Asynchronous Online Federated Learning for Edge Devices with Non-IID Data

Yujing Chen,[1] Yue Ning,[2] Martin Slawski,[3] Huzefa Rangwala[1]
[1]*Department of Computer Science, George Mason University*
[2]*Department of Computer Science, Stevens Institute of Technology*
[3]*Department of Statistics, George Mason University*
Email: ychen37@gmu.edu, yue.ning@stevens.edu, mslawsk3@gmu.edu, rangwala@gmu.edu

***Abstract*—Federated learning (FL) is a machine learning paradigm where a shared central model is learned across distributed devices while the training data remains on these devices. Federated Averaging (*FedAvg*) is the leading optimization method for training non-convex models in this setting with a synchronized protocol. However, the assumptions made by FedAvg are not realistic given the heterogeneity of devices. First, the volume and distribution of collected data vary in the training process due to different sampling rates of edge devices. Second, the edge devices themselves also vary in latency and system configurations, such as memory, processor speed, and power requirements. This leads to vastly different computation times. Third, availability issues at edge devices can lead to a lack of contribution from specific edge devices to the federated model. In this paper, we present an Asynchronous Online Federated Learning (ASO-Fed) framework, where the edge devices perform online learning with continuous streaming local data and a central server aggregates model parameters from clients. Our framework updates the central model in an asynchronous manner to tackle the challenges associated with both varying computational loads at heterogeneous edge devices and edge devices that lag behind or dropout. We perform extensive experiments on a benchmark image dataset and three real-world datasets with non-IID streaming data. The results demonstrate ASO-Fed converging fast and maintaining good prediction performance.**

***Index Terms*—Asynchronous federated Learning, Online learning, Edge devices, Non-IID data**

I. Introduction

As massive data is generated from modern edge devices (e.g., mobile phones, wearable devices, and GPS), distributed model training over a large number of computing nodes has become essential for many machine learning applications. With increasing popularity and computation power of these edge devices, federated learning (FL) has emerged as a potentially viable solution to enable the training of statistical models locally on the devices [1]–[3]. FL involves training a shared central model from a federation of distributed devices under the coordination of a central server; while the training data is kept on the edge device. Each edge device performs training on its local data and updates model parameters to the server for aggregation. Many applications can leverage this FL framework such as learning activities of mobile device users, forecasting weather pollutants, and predicting health events.

Many prior FL approaches use a synchronous protocol (e.g., *FedAvg* [1] and its extensions [2]–[6]), where at each global iteration, the server distributes the central model to a selected portion of devices (clients) and aggregates all updates from these clients by applying a weighted averaging strategy.[1] These methods are costly due to a synchronization step (shown in Figure 1), where the server needs to wait for all local updates before aggregation [7]. The existence of lagging devices (i.e., stragglers, stale workers) is inevitable due to device heterogeneity and network unreliability. To address this problem, asynchronous federated learning methods [8], [9] were proposed, where the server can aggregate without waiting for the lagging devices. However, these asynchronous frameworks assume that the number of data samples on each device will not change during the training process, which is not practical in real-life settings. Data on local devices may increase during training, since sensors on these distributed devices usually have a high sampling frequency. In addition, Non-IID (i.e., not independent and identically distributed) and highly imbalanced characteristics of device data create challenges for effective model training [10]. In this paper, we focus on the question: *Can we develop an asynchronous online federated learning framework with a convergence rate guarantee while maintaining an optimal prediction performance?*

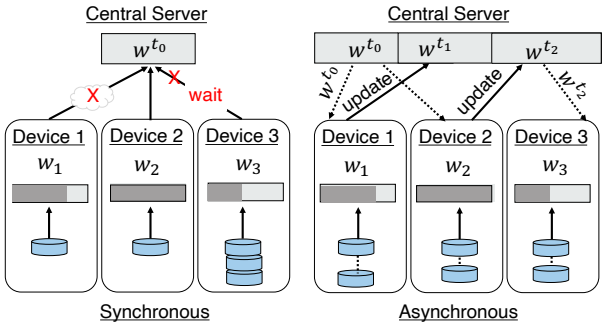


Fig. 1. Illustration of Synchronous vs. Asynchronous update. In synchronous optimization, Device 1 has no network connection and Device 3 needs more computation time, thus the central server has to wait. Asynchronous updates do not need to wait.

We propose an asynchronous online federated learning framework (ASO-Fed), where the central model does not wait for collecting and aggregating the gradient information

978-1-7281-6251-5/20/$31.00 ©2020 IEEE

[1]We use 'client' and 'device' interchangeably in this paper.

from lagging clients, and clients perform online learning to deal with local streaming data. By design, ASO-Fed enables wait-free computation and communication, which ensures the model always converges better than synchronized FL frameworks. This study focuses on improving prediction performance and computation efficiency in FL instead of communication costs or privacy issues. ASO-Fed shares similar privacy benefits as other general FL algorithms [1]–[4] as the data does not leave the edge devices.

In practice, we find that ASO-Fed is particularly useful for streaming data with heterogeneous devices having different computing/communication speeds. Besides the prediction performance, we also simulate different network delays for each device to show the computational efficiency of ASO-Fed. We summarize the main contributions of this paper as follows:

- We propose an asynchronous federated learning framework (ASO-Fed) under a non-IID setting that allows updates from clients with continuously arriving data. The proposed framework learns inter-client relatedness effectively using regularization and a central feature learning module. We provide theoretical guarantees for the convergence of this proposed model.
- We design a novel online learning procedure with a decay coefficient to balance the previous and current gradients on clients when handling streaming data.
- We introduce a dynamic learning strategy for step size adaptation on local devices to mitigate the impact of stragglers on the convergence of the central model. We show empirically that ASO-Fed is robust against data heterogeneity and network connections with high communication delays between the server and some clients.
- We conduct extensive experiments on real-world and benchmark datasets, including a comparison with other state-of-the-art models. The results demonstrate that the proposed method achieves competitive prediction performances and converges fast with lower computation cost.

II. Related Work

A. Distributed Optimization

As massive data are generated from edge devices such as mobile phones, wearable devices, and sensors, the computing power on these devices is also growing rapidly. Learning models directly on these distributed devices is gaining an increasing amount of attention. Multi-task learning models are not suitable for edge device training given that they assume all clients (devices) participate in each training round. This requires that all clients are available because each client is training an individual specific model [11]. However, edge devices could be frequently offline during the training process due to unreliable networks or other factors. Parameter servers, where server nodes maintain globally shared parameters with data distributed on local nodes, however, often suffer from problems such as high network bandwidth or communication overhead [1], [12], [13]. Due to problems with stragglers and the non-IID character of edge device data, numerous other distributed optimization methods [12], [14]–[18] in recent years are also not suitable for on-device learning. Federated Learning provides a promising solution that is capable of dealing with heterogeneous devices across adhoc communication networks.

Federated optimization methods have shown significant improvements on balancing communication versus computation over traditional distributed approaches [19], [20]. Federated learning was first introduced by McMahan *et al.* [1] and has been benchmarked on image and language datasets. Many extensions have been explored based on this original federated learning setting [2]–[4], [21], [22]. A better approach to deal with non-IID data distribution is proposed by sharing a small amount of data with other devices [23]. However, all these studies update the federated model in a synchronous fashion and do not tackle the problem of stragglers and dropouts.

Smith *et al.* [10] developed a primal-dual optimization method within a multi-task learning paradigm. This involved learning separate models for each device and dealing with stragglers. However, this approach was not suitable for non-convex formulations (e.g., deep learning), where strong duality is no longer guaranteed. Xie *et al.* [8] proposed an asynchronous update procedure for federated optimization by updating the central model with weighted averaging, but this did not consider real-world scenarios where edge devices faced continuous streaming data. Our proposed model assumes no constraints on the server aggregation procedure and obtains the optimal prediction performance on clients with heterogeneous data. In addition, we also incorporate online learning on clients to leverage the continuous arrival of new data points.

B. Online Learning with Multiple Clients

Online learning methods operate on a group of data instances that arrive in a streaming fashion. Most existing work in online learning across multiple clients are within the multi-task learning paradigm. Each client seeks to learn an individual model, in conjunction with related clients. The online learning problem with multiple tasks was first introduced by Dekel *et al.* [24]. The relatedness of participated tasks is captured by a global loss and aims to reduce the cumulative loss over multiple rounds. To better model task relationships, Lugosi *et al.* [25] impose a hard constraint on K simultaneous actions taken by the learner in the expert setting; Agarwal *et al.* [26] use matrix regularization and Murugesan *et al.* [27] learn task relationship matrix automatically from the data. All these methods are proposed with synchronized protocols and not adaptable for real-world asynchronous learning.

Jin *et al.* [28] developed an asynchronous distributed framework to perform local training and central learning alternatively with a soft confidence-weighted classifier. However, it assumes that the local data is normally distributed, which is restrictive for non-convex neural network objectives. Besides, it lacks theoretical convergence guarantees and also requires each client to send a portion of its local data to the server.

Different from the above online learning approaches, we design an iterative local computation procedure to balance the previous and current gradients.

III. Definitions and Preliminaries

In this section, we first present the general form of federated learning. Then we briefly introduce the commonly used *FedAvg* [1] and identify the issues in synchronized federated settings.

Assume that we have K distributed devices. Let $\mathcal{D}_k$ denote data captured on device k, and define $n_k = |\mathcal{D}_k|$ to be the number of samples on device k. We denote $N = \sum_{k=1}^{K} |\mathcal{D}_k|$ as the total number of samples in K devices. Assuming for any $k \neq k^{'}$, $\mathcal{D}_k \bigcap \mathcal{D}_{k'} = \varnothing$. We then define local empirical loss of client k as:

$$f_k(w_k) \overset{def}{=} \frac{1}{n_k} \sum_{i \in \mathcal{D}_k} \ell_i(x_i, y_i; w_k). \tag{1}$$

where $\ell_i(x_i, y_i; w_k)$ is the corresponding loss function for data point $\{x_i, y_i\}$ and w_k is the local model parameter. We can obtain the following central objective function:

$$F(w) = \sum_{k=1}^{K} \frac{n_k}{N} f_k(w). \tag{2}$$

where w is the aggregated central model.[2] The overall goal is to find a model w_* with:

$$w_* = \arg\min F(w). \tag{3}$$

A. Synchronized Federated Optimization

As shown in Algorithm 1, for *FedAvg*, at each global iteration, a subset of the devices are selected to run gradient descent optimization (e.g., SGD) locally to optimize the local objective function f_k on device k. Then these local devices communicate their local model updates to the server for aggregation. With heterogeneous local objectives f_k, carefully tuning of local epochs is crucial for *FedAvg* to converge. However, a larger number of local epochs may lead each device towards the optima of its local objective as opposed to the central objective. Besides, data continue to be generated on local devices which increases local gradient variations relative to the central model. Therefore, we incorporate a constraint to restrict the amount of local deviation by penalizing large changes from the current model at the server. We explain this in detail in Section IV-B.

Most synchronized federated optimization methods have a similar update structure as *FedAvg*. One apparent disadvantage of this structure is that, at each global iteration, when one or more clients are suffering from high network delays or clients which have more data and need longer training time, all the other clients must wait. Since the server aggregates after all clients finish, the extended period of waiting time in a synchronized optimization protocol will lead to idling and wastage of computing resources [29], [30].

IV. Proposed Method

We propose to perform asynchronous online federated learning where the server begins to update the central model w after receiving updates from one client, without waiting for the other clients to finish. The server maintains the current central model w, while all clients maintain their own copies (w_k) of w in the memory. Note that the copy of w at one client may be different from the copies at other clients.

[2]We use w and w_k to represent the central model and client model, respectively

Algorithm 1 Algorithm for FedAvg

Input: K indexed by k, local minibatch size B, local epochs E and learning rate η.
Central Server:
for global iterations $t = 1, 2, ..., T$ **do**
  Server chooses a subset S_t of K devices at random
  for each client $k \in S_t$ in parallel **do**
    $w_k^{t+1} \leftarrow \text{ClientUpdate}(k, w^t)$
  $w^{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{N} w_k^{t+1}$
ClientUpdate(k, w^t)**:**
device k updates w^t for E epochs of SGD on f_k with η
return w_k^{t+1} to server

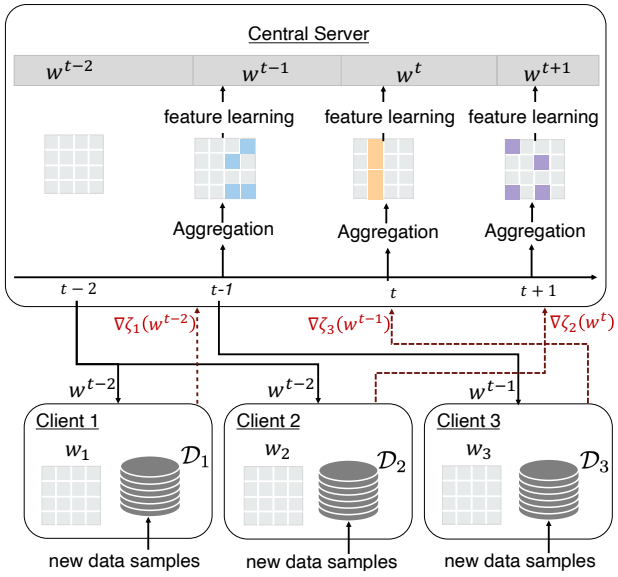


Fig. 2. Illustration of update procedure for the proposed **ASO-Fed** model. Server aggregates after receiving update from one client, and local clients may have new data samples during the training process. Each w is used to represent the whole central/local model, $\nabla\zeta$ is the gradient of local client.

Figure 2 illustrates the update procedure for ASO-Fed. The server starts aggregation after receiving one client's update, and performs feature learning on the aggregated parameters to extract a cross-client feature representation. Then the server starts the next iteration and distributes the new central model to the ready clients. Clients may have new data samples during the training process. To better capture the inter-client relatedness, we use a decay coefficient to balance the previous and current local gradients with an iterative local computation procedure. The approach of ASO-Fed is detailed in Algorithm 2. We will explain each part in detail in the following sections.

As an example in Figure 2, when the server node receives the gradient uploaded from the lagging clients (e.g., Client 2), it has already updated the central model twice. We can

observe that there is an inconsistency in the asynchronous update scheme when it comes to obtaining model parameters from the server. Such inconsistency is common in the real-world settings and is caused by data and system heterogeneity, or network delay. We address this problem by learning a global feature representation on the server and using a dynamic learning step size for training local clients.

A. Learning on Central Server

The server aggregates the central model w after each global iteration. At global iteration $t+1$, assume the server receives an update from client k. Let w^{t+1} be the server model, w_k^t be the local model of client k at iteration t, $\nabla\zeta_k$ be the gradient on the local data of client k, η_k^t be the learning rate of client k and $N' = n_1 + \cdots + n_k' + \cdots + n_K$ be the current total number of data samples where n_k and N becomes n_k' and N' due to new data at client k. By aggregating the update from client k, the server update is computed as follows:

$$\begin{aligned} w^{t+1} &= w^t - \frac{n_k'}{N'}(w_k^t - w_k^{t+1}) \\ &= w^t - \frac{n_k'}{N'}(w_k^t - (w_k^t - \eta_k^t \nabla\zeta_k(w^t))) \\ &= w^t - \eta_k^t \frac{n_k'}{N'} \nabla\zeta_k(w^t). \end{aligned} \tag{4}$$

Feature Representation Learning on Server. To address the potential effect on model performance caused by asynchronous updates, we apply feature representation learning on the server to extract a cross device feature representation. Attention mechanisms have shown to be effective in identifying key features and their representations [31], [32]. Our feature learning approach is inspired by this, and additionally, we combine weight normalization to reduce the computation cost [33], [34]. We use simple network architectures in this study so that it can be easily handled by mobile devices. We apply feature extraction on the first layer (e.g., LSTM or CNN in this paper) after the input to generate the feature representation, and denote the parameters of this layer as $w_{(1)}^{t+1}$. For each element $w_{(1)}^{t+1}[i,j]$ in column $w_{(1)}^{t+1}[j]$ of $w_{(1)}^{t+1}$, we adopt the below operations to obtain the updated $w_{(1)}^{t+1}$:

$$\alpha_{(1)}^{t+1}[i,j] \leftarrow \frac{\exp(|w_{(1)}^{t+1}[i,j]|)}{\sum_j \exp(|w_{(1)}^{t+1}[i,j]|)}, \tag{5}$$

$$w_{(1)}^{t+1}[i,j] = \alpha_{(1)}^{t+1}[i,j] * w_{(1)}^{t+1}[i,j]. \tag{6}$$

B. Learning on Local Clients

In order to mitigate the deviations of the local models from the central model, instead of just minimizing the local function f_k, device k applies a gradient-based update using the following surrogate objective s_k:

$$s_k(w_k) = f_k(w_k) + \frac{\lambda}{2}||w_k - w||^2. \tag{7}$$

Local Update with Decay Coefficient. Data continue arriving at local clients during the training process, so each client needs to perform online learning. For this process, each client requests the latest model from the server and updates the model with its new data. Thus there needs to be a balance between previous model and current model. At global iteration t, device k receives model w^t from the server. Let $\nabla s_k^{(\text{pre})}$ be the previous local gradients, the optimization of device k at this iteration is formulated as:

$$\nabla\zeta_k \leftarrow \nabla s_k - \nabla s_k^{(\text{pre})} + h_k^{(\text{pre})}, \tag{8}$$

$$h_k^{(\text{pre})} = \beta h_k^{(\text{pre})} + (1-\beta)\nabla s_k^{(\text{pre})}. \tag{9}$$

where $h_k^{(\text{pre})}$ is used to balance the previous and current local gradients and initialized to be 0, β is the decay coefficient to balance the previous model and the current model. The update procedure of $h_k^{(\text{pre})}$ can be found in Algorithm 2.

With η_k^t being the learning rate for client k, the closed form solution for model update of client k is given by:

$$\begin{aligned} w_k^{t+1} &= w_k^t - \eta_k^t \nabla\zeta_k(w^t) \\ &= w_k^t - \eta_k^t \Big(\nabla f_k(w_k^t) - \nabla s_k^{(\text{pre})} + h_k^{(\text{pre})} + \lambda(w_k^t - w^t) \Big). \end{aligned} \tag{10}$$

Dynamic Learning Step Size. In real-world settings, the activation rates, i.e., how often clients provide updates to the central model, vary due to a host of reasons. Devices with low activation rates are referred as stragglers, which are caused by several reasons such as communication bandwidth, network delay or data heterogeneity. Thus, we apply a dynamic learning step size with the intuition that if a client has more data or poor communication bandwidth, the activation rate of this client towards the central update will be small and thus the corresponding learning step size should be large. Dynamic learning step sizes are used in asynchronous optimization to achieve better learning performance [35], [36]. Initially, we set $\eta_k^t = \bar{\eta}$ for all clients. The update process (10) can be revised as:

$$w_k^{t+1} = w_k^t - r_k^t \bar{\eta} \nabla\zeta_k(w^t). \tag{11}$$

Algorithm 2 Algorithm for ASO-Fed

Input: Multiple related learning clients distributed at client devices, regularization parameter λ, multiplier r_k, learning rate $\bar{\eta}$, decay coefficient β.
Initialize: $h_k^{pre} = h_k = 0$, $v_k = 0$
Procedure at Central Server
for global iterations $t = 1, 2, ..., T$ **do**
  /* get the update on w^t */
  compute w^t ▷ [Eq.(4)]
  update w^t with feature learning ▷ [Eq.(5) - Eq.(6)]
end for
Procedure of Local Client k at round t
receive w^t from the server
Compute ∇s_k
Set $h_k^{(\text{pre})} = h_k$
Set $\nabla\zeta_k \leftarrow \nabla s_k - \nabla s_k^{(\text{pre})} + h_k^{(\text{pre})}$ ▷ [Eq.(7) -Eq.(10)]
Update $w_k^{t+1} \leftarrow w_k^t - r_k^t \bar{\eta} \nabla\zeta_k$
Compute and update $h_k = \beta h_k + (1-\beta) v_k$
Update $v_k = \nabla s_k(w^t; w_k^t)$
upload w_k^{t+1} to the server

where r_k^t is a time related multiplier, and is given by $r_k^t = \max\{1, \log(\bar{d}_k^t)\}$, where $\bar{d}_k^t = \frac{1}{t}\sum_{\tau=1}^{t} d_k^\tau$ is the average time cost of the past t iterations. Then the actual learning step size is scaled by the past communication delays. This dynamic learning step size strategy can reduce the effect of stragglers on model convergence. Since the stragglers usually have longer delays, the larger step sizes are assigned to these lagging clients to compensate for the loss.

C. Convergence Analysis

In this section, we prove theoretical analysis on the convergence of ASO-Fed. First, we introduce some definitions and assumptions for our convergence analysis.

Definition 1. *(Smoothness) The function f has Lipschitz continuous gradients with constant $L > 0$ (in other words, f is L-smooth) if $\forall x_1, x_2$,*

$$f(x_1) - f(x_2) \leq \langle \nabla f(x_2), x_1 - x_2 \rangle + \frac{L}{2}||x_1 - x_2||^2. \quad (12)$$

Definition 2. *(Strong convexity) The function f is μ-strongly convex with $\mu > 0$ if $\forall x_1, x_2$,*

$$f(x_1) - f(x_2) \geq \langle \nabla f(x_2), x_1 - x_2 \rangle + \frac{\mu}{2}||x_1 - x_2||^2. \quad (13)$$

In order to quantify the dissimilarity between devices in a federated network, following Li *et al* [37], we define the following definition on local non-IID data.

Definition 3. *(Bounded gradient dissimilarity): The local functions ζ_k are V-locally dissimilar at w if $\mathbb{E}||(\nabla \zeta_k(w))||^2 \leq ||\nabla F(w)||^2 V^2$.*

With Definition 3 we further define $V(w) = \sqrt{\frac{\mathbb{E}||\nabla F(w)||^2}{||(\nabla \zeta_k(w))||^2}}$ when $||(\nabla \zeta_k(w))||^2 \neq 0$. When all the local functions are the same, which is the samples on all the devices are in IID fashion, we have $V(w) \to 1$ for all w [37]. However, in federated setting with heterogeneous data, we often have $V > 1$ due to device discrepancies. Therefore, $V \geq 1$, and the larger V is, the larger the dissimilarity among the local functions, which is the more heterogeneous the local data.

Further, we make the following assumptions on the objective functions and introduce one lemma.

Assumption 1. *Suppose that:*
1. The central objective function $F(w)$ is bounded from below, i.e., $F_{\min} = F(w_*) > -\infty$.
2. There exists $\epsilon > 0$ such that $\mathbb{E}(\nabla \zeta_k(w)) \leq ||\nabla F(w)||$, and $\nabla F(w)^\top \mathbb{E}(\nabla \zeta_k(w)) \geq \epsilon ||\nabla F(w)||^2$ holds for all w.

Note that if $\epsilon = 1$, then $\nabla \zeta_k(w)$ is an unbiased estimator of $\nabla F(w)$.

Lemma 1. *If $F(w)$ is μ-strongly convex, then with Assumption 1.1, we have:*

$$2\mu(F(w^t) - F(w_*)) \leq ||\nabla F(w^t)||^2. \quad (14)$$

While the proof of **Lemma 1** is supported by the literature [38], [39], we also provide a detailed proof in Appendix A.

Theorem 1. *Suppose that the central objective function $F(w)$ is L-smooth and μ-strongly convex. Assume the local functions ζ_k are bounded dissimilar. Let Assumption 1 hold. Let $\bar{\eta}_k \leq \eta_k^t < \eta_k = \frac{2\epsilon N'}{LV^2 n_k'}$, then after T global updates on the server, ASO-Fed converges to a global optimum w_*:*

$$\mathbb{E}(F(w^T) - F(w_*)) \leq (1 - 2\mu\gamma'\bar{\eta}_k)^T (F(w^0) - F(w_*)) \quad (15)$$

where $\gamma' = \epsilon - \frac{L\eta_k V^2}{2}$.

The detailed proof of Theorem 1 is provided in Appendix B. Theorem 1 converges under the special case of convex central loss and gives an error bound for the general form of model aggregation.

Theorem 2. *Suppose that the central objective function $F(w)$ is L-smooth and non-convex. Let Assumption 1 hold. Assume the local functions ζ_k are bounded dissimilar. If it holds that $\eta_k^t < \frac{2\epsilon - 1}{LV^2} \leq \max(r_k^t \bar{\eta}) = \bar{\eta}$ for all t, then after T global iterations, we have*

$$\sum_{t=0}^{T-1} \frac{\eta_k^t}{2} \mathbb{E}(||\nabla F(w^t)||^2) \leq F(w^0) - F(w_*) \quad (16)$$

We direct the reader to Appendix C for a detailed proof of Theorem 2. The model convergence rate can be controlled with a balance between the bounded gradient dissimilarity value V and the learning rate η_k^t.

V. Experimental Setup

We perform extensive experiments on three real-world datasets and one benchmark dataset (Fashion-MNIST).

A. Datasets

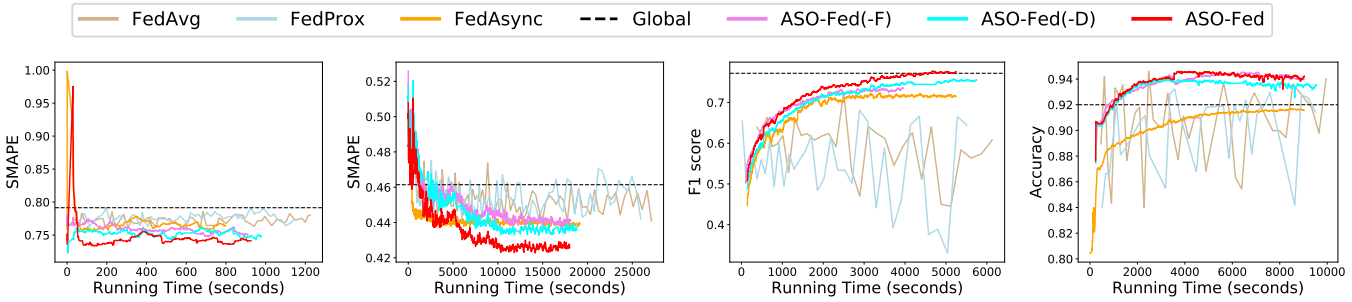
- **FitRec Dataset[3]:** User sport records generated on mobile devices and uploaded to Endomondo, including multiple sources of sequential sensor data such as heart rate, speed, and GPS as well as the sport type (e.g., biking, hiking). Following [40], we re-sampled the data in 10-second intervals, and further generated two derived sequences: derived distance and derived speed. We use data of randomly selected 30 users (clients) for heart rate and speed prediction, and data of each user has features of one sport type.
- **Air Quality Dataset[4]:** Air quality data collected from multiple weather sensor devices distributed in 9 locations of Beijing with features such as thermometer and barometer. Each area is modeled as a separate client and the observed weather data is used to predict the measure of six air pollutants (e.g., PM2.5).
- **ExtraSensory Dataset[5]:** Mobile device sensor data (e.g., high-frequency motion-reactive sensors, location services, audio, watch compass) and watch sensor data (accelerator) collected from 60 users; performing any of 51 activities [41]. We use the provided 225-length feature vectors of time and frequency domain variables generated for each instance.

[3]https://sites.google.com/eng.ucsd.edu/fitrec-project/home
[4]https://biendata.com/competition/kdd_2018/data/
[5]http://extrasensory.ucsd.edu/

TABLE IV.1

PREDICTION PERFORMANCE COMPARISON. BOLD NUMBERS ARE THE BEST PERFORMANCE, NUMBERS WITH UNDERLINES ARE THE SECOND BEST VALUES. IMPROV.(1) SHOWS THE PERCENTAGE IMPROVEMENT OF ASO-FED OVER FEDAVG. IMPROV.(2) SHOWS THE PERCENTAGE IMPROVEMENT OF ASO-FED OVER THE BEST BASELINE RESULTS.

Method	FitRec				Air Quality		ExtraSensory				Fashion-MNIST
	MAE ↓ (Speed)	SMAPE↓ (Speed)	MAE↓ (HeartRate)	SMAPE↓ (HeartRate)	MAE↓	SMAPE↓	F1↑	Precision↑	Recall↑	BA↑	Accuracy↑
FedAvg	13.61	0.78	13.72	0.78	44.30	0.44	0.66	0.87	0.55	0.77	0.87
FedProx	14.21	0.82	14.53	0.83	44.30	0.44	0.67	0.82	0.57	0.77	0.88
FedAsync	13.56	0.78	13.67	0.78	37.98	0.43	0.72	0.84	0.65	0.82	0.90
Local-S	12.76	0.75	13.27	0.76	36.72	0.56	0.65	0.72	0.61	0.79	0.89
Global	12.95	0.78	12.79	0.79	37.61	0.44	**0.77**	**0.92**	0.66	0.83	0.92
ASO-Fed(-D)	12.46	0.74	12.51	0.75	37.13	0.43	0.76	0.88	0.69	**0.85**	0.94
ASO-Fed(-F)	12.62	0.76	12.71	0.76	37.72	0.43	0.75	0.86	0.68	0.84	0.94
ASO-Fed	**12.31**	**0.73**	**12.36**	**0.74**	**36.71**	**0.42**	**0.77**	0.88	**0.70**	**0.85**	**0.95**
improv.(1)	9.55%	6.41%	9.91%	5.13%	17.13%	2.32%	16.66%	1.15%	27.27%	10.39%	9.19%
improv.(2)	3.52%	2.67%	3.36%	2.63%	0.03%	4.54%	0.00%	-4.34%	6.06%	2.41%	3.26%



(a) FitRec (SMAPE ↓) (b) Air Quality (SMAPE ↓) (c) ExtraSensory (F1-score ↑) (d) Fashion-MNIST (Accuracy ↑)

Fig. 3. Test set performance vs. running time for four datasets. Lower SMAPE value indicates better model performance. For the synchronized federated frameworks, we plot results of *FedAvg* and *FedProx* at every 10 global iterations.

We model device of each user as a client and predict their activities (e.g., walking, talking, running).

- **Fashion-MNIST:** This is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes (e.g., Dresses, Coat, Bag). Each class has the same number of examples. We follow a non-IID setting as in *FedAvg* [1] and divide the data into 20 parts according to their labels. We first sort the data by category label, divide each category into 4 different sizes $\{2000, 2750, 3250, 4000\}$, and assign each of 20 parts 2 different sizes. We model each part as a separate client and predict the target labels.

B. Comparative Methods

1) Baseline Methods

We compare the proposed ASO-Fed with the following synchronous and asynchronous federated learning approaches, single-client and global models.

- FedAvg [1]–[3]: the commonly used synchronous federated learning approach proposed by McMahan *et al.* [1].
- FedProx [37]: synchronous federated learning framework with a proximal term on the local objective function to mitigate the data heterogeneity problem and to improve the model stability compared to FedAvg.
- FedAsync [8]: asynchronous federated learning framework using a weighted average to update the server model.
- Local-S: each client learns separate model with its own data, and the model structure is the same as ASO-Fed.
- Global: combining the data of all clients and processed in a batch setting on a single machine.

2) Ablation Studies

We also perform ablation studies to study the effect of feature representation learning on server and local dynamic learning step size:

- ASO-Fed(-D): the proposed ASO-Fed without dynamic learning step size.
- ASO-Fed(-F): the proposed ASO-Fed without central feature representation learning.

C. Training Details

For each dataset, we split the each client's data into 60%, 20%, 20% for training, validation, and testing, respectively. As for each client's training data, we start with a random portion of the total training size, and increase by $0.05\% - 0.1\%$ each iteration to simulate the arriving data. We set the fraction C of FedAvg as 0.2, decay coefficient β as 0.001, $\bar{\eta} = 0.001$, $\lambda = 1.0$ for FitRec and Air Quality datasets, $\lambda = 0.8$ for ExtraSensory dataset, and $\lambda = 0.5$ for Fashion-MNIST. For FedAsync model, we set $\gamma = 0.1$, $\rho = 0.005$ and $\alpha = 0.6$. We

use a single layer LSTM followed by a fully connected layer for the three real-world datasets and two CNN layers followed by a max pooling layer for Fashion-MNIST. The local epoch number of each client is set as 2. We design simple network architectures for all datesets so that it can be easily handled by mobile devices. All of the experiments are conducted with two Intel E5-2660 v3 10-core CPU at 2.60GHz [42].

Simulation parameters. To simulate the stragglers and dropouts situations, we set different network settings for our experiments, a random offset parameter ($10 \sim 100$ seconds) was taken as an input from the client. This parameter represents the average delay related to the infrastructure of the network for a client. We direct the reader to Section VI for the detailed results.

VI. Experimental Results

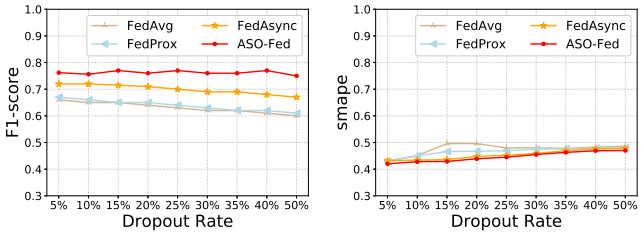
A. Predictive Performance Comparison

Table IV.1 reports the predictive performance comparing ASO-Fed to the baseline approaches. In case of regression problems, we report the average *MAE* and *SMAPE* values, for ExtraSensory classification benchmark we report the average *F1*, *Precision*, *Recall* and *Balanced Accuracy (BA)*, and for Fashion-MNIST we report the *Accuracy*. From Table IV.1, we observe that ASO-Fed achieves the lowest MAE and SMAPE values for FitRec and Air Quality datasets, and has the overall best performance for ExtraSensory and Fashion-MNIST datasets. Across the four datasets, ASO-Fed outperforms the Global model (acquires all data at a single server) by $2.39\% \sim 6.41\%$. FedAvg and FedProx do not perform well on the highly unbalanced and non-IID datasets (FitRec, ExtraSensory and Fashion-MNIST). In particular, for the FitRec and Fashion-MNIST datasets, Local-S (the local single client model) outperforms FedAvg and FedProx.

Figure 3 shows the prediction performance for the different approaches as a function of running time. From Figure 3, we notice large fluctuations on the performance of FedAvg and FedProx during the whole training process on all four benchmarks. This shows that synchronous federated frameworks do not perform well on streaming data with skewed and non-IID data distribution. FedAsync achieves better performance than the two synchronous federated frameworks, but not as good as ASO-Fed. ASO-Fed has steady improvements with running time (and converges quickly).

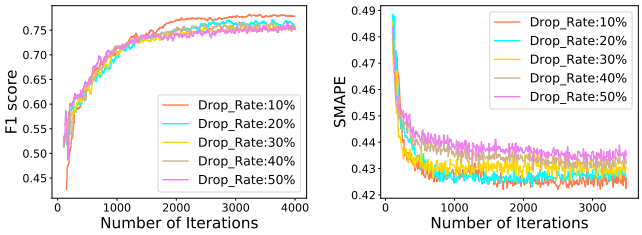
TABLE VI.1
Computation time (in minutes) to reach target test performance. The network delay of each client was set to be a random value between $10 \sim 100$ seconds.

Method	FitRec	Air Quality	ExtraSensory	FMNIST
FedAvg	20.42	460.02	104.86	160.72
FedProx	19.26	439.95	99.45	160.36
FedAsync	15.41	326.45	87.97	151.72
ASO-Fed(-D)	16.31	332.74	95.77	158.83
ASO-Fed(-F)	**15.17**	320.92	**65.87**	150.54
ASO-Fed	15.43	**319.41**	87.40	**150.46**



(a) ExtraSensory (F1-socre ↑) (b) Air Quality (SMAPE ↓)

Fig. 4. Performance comparison of federated approaches as dropout rate of clients increases. ASO-Fed has better performance than the other federated frameworks.



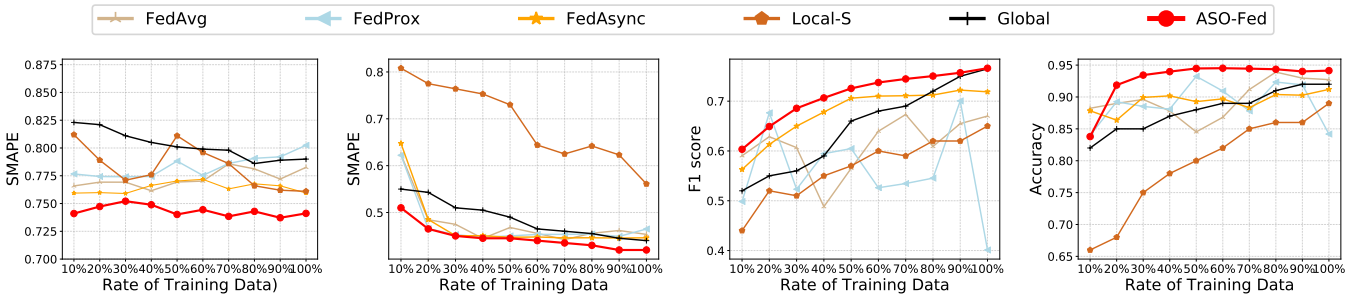
(a) ExtraSensory (F1-score ↑) (b) Air Quality (SMAPE ↓)

Fig. 5. The performance of ASO-Fed with clients periodically dropping out.

Ablation Studies. To evaluate the performance of central feature representation learning and local dynamic learning step size, we perform ablation studies with two additional models: ASO-Fed(-F) and ASO-Fed(-D). From Table IV.1 we notice that ASO-Fed outperforms ASO-Fed(-F) by $1.06\% \sim 5.26\%$, which shows the effectiveness of central feature learning at generating a better feature representation across clients. We show the visualizations of learned features for the three real-world datasets in Section VI-E. ASO-Fed(-D) has close prediction performance as ASO-Fed, but as shown in Figure 3, ASO-Fed(-D) needs a longer training time to converge than ASO-Fed. This shows that local dynamic learning approach works effectively at lowering the overall computation cost.

B. Evaluation of Time Efficiency

We report the running time of synchronous and asynchronous FL approaches to reach target test performance in Table VI.1. As seen from this table, FedAvg and FedProx have the highest computation cost across all four benchmarks. This is expected given that in synchronous protocol, the server aggregation has to wait for the slow client nodes to finish their computations. Among the comparison of asynchronous update models, ASO-Fed is more time efficient than ASO-Fed(-D). FedAsync is close in running time as ASO-Fed(-D). From the empirical results we note that the dynamic learning step size is a promising strategy and effective when there are significant delays in the network. ASO-Fed is close to ASO-Fed(-F) in terms of computational cost except for the FitRec and ExtraSensory dataset. Because these two datasets have more complex features and additional computation for feature extraction requires more time, but ASO-Fed obtains better prediction performance with a little sacrifice of time efficiency.



(a) FitRec (SMAPE ↓) (b) Air Quality (SMAPE↓) (c) ExtraSensory (F1-socre ↑) (d) Fashion-MNIST (Accuracy ↑)

Fig. 6. Average performance comparison (SMAPE, F1, accuracy) on four datasets as training data increases.

C. Robustness to Stragglers and Dropouts

Stragglers are clients that lag in providing updates to the server due to a variety of reasons: communication bandwidth, computation load, and data variability. In this Section, we investigate a common real-world scenario, when clients have no response during the entire training process or are not available in certain time windows. We refer to these non-contributing clients as dropouts.

We set a certain portion of clients to be non-responsive. These clients will not participate in the training process. However, the reported results are evaluated on test data from all clients. Figure 4 shows the performance of federated learning approaches on Extrasensory classification and Air Quality regression benchmarks with an increasing fraction of clients being dropped from the learning process. As shown in Figure 4, for the ExtraSensory dataset, we observe that as the rate of dropout clients increases, the performance of the FedAvg model also drops. The same trend is noticed for FedProx. As for ASO-Fed, the prediction performance is steady except for a slight decrease when the dropout rate exceeds 40%. Even when 50% of clients are subject to dropout during training, ASO-Fed can still achieve at least a 10% improvement over synchronous FL models and FedAsync. For the Air Quality dataset, ASO-Fed has lower SMAPE errors than all other models as the dropout rate increases and the performance of ASO-Fed is relatively stable. However, as expected, if one of the nodes never sends updates to the central server, the model does not generalize. This explains the poor performance as the dropout rate increases.

We also explore the performance effect of nodes periodically dropping and not providing updates to the server. To the best of our knowledge, no other methods for asynchronous federated learning with local streaming data directly address this issue. Therefore we do not draw comparisons to other methods. At each global iteration, we randomly select a certain fraction of clients as not participating in the training during the current iteration. Figure 5 shows the convergence trend for different rates of periodically dropping clients. We notice that as the rate of periodically dropout clients increases, ASO-Fed still converges well with slightly worse performance. This shows that the performance of ASO-Fed is robust to relatively high rates of periodical dropouts.

D. Results of Varying Training Samples

To evaluate the incremental online learning process more explicitly, we display how the prediction performance changes with an increasing number of training samples in Figure 6. We perform experiments with different rates of all clients' training data and depict the average performance on all clients. From Figure 6, for all datasets, ASO-Fed achieves the best performance with increasing rates of training data. Large fluctuations are observed in the results of FedAvg and FedProx, which exhibit an unstable model performance for synchronous approaches as the local data sets increase. Compared with the Global approach, asynchronous frameworks have more stable performance as the amount of training data increases, while the individual models learned by Local-S do not perform well. The analysis shows that that ASO-Fed learns an effective model with a smaller portion of training data. With increasing local data, ASO-Fed still outperforms the other competitors.

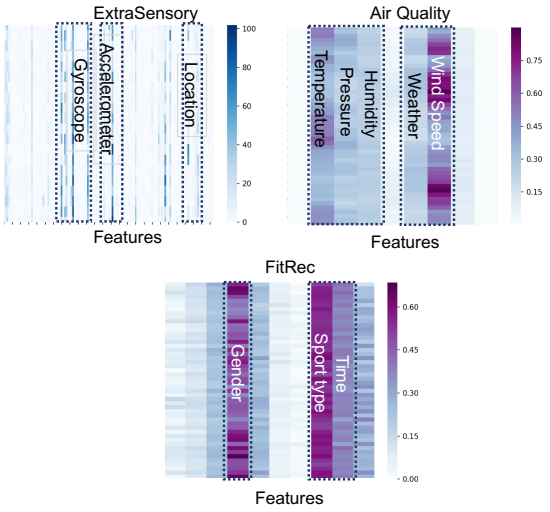


Fig. 7. Feature representation learned on the server of three real-world datasets. Each column is the weights vector within 48 time steps over the input series.

E. Feature Representation learning

In this section, we present the qualitative results of the proposed feature representation learning on the server. In Figure 7, we show the features learned from one client of three datasets respectively. For the client in ExtraSensory, the highlighted

features are 'Gyroscope', 'Accelerometer' and 'Location', and the corresponding labels are 'walking', 'at_home'. For the client from Air Quality dataset, we observe that features with high weights are 'Wind Speed' and 'Temperature'. This makes sense given that the target values are air pollutants (e,g,. PM2.5, SO) and 'Wind Speed' decides whether these pollutants can be dispersed. Air pollutants vary with seasons, and a higher concentration of air pollutants appears in winter time due to fuel consumption for heating in winter. Therefore 'Temperature' is also a strong indicator for air pollutants. For the client from FitRec, the extracted features are 'gender', 'sport type' and 'time'. Since the prediction targets are speed and heart rate, these three features have strong correlations with the targets. The above results show the effectiveness of feature learning in ASO-Fed.

VII. Conclusions and Future Work

We propose a novel asynchronous online federated learning approach to tackle the learning problems on distributed edge devices. To the best of our knowledge, this is the first attempt to solve asynchronous federated learning with local streaming data. Compared to synchronized FL approaches (FedAvg and FedProx), ASO-Fed is computationally efficient since the central server does not need to wait for lagging clients to perform aggregation. Compared with asynchronous approach (FedAsync), the proposed approach achieves better performance on all provided datasets, which indicates that the proposed asynchronous update method can better handle local streaming data. Time efficiency is compared on multiple benchmarks and the results show that the proposed ASO-Fed is faster than synchronized FL. The proposed ASO-Fed inherits the idea of using asynchronous update scheme as [8], [9]. Likewise, it shares the same communication bottleneck problem pointed out in [11]. Nevertheless, it is still an open issue to build communication efficient methods in asynchronous federated learning frameworks. Besides, feature learning component needs longer computation time when dealing with datasets which have complex features. Thus, there is a trade off between the prediction performance and computational costs. In the future, we plan to develop communication efficient federated learning methods with asynchronous updating strategies.

Appendix

A. Proof of Lemma 1

Proof. $F(w)$ is *μ-strongly convex*, we can get:

$$F(w') - F(w^t) \geq \langle \nabla F(w^t), w' - w^t \rangle + \frac{\mu}{2}||w' - w^t||^2, \quad (17)$$

Let us define $C(w')$ such that:

$$C(w') = F(w^t) + \langle \nabla F(w^t), w' - w^t \rangle + \frac{\mu}{2}||w' - w^t||^2, \quad (18)$$

$C(w')$ is a quadratic function of w', then it has minimal value when $\nabla C(w') = \nabla F(w^t) + \mu(w' - w^t) = 0$. Then the minimal value of $C(w')$ is obtained when $w' = w^t - \frac{\nabla F(w^t)}{\mu}$, which is:

$$C_{\min} = F(w^t) - \frac{||\nabla F(w^t)||^2}{2\mu}, \quad (19)$$

For $F(w)$ is *μ-strongly convex*, we can complete the proof:

$$F(w_*) \geq C(w_*) \geq C_{\min} = F(w^t) - \frac{||\nabla F(w^t)||^2}{2\mu}, \quad (20)$$

$$2\mu(F(w^t) - F(w_*)) \leq ||\nabla F(w^t)||^2. \quad (21)$$

□

B. Proof of Theorem 1

Proof. With $F(w)$ is *L-smooth*, we have:

$$\begin{aligned} &F(w^{t+1}) - F(w^t) \leq \langle \nabla F(w^t), w^{t+1} - w^t \rangle + \frac{L}{2}||w^{t+1} - w^t||^2 \\ &= -\nabla F(w^t)^\top \eta_k^t \frac{n_k'}{N'} \nabla \zeta_k(w^t) + \frac{L}{2}||\eta_k^t \frac{n_k'}{N'} \nabla \zeta_k(w^t)||^2, \end{aligned} \quad (22)$$

let $m = \eta_k^t \frac{n_k'}{N'} > 0$, with Assumption 1 and local Bounded gradient dissimilarity, we can get:

$$\begin{aligned} &\mathbb{E}(F(w^{t+1})) - F(w^t) \\ &\leq -m \nabla F(w^t)^\top \mathbb{E}(\nabla \zeta_k(w^t)) + \frac{Lm^2}{2}\mathbb{E}(||\nabla \zeta_k(w^t)||^2) \\ &\leq -m\epsilon||\nabla F(w^t)||^2 + \frac{Lm^2V^2}{2}||\nabla F(w^t)||^2 \\ &= -m(\epsilon - \frac{LmV^2}{2})||\nabla F(w^t)||^2, \end{aligned} \quad (23)$$

We can easily prove that $-m(\epsilon - \frac{LmV^2}{2})$ is monotonically increasing while $m > 0$. Since $n_k' < N'$, thus $m = \eta_k^t \frac{n_k'}{N'} < \eta_k^t$. Then we have $-m(\epsilon - \frac{LmV^2}{2}) < -\eta_k^t(\epsilon - \frac{L\eta_k^t V^2}{2})$.

With Lemma 1, and let $\gamma = \epsilon - \frac{L\eta_k^t V^2}{2}$, we can rewrite Equation (23) as:

$$\mathbb{E}(F(w^{t+1})) - F(w^t) \leq -2\mu\eta_k^t\gamma(F(w^t) - F(w_*)), \quad (24)$$

Then we move $F(w^t)$ on left side to right and subtract $F(w_*)$ from both sides, and get:

$$\mathbb{E}(F(w^{t+1})) - F(w_*) \leq (1 - 2\mu\eta_k^t\gamma)(F(w^t) - F(w_*)), \quad (25)$$

Since $\bar{\eta}_k < \eta_k^t$, then by taking expectation of both sides, and telescoping, we have:

$$\mathbb{E}(F(w^{t+1}) - F(w_*)) \leq (1 - 2\mu\bar{\eta}_k\gamma')(\mathbb{E}(F(w^t) - F(w_*)). \quad (26)$$

When $t+1 = T$, the above inequality becomes Equation (15). Thus we complete the proof. □

C. Proof of Theorem 2

Proof. With $m < \eta_k^t$ and $\gamma = \epsilon - \frac{L\eta_k^t V^2}{2}$, we replace m with η_k^t and take the full expectation of Equation (23), we have:

$$\mathbb{E}(F(w^{t+1})) \leq \mathbb{E}(F(w^t)) - \eta_k^t\gamma\mathbb{E}(||\nabla F(w^t)||^2), \quad (27)$$

Then summing up (27) over global iteration T, we can get:

$$\mathbb{E}(F(w^{t+1})) \leq F(w^0) - \sum_{t=0}^{T-1} \eta_k^t\gamma\mathbb{E}(||\nabla F(w^t)||^2), \quad (28)$$

From Assumption 1.1 we can get $F(w_*) \leq \mathbb{E}(F(w^{t+1}))$, then we have:

$$F(w_*) \leq F(w^0) - \sum_{t=0}^{T-1} \eta_k^t \gamma \mathbb{E}(||\nabla F(w^t)||^2), \tag{29}$$

If we set $\eta_k^t < \frac{2\epsilon-1}{LV^2} \leq max(r_k^t \bar{\eta}) = \bar{\eta}$, we can get $\eta_k^t(\epsilon - \frac{L\eta_k^t V^2}{2}) > \frac{\eta_k^t}{2}$. Rearrange Equation (29) we can get:

$$\begin{aligned}\sum_{t=0}^{T-1} \frac{\eta_k^t}{2} \mathbb{E}(||\nabla F(w^t)||^2) &\leq \sum_{t=0}^{T-1} \eta_k^t(\epsilon - \frac{L\eta_k^t V^2}{2}) \mathbb{E}(||\nabla F(w^t)||^2) \\ &\leq F(w^0) - F(w_*).\end{aligned} \tag{30}$$

Then we get Equation (16) and complete the proof. □

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[2] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[3] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[4] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[5] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018.

[6] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau, "Federated learning for keyword spotting," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[7] T. Chen, G. Giannakis, T. Sun, and W. Yin, "Lag: Lazily aggregated gradient for communication-efficient distributed learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 5050–5060.

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

[9] M. Chen, B. Mao, and T. Ma, "Efficient and robust asynchronous federated learning with stragglers," in *Submitted to International Conference on Learning Representations*, 2019.

[10] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4424–4434.

[11] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[12] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Advances in Neural Information Processing Systems*, 2014, pp. 19–27.

[13] M. Li, L. Zhou, Z. Yang, A. Li, F. Xia, D. G. Andersen, and A. Smola, "Parameter server for distributed machine learning," in *Big Learning NIPS Workshop*, vol. 6, 2013, p. 2.

[14] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, "Asynchronous distributed optimization using a randomized alternating direction method of multipliers," in *52nd IEEE conference on decision and control*. IEEE.

[15] N. Aybat, Z. Wang, and G. Iyengar, "An asynchronous distributed proximal gradient method for composite convex optimization," in *International Conference on Machine Learning*, 2015, pp. 2454–2462.

[16] Y. Zhang, J. C. Duchi, and M. J. Wainwright, "Communication-efficient algorithms for statistical optimization," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 3321–3363, 2013.

[17] S. Zhang, A. E. Choromanska, and Y. LeCun, "Deep learning with elastic averaging sgd," in *Advances in neural information processing systems*, 2015.

[18] V. Smith, S. Forte, C. Ma, M. Takáč, M. I. Jordan, and M. Jaggi, "Cocoa: A general framework for communication-efficient distributed optimization," *The Journal of Machine Learning Research*, 2017.

[19] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, 2011.

[20] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction using mini-batches," *Journal of Machine Learning Research*, vol. 13, no. Jan, pp. 165–202, 2012.

[21] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019.

[22] S. Caldas, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[23] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[24] O. Dekel, P. M. Long, and Y. Singer, "Online multitask learning," in *International Conference on Computational Learning Theory*. Springer, 2006.

[25] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile, "Linear algorithms for online multitask classification," *Journal of Machine Learning Research*, vol. 11, no. Oct, pp. 2901–2934, 2010.

[26] A. Agarwal, A. Rakhlin, and P. Bartlett, "Matrix regularization techniques for online multitask learning," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-138*, 2008.

[27] K. Murugesan, H. Liu, J. Carbonell, and Y. Yang, "Adaptive smoothed online multi-task learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 4296–4304.

[28] X. Jin, P. Luo, F. Zhuang, J. He, and Q. He, "Collaborating between local and global learning for distributed online multiple tasks," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 113–122.

[29] Z. Chai, H. Fayyaz, Z. Fayyaz, A. Anwar, Y. Zhou, N. Baracaldo, H. Ludwig, and Y. Cheng, "Towards taming the resource and data heterogeneity in federated learning," in *2019 USENIX Conference on Operational Machine Learning (OpML 19)*, 2019, pp. 19–21.

[30] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, and Y. Cheng, "Tifl: A tier-based federated learning system," in *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*, 2020, p. 125–136.

[31] O. Firat, K. Cho, and Y. Bengio, "Multi-way, multilingual neural machine translation with a shared attention mechanism," *arXiv preprint arXiv:1601.01073*, 2016.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[33] Y.-M. Wang and T. M. Elhag, "On the normalization of interval and fuzzy weights," *Fuzzy sets and systems*, 2006.

[34] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[35] Y. K. Cheung and R. Cole, "Amortized analysis on asynchronous gradient descent," *arXiv preprint arXiv:1412.0159*, 2014.

[36] I. M. Baytas, M. Yan, A. K. Jain, and J. Zhou, "Asynchronous multi-task learning," in *2016 IEEE 16th International Conference on Data Mining*.

[37] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "Federated optimization for heterogeneous networks," *arXiv preprint arXiv:1812.06127*, vol. 1, no. 2, p. 3, 2018.

[38] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, 2013, vol. 87.

[39] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.

[40] J. Ni, L. Muhlstein, and J. McAuley, "Modeling heart rate and activity data for personalized fitness recommendation," in *The World Wide Web Conference*. ACM, 2019, pp. 1343–1353.

[41] Y. Vaizman, K. Ellis, G. Lanckriet, and N. Weibel, "Extrasensory app: Data collection in-the-wild with rich user interface to self-report behavior," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 2018, p. 554.

[42] D. Duplyakin, R. Ricci, A. Maricq, G. Wong, J. Duerig, E. Eide, L. Stoller, M. Hibler, D. Johnson, K. Webb *et al.*, "The design and operation of cloudlab," in *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, 2019, pp. 1–14.