other benchmarks in terms of accuracy growth rate, due to the emphasis on latency, but the total latency of ASFL is greatly reduced.

6 CONCLUSION

In this work, we propose ASFL, an adaptively semi-asynchronous FL for heterogeneous tasks in mobile edge networks, to adaptively balance the total latency and the accuracy. ASFL designs a latency-accuracy objective value to model different training task preferences in terms of the total latency and model accuracy. ASFL first selects high-quality devices to maximize the accuracy. Then, ASFL adopts DRL to adaptively determines the number of picked devices and the lag tolerance to balance the total latency and the accuracy. The extensive experiments based on real-world data verify that ASFL can significantly improve the objective value, compared with existing FL frameworks.

ACKNOWLEDGMENTS

The research was supported in part by the National Natural Science Fundation of China (NSFC) Grants (62072344, U20A20177, 62232004 and 62202300), and Shanghai Pujiang Program (22PJ1404600), in part by a RGC RIF grant under the contract R6021-20, and RGC GRF grants under the contracts 16209120, 16200221 and 16207922.

REFERENCES

[1] Yae Jee , Jianyu Wang, and Gauri Joshi. 2022. Towards understanding biased client selection in federated learning. In *Proc. of AISTATS*. PMLR, 10351–10375.
[2] Qimei Chen, Zehua You, and Hao Jiang. 2021. Semi-asynchronous hierarchical federated learning for cooperative intelligent transportation systems. *arXiv preprint arXiv:2110.09073* (2021).
[3] Petros Christodoulou. 2019. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207* (2019).
[4] Moming Duan, Duo Liu, Xianzhang Chen, Renping Liu, Yujuan Tan, and Liang Liang. 2020. Self-balancing federated learning with global imbalanced data in mobile systems. *IEEE Transactions on Parallel and Distributed Systems* 32, 1 (2020), 59–71.
[5] Guibing Guo, Enneng Yang, Li Shen, Xiaochun Yang, and Xiaodong He. 2019. Discrete Trust-aware Matrix Factorization for Fast Recommendation.. In *Proc. of IJCAI*. 1380–1386.
[6] Dzmitry Huba, John Nguyen, Kshitiz Malik, Ruiyu Zhu, Mike Rabbat, Ashkan Yousefpour, Carole-Jean Wu, Hongyuan Zhan, Pavel Ustinov, Harish Srinivas, Kaikai Wang, Anthony Shoumikhin, Jesik Min, and Mani Malek. 2022. PAPAYA: Practical, Private, and Scalable Federated Learning. In *Proc. of MLSys*, Vol. 4. 814–832.
[7] Angelos Katharopoulos and François and Fleuret. 2018. Not all samples are created equal: Deep learning with importance sampling. In *Proc. of the PMLR ICML*. 2525–2534.
[8] Fan Lai, Xiangfeng Zhu, Harsha V Madhyastha, and Mosharaf Chowdhury. 2021. Oort: Efficient federated learning via guided participant selection. In *Proc. of USENIX OSDI*. 19–35.
[9] Feiyuan Liang, Qinglin Yang, Ruiqi Liu, Junbo Wang, Kento Sato, and Jian Guo. 2022. Semi-Synchronous Federated Learning Protocol with Dynamic Aggregation in Internet of Vehicles. *IEEE Transactions on Vehicular Technology* (2022).
[10] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. 2020. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials* 22, 3 (2020), 2031–2063.
[11] Jianchun Liu, Hongli Xu, Lun Wang, Yang Xu, Chen Qian, Jinyang Huang, and He Huang. 2021. Adaptive Asynchronous Federated Learning in Resource-Constrained Edge Computing. *IEEE Transactions on Mobile Computing* (2021).
[12] Jianchun Liu, Hongli Xu, Yang Xu, Zhenguo Ma, Zhiyuan Wang, Chen Qian, and He Huang. 2021. Communication-efficient asynchronous federated learning in resource-constrained edge computing. *Computer Networks* 199 (2021), 108429.
[13] Bing Luo, Wenli Xiao, Shiqiang Wang, Jianwei Huang, and Leandros Tassiulas. 2022. Tackling system and statistical heterogeneity for federated learning with adaptive client sampling. In *Proc. of INFOCOM*. IEEE, 1739–1748.
[14] Qianpiao Ma, Yang Xu, Hongli Xu, Zhida Jiang, Liusheng Huang, and He Huang. 2021. FedSA: A semi-asynchronous federated learning mechanism in heterogeneous edge computing. *IEEE Journal on Selected Areas in Communications* 39, 12 (2021), 3654–3672.
[15] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proc. of AISTATS*. PMLR, 1273–1282.
[16] Kang Loon Ng, Zichen Chen, Zelei Liu, Han Yu, Yang Liu, and Qiang Yang. 2021. A multi-player game for studying federated learning incentive schemes. In *Proc. of IJCAI*. 5279–5281.
[17] Nang Hung Nguyen, Phi Le Nguyen, Thuy Dung Nguyen, Trung Thanh Nguyen, Duc Long Nguyen, Thanh Hung Nguyen, Huy Hieu Pham, and Thao Nguyen Truong. 2022. FedDRL: Deep Reinforcement Learning-based Adaptive Aggregation for Non-IID Data in Federated Learning. In *Proc. of ICPP*. 1–11.
[18] Takayuki Nishio and Ryo Yonetani. 2019. Client selection for federated learning with heterogeneous resources in mobile edge. In *Proc. of IEEE ICC*. 1–7.
[19] Jinlong Pang, Jieling Yu, Ruiting Zhou, and John CS Lui. 2022. An incentive auction for heterogeneous client selection in federated learning. *IEEE Transactions on Mobile Computing* (2022).
[20] Laércio Lima Pilla. 2021. Optimal task assignment for heterogeneous federated learning devices. In *Proc. of IPDPS*. IEEE, 661–670.
[21] Conghui Tan, Di Jiang, Jinhua Peng, Xueyang Wu, Qian Xu, and Qiang Yang. 2021. A de novo divide-and-merge paradigm for acoustic model optimization in automatic speech recognition. In *Proc. of IJCAI*. 3709–3715.
[22] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In *Proc. of the IEEE/CVF CVPR*.
[23] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. 2019. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications* 37, 6 (2019), 1205–1221.
[24] Yue Wang, Xiaofeng Tao, Xuefei Zhang, Ping Zhang, and Y Thomas Hou. 2019. Cooperative task offloading in three-tier mobile computing networks: An ADMM framework. *IEEE Transactions on Vehicular Technology* 68, 3 (2019), 2763–2776.
[25] Joel Wolfrath, Nikhil Sreekumar, Dhruv Kumar, Yuanli Wang, and Abhishek Chandra. 2022. Haccs: Heterogeneity-aware clustered client selection for accelerated federated learning. In *Proc. of IPDPS*. IEEE, 985–995.
[26] Wentai Wu, Ligang He, Weiwei Lin, Rui Mao, Carsten Maple, and Stephen Jarvis. 2020. SAFA: A semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Trans. Comput.* 70, 5 (2020), 655–668.
[27] Cong Xie, Sanmi Koyejo, and Indranil Gupta. 2020. Asynchronous federated optimization. In *Proc. of OPT*.
[28] Chaoqun You, Daquan Feng, Kun Guo, Howard H Yang, Chenyuan Feng, and Tony QS Quek. 2022. Semi-Synchronous Personalized Federated Learning over Mobile Edge Networks. *IEEE Transactions on Wireless Communications* (2022).
[29] Liangkun Yu, Xiang Sun, Rana Albelaihi, and Chen Yi. 2022. Latency Aware Semi-synchronous Client Selection and Model Aggregation for Wireless Federated Learning. *arXiv preprint arXiv:2210.10311* (2022).
[30] Chendi Zhou, Hao Tian, Hong Zhang, Jin Zhang, Mianxiong Dong, and Juncheng Jia. 2021. TEA-fed: time-efficient asynchronous federated learning for edge computing. In *Proc. of CF*. 30–37.
[31] Ruiting Zhou, Ruobei Wang, Jieling Yu, Bo Li, and Yuqing Li. 2023. Heterogeneous Federated Learning for Balancing Job Completion Time and Model Accuracy. In *Proc. of ICPADS*. IEEE, 562–569.
[32] Feng Zhu, Jiangshan Hao, Zhong Chen, Yanchao Zhao, Bing Chen, and Xiaoyang Tan. 2022. STAFL: Staleness-Tolerant Asynchronous Federated Learning on Non-iid Dataset. *Electronics* 11, 3 (2022), 314.
[33] Hongbin Zhu, Miao Yang, Junqian Kuang, Hua Qian, and Yong Zhou. 2022. Client selection for asynchronous federated learning with fairness consideration. In *Proc. of ICC Workshops*. IEEE, 800–805.
[34] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. 2008. Maximum entropy inverse reinforcement learning.. In *Proc. of AAAI*, Vol. 8. Chicago, IL, USA, 1433–1438.