Received 25 May 2022; revised 21 June 2022; accepted 29 June 2022. Date of publication 7 July 2022; date of current version 22 July 2022.

Digital Object Identifier 10.1109/OJCOMS.2022.3188051

Paving the Way for Distributed Artificial Intelligence Over the Air

GUOQING MA[1] (Graduate Student Member, IEEE), CHUANTING ZHANG[1,2] (Member, IEEE), SHUPING DANG[1] (Member, IEEE), AND BASEM SHIHADA[1] (Senior Member, IEEE)

[1]Computer, Electrical and Mathematical Science and Engineering Division, King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia

[2]University of Bristol, Bristol BS8 1TH, U.K.

CORRESPONDING AUTHOR: B. SHIHADA (e-mail: basem.shihada@kaust.edu.sa)

This work was supported by King Abdullah University of Science and Technology.

ABSTRACT Distributed Artificial Intelligence (DAI) is one of the most promising techniques to provide intelligent services under strict privacy protection regulations for multiple clients. By applying DAI, training on raw data is carried out locally. At the same time, the trained outputs, e.g., model parameters from multiple local clients, are sent back to a central server for aggregation. DAI is recently studied in conjunction with wireless communication networks to achieve better practicality, incorporating various random effects brought by wireless channels. However, because of wireless channels' complex and case-dependent nature, a generic simulator for applying DAI in wireless communication networks is still lacking. To accelerate the development of DAI in wireless communication networks, we propose a generic system design in this paper and an associated simulator that can be set according to wireless channels and system-level configurations. Details of the system design and analysis of the impacts of wireless environments are provided to facilitate further implementations and updates. We employ a series of experiments to verify the effectiveness and efficiency of the proposed system design and reveal its superior scalability.

INDEX TERMS Distributed deep learning (DDL), federated learning (FL), system design, simulator design, wireless environment, convergence analysis.

I. INTRODUCTION

AS SPECULATED in the perspective paper, 'What should 6G be?' [1], sixth-generation (6G) communication networks are expected to be human-centric, posing much higher requirements for privacy protection. On the other hand, based on existing artificial intelligence (AI) architectures, protecting digital privacy is, to some extent, contradictory to the demand for user data by intelligent communication services [2]. This is because user data are required to be collected, processed, and utilized to precisely identify user demands so that truly intelligent and high-quality communication services can be provided to end-users [3]. These user data inevitably contain personal and sensitive information that users are unwilling to share and should be restricted by legislation [4]. Collecting and processing user data by such a centralized architecture could also lead to a high divulging risk, which has become much more common nowadays [5]. Moreover, relying on such a centralized architecture for intelligent communication services, one can never rule out the possibility that a malicious *Big Brother* takes advantage of user data and manipulates users and even the entire society with ulterior motives [6].

To solve the dilemma between high-intelligence communication services and user privacy protection, distributed deep learning (DDL) is proposed. It soon attracted researchers' attention in the communication and computing research communities [7]. The large-scale DDL was first investigated in [8] to solve the insufficient computation ability in a single node, in which a central server aggregates the one-step model gradients updated from all agents with the randomly assigned dataset. However, aggregating the gradients at each stochastic gradient descent (SGD) updating round increases communication overhead [9], constraining it to be only suitable at high-bandwidth data centers. To reduce communication overhead and extend the deployment on edge devices, local SGD [10] has been proposed. Instead of gradients, the multiple clients update model parameters

to the central server for aggregation after a preset local SGD updating steps. However, all clients synchronously updating model parameters makes it unsuitable for the application scenarios under unreliable communications and heterogeneous computing resources. Federated learning (FL) is a further advancement of local SGD [10], by which only a subset of clients will update their model parameters to the central server instead of all clients. Due to the variability of the local steps and the proportion of activated clients, FL is sometimes believed to supersede the concepts of DDL and local SGD. However, the theoretical convergence guarantees of these learning strategies are distinct, leading to varied applicability in practice depending on the reliability of communications and homogeneity of computing devices. Despite subtle differences among these learning strategies, they all belong to the distributed artificial intelligence (DAI) family [11] due to the *decoupling of client training and server aggregation*. Hence, we apply the term DAI instead of carefully distinguishing them.

Different from classical machine learning (ML) or deep learning (DL) techniques adopting centralized processing architectures [12]–[14], DAI utilizes a distributed processing architecture that consists of one DAI server (viz. the model owner) and multiple clients (viz. the data owners) [15]. The clients directly collect users' raw data and process them by local training algorithms to obtain *local model* parameters. These local model parameters are then aggregated in a certain way at the DAI server. The aggregated model produced at the DAI server is called the *global model*, which will subsequently be updated to the clients for intelligent communication services. In this way, the global model training and first-hand raw data accessing can be decoupled, and thereby the data minimization principle for the privacy of consumer data is followed [16].

Due to the distributed processing architecture and exemption from users' raw data, DAI is believed to be one of the most promising techniques to provide intelligent services under strict privacy protection regulations [15], [17]–[19]. In addition, DAI can also facilitate the implementations of other promising 6G communication techniques by releasing privacy concerns and reducing the volume of data required to transmit [10]. Consequently, spectral efficiency, energy efficiency, and latency of communication systems would all be improved by DAI [20].

As described above, DAI computation is performed at both the DAI server and clients, and the exchange of model parameters is frequent and necessary. As a result, the communication and computing procedures of DAI are coupled, which should be jointly considered and analyzed as a whole. Recently, many research works have analyzed both communication and computing issues related to DAI in wireless communication networks [21]–[23] (details of them will be given and reviewed in the next section). However, existing works on wireless communications treat DAI as isolated optimization algorithms in ideal and guaranteed wireless environments. Their objective functions aim at optimizing specific model characteristics, such as transmission time and energy consumption [21]–[23]. To solve the formulated optimization problems, they assume simple wireless communications constraints and specific communication models without considering the effects of unreliable and diversiform communication and computational resources in realistic situations, resulting in a difficulty to be deployed in practice. Meanwhile, a generic system for designing and testing DAI algorithms in wireless communication networks is still lacking, which impedes DAI development in wireless environments and DAI-aided wireless networks. First, without a benchmark system, researchers interested in DAI algorithms implemented in wireless environments need to program individual communication scenarios for investigation. Also, the simulation results provided by DAI can hardly be verified by reproduction and compared with results generated by other benchmark algorithms. At last, even with the increasing awareness of the generic design of DAI systems [24], [25], the researchers neglect the simulations on wireless environments, which proves to be an essential factor in our work.

In this regard, we propose distributed artificial intelligence over-the-air (AirDAI), a generic system design for DAI over the air, aiming at accelerating the relevant research progress on DAI in wireless environments.[1] Compared to existing solutions, the proposed system can be easily adapted to different settings for designing, testing, and investigating DAI applied in different wireless scenarios with more realistic parameter settings. Designers can alter the wireless communication environment and introduce self-defined quality of service (QoS) metrics with our provided simulator to examine newly-designed DAI algorithms and generate reproducible results. The contributions of this paper are listed as follows:

- To ensure generality and practicability, we generalize the system design by considering a series of realistic wireless features, including path loss, shadowing, multipath fading, and mobility.
- We further analyze the convergence rate of DAI applied in wireless environments and affected by a set of stochastic factors.
- We also provide a Python-based simulator according to the proposed system, which can be easily integrated into popular ML and DL frameworks, e.g., PyTorch [26] and TensorFlow [27].
- Moreover, the proposed system design and simulator modules can be customized because of their generic nature.

The rest of the paper is organized as follows. In Section II, we carry out comprehensive literature research on the works related to DAI in wireless communication networks. Summarizing the existing literature and research directions,

1. The codes associated with the proposed system as well as its simulator can be found from the open GitHub repository link: https://github.com/KAUST-Netlab/AirDAI

TABLE 1. Notations and the corresponding descriptions applied in this paper.

Notations	Descriptions
cell	A simulated cellular space containing multiple wireless connected clients.
Cr	The number of CPU cores applied during simulation.
C	The number of simulated cells.
M	The number of wireless connected clients per cell.
N	The number of total clients during simulation.
p_n	The percentage of the partitioned dataset for client n.
r	The ratio of activated clients during simulation.
$RBER$	The received bytes error rate.
E	The number of local SGD updating steps.
bs	The local training batch-size.
NIS_a	The addictive noise caused by malicious clients.
NIS_m	The multiplicative noise caused by malicious clients.
η_t	The local learning rate at communication round t.

we propose the system design in Section III and present the details of wireless environmental setups and convergence analysis in Section IV. The effectiveness and efficiency of the proposed system design and its associated simulator are verified through several applications in Section V. Finally, the paper is concluded in Section VI. For readers' convenience, we list key notations and the corresponding descriptions used in this paper in Table 1.

II. RELATED WORKS

Before planning the generic simulator design of AirDAI, we need to have a profound insight into the research trends and demands of DAI in wireless communications in recent years. To capture the research trends and demands well, we carry out a comprehensive literature review of most key research works and milestones in this section.

It has been recognized in [15], [28] that communications are the critical bottleneck for DAI because of the heterogeneity of wireless networks. Therefore, communication-efficient protocols are imperative for sending messages of model updates as part of the training process, which should stipulate the number of communication rounds and the size of transmitted messages at each round [29]–[31]. Another core challenge mentioned in [15] is that massive clients' unreliable connections must be considered when modeling and analyzing DAI in wireless communication networks. Most importantly, the statistical heterogeneity of clients must be considered, which indicates that the signal propagation environments and system configurations of clients are diverse. As a result, personalized and client-specific modeling for DAI in wireless communication networks is required.

An essential application of DAI in wireless communication networks is related to mobile edge computing [7]. In [4], DAI in mobile edge networks is comprehensively reviewed, and a DAI-aided edge computing system is constructed. This work also summarizes three unique characteristics of DAI-aided edge computing networks: Slow and unstable communications, heterogeneous clients, and privacy/security concerns. The resource allocation problems for DAI-aided edge computing networks are briefly discussed, including client selection, adaptive aggregation, and incentive mechanisms. It has also been pointed out in [4] that DAI-aided edge computing can help with several wireless applications, e.g., base station (BS) association and vehicular communications.

In a broader context, the motivation, opportunities, and challenges of leveraging DAI for wireless communications are discussed in [20]. The optimization of learning time versus energy consumption by using the Pareto efficiency model and the equilibrium between computation and communication for DAI in wireless communication networks are presented in [22], in which qualitative insights into DAI in wireless communication networks and a simplified multi-access communication model are provided. The model quantifies the transmission time and energy consumption for a given amount of data in DAI-aided wireless communication networks. The following study on the resource allocation problems, including transmission time, energy consumption, and DAI convergence, is presented in [21]. However, they optimize the total energy or transmission time consumption of all users while constrained by a simplified communication and computation model. A more realistic communication model of DAI for wireless communication networks is constructed in [23], in which learning, wireless resource allocation, and client selection are jointly optimized to minimize the DAI loss function under the constraints of latency and energy consumption. The same model is also utilized in [32] to reduce the convergence time for DAI over wireless communication networks.

DAI has also been utilized in more complicated wireless application scenarios, e.g., the Internet of Things (IoT), wireless sensor networks, and vehicular communication networks. In [33], DAI is applied to power-constrained IoT devices with slow and sporadic connections, and a fully decentralized DAI system without the DAI server is proposed. The decentralized DAI system relies on device-to-device (D2D) communication protocols and is particularly suited for dense networks consisting of massive cooperative devices. In [34], an incentive mechanism is proposed and studied to encourage clients to contribute to DAI in the IoT. The participation of massive clients in the DAI system is formulated as a Stackelberg game, and the Nash equilibrium of the game is derived. DAI is also employed to estimate the tail distribution of vehicle's queue lengths in vehicular communication networks, which has been verified to produce comparable accuracy to centralized learning methods [35].

III. SYSTEM PROPOSAL

We propose the AirDAI system in this section. We analyze and decompose the essential elements of DAI in general, introduce the programming procedures for its associated simulator, and expound on its scalability. The AirDAI process can be expressed directly as follows: *Iteratively, a server aggregates messages from clients and broadcasts updates back, while clients train the local models with the received message on local datasets.* To make a global view of the holistic process and visualize it, we abstractly decompose

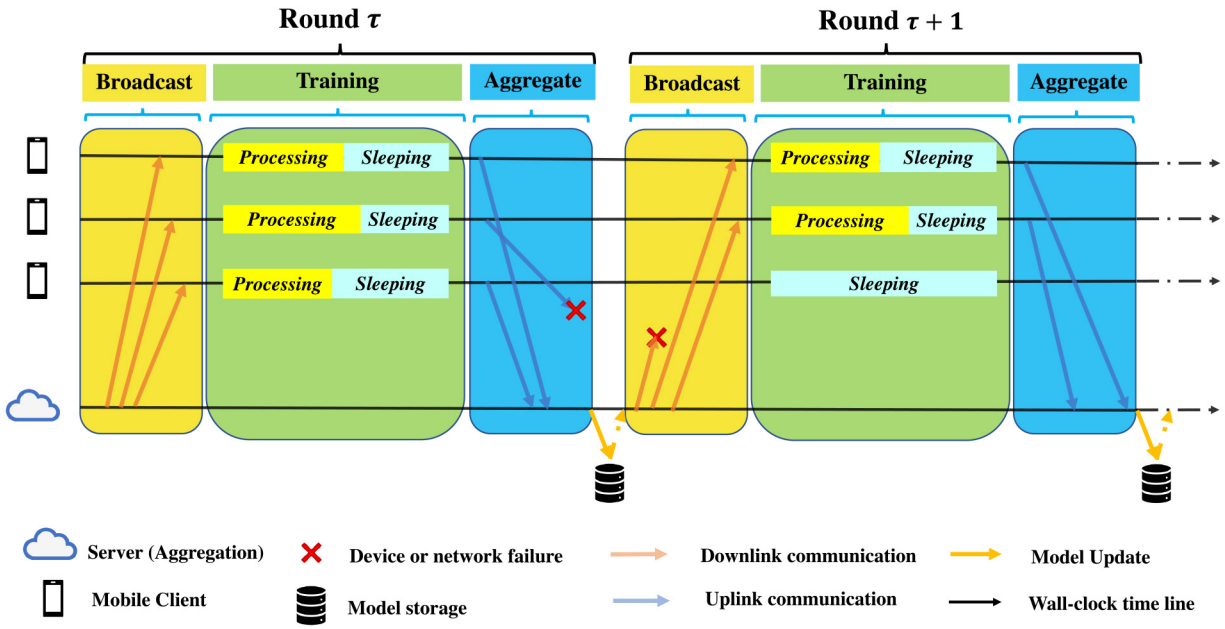


FIGURE 1. Virtualization of two successive operational rounds of AirDAI.

the process into two observation aspects: temporal and spatial. From the temporal perspective, the process comprises the computing module (local training) and the communication module (broadcast and aggregation) due to the mutually exclusive time slots, in which the two modules are executed iteratively. From the spatial perspective, an AirDAI task mainly comprises the computing clients and the central server, whose topology varies according to the instant characteristics at each communication round due to dynamical wireless environments. For the convenience of illustration, we refer to both clients and servers as agents in the following without ambiguity.

Specifically, at the beginning of each time slot, clients process the pre-defined training tasks based on the local datasets and send the computed results to the server for aggregation. Once received messages from the clients, the server further processes messages by a pre-defined aggregation function. Then, based on specific broadcasting strategies, the server sends the processed data back in a limited time window or after completing the reception phase from all clients. The interaction, which begins with the server broadcasting and ends when the server aggregates the result, is defined as a *round*, as illustrated in Fig. 1.

With the above explanations and settings, we represent the τth round abstractly as follows:

$$\begin{cases} \texttt{Server}: \mathcal{K}_\tau^n \longleftarrow \texttt{broadcast}\{\texttt{aggregate}\{\mathcal{J}_\tau^n\}\} \\ \texttt{Clients}: \mathcal{J}_{\tau+1}^n \longleftarrow \Phi_{\mathcal{D}_n}(\mathcal{J}_\tau^n, \mathcal{K}_\tau^n) \end{cases}, \tag{1}$$

where we utilize $\mathcal{J}_\tau^n$ to denote messages sent out from the client n at round τ and $\mathcal{K}_\tau^n$ to indicate messages sent back from the server to the client n at round τ. After one complete iteration, the system begins the $(\tau+1)$th round and the client k processes its pre-defined task Φ based on its own dataset $\mathcal{D}_n$ with received messages $\mathcal{K}_\tau^n$ at round τ. After finishing the computation phase, it sends the computed result $\mathcal{J}_{\tau+1}^n$ to the server for aggregation. It is worth noting that the 'aggregation' and 'broadcast' may only affect a subset of clients according to specific policies. The above (1) is a generic virtualized process that covers the most well-known DAI paradigms of FL, local SGD, and DDL [22], [24].

1) Synchronous and Asynchronous Settings: Considering whether clients receive the same messages from the server during each *round*, DAI schemes can be classified into synchronous and asynchronous categories [10], [36]. With the asynchronous settings, the server receives the data from a single client, then aggregates it with the historical data from other clients, and sends it back to the corresponding client before aggregating the data from newly coming clients. The server has to suspend broadcasting before aggregating data from all clients or the activated clients within a pre-defined time window with the synchronous settings. The broadcast results after aggregation are identical to the activated clients during each *round*. These schemes can be achieved by adjusting the virtual functions of `broadcast{·}` and `aggregate{·}` at the server end, making both the synchronous and asynchronous schemes compatible within the format of virtualization (1).

2) Network Topology and Virtual Channels: To enable topological formulations, we can treat the agents, including clients and servers, as vertices and the communication channels as edges. The network topology can be built as a bi-directional graph. Intuitively, we can represent the system as a graph $\mathcal{G} = (\mathcal{N}, \Theta)$, where $\mathcal{N}$ denotes the set of the

clients and server, and Θ denotes the set of effective virtual channels. The system can be flexibly configured with varied wireless environmental settings by assigning specific parameters to corresponding vertices and edges, such as communication and computation power to different agents or WiFi/LTE settings.

3) QoS and Termination Conditions: While not only paying attention to the validation accuracy or loss similar to conventional DL tasks, the proposed AirDAI system focuses on the output of system QoS, e.g., total energy/time consumed, the number of activated clients per round, the number of packets lost, etc. Meanwhile, the server monitors the simulator states for each round and stops the simulation if one or more user-defined termination conditions are satisfied, e.g., validation accuracy reaches 98%; simulation time is more than 30 minutes; total energy consumed is more significant than 300 J, etc.

A. AIRDAI PROGRAMMING PROCEDURES

According to the proposed system, a typical AirDAI task can be generalized into three steps:

- Building the network topology with virtual channels;
- Defining the aggregating and broadcasting functions;
- Partitioning the training dataset and building the DL model.

We give introductions to all these steps as follows.

1) BUILDING NETWORK TOPOLOGY

We provide a Python-written interface to automatically build the network topology with a specified data structure as input. The input is organized by agents with varied attributes. Each agent is represented by a tree-like data structure with its identity denoting the tree root. We arrange different layers for each agent data structure to place the attributes according to the corresponding characteristics. For instance, we manually set the attribute "role" in the first layer of each tree with different string values to distinguish between the clients and the server. Generally, we arrange the attributes related to the agent itself in the first layer, such as the battery capacity, the initial location and mobility speed, the computation and communication power, etc. We cannot omit the attributes between a pair of adjacent nodes considering asymmetric channels between nodes. For those attributes shared among multiple nodes, such as the virtual channels between pairs of adjacent nodes, we set the attribute "adj" in the first layer and the adjacent node identities in the second layer with the related attributes in the third layer. Therefore, this definition of data structure is also memory efficient. The embedded interface will parse the data structure and complete the topology automatically. The underlying codes for simulating wireless networks are achieved within the system of ns-3 [37] to take advantage of the existing functions of network simulators. We present an example of the bi-directional network topology in Fig. 2 where one server, four APs, and several mobile clients are communicated through optical fiber, wireless broadcasting, and WiFi. The corresponding data structures of server 0, AP 1, and Client 1-1 are demonstrated in Fig. 3.

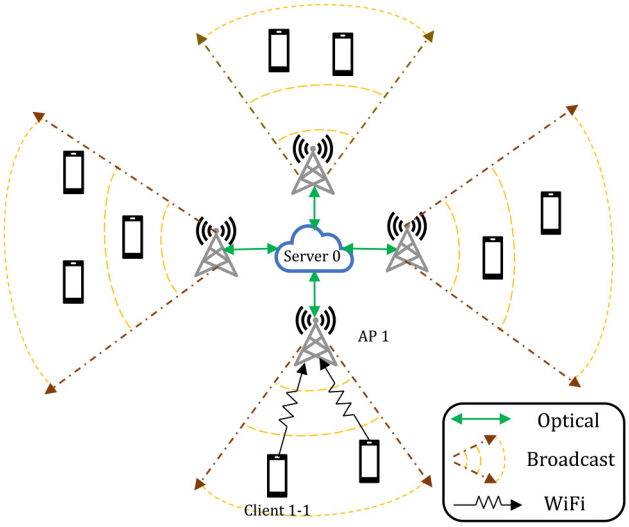


FIGURE 2. Example of bi-directional network topology with one server, four APs and several clients communicated through optical fiber, wireless broadcasting and WiFi.

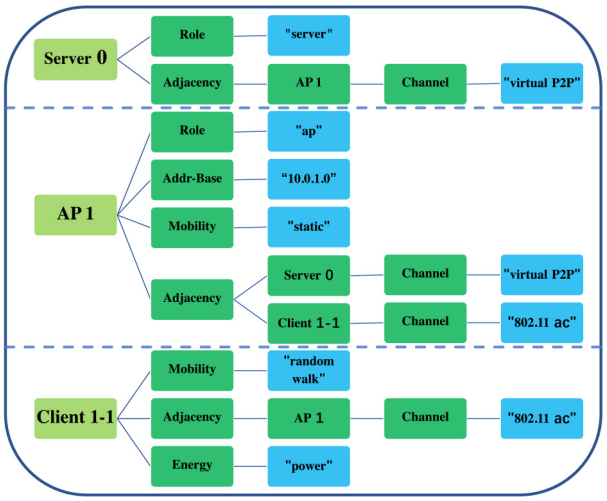


FIGURE 3. Data structures of server 0, AP1 and Client 1-1 for demonstration purposes.

2) DEFINING AGGREGATION AND BROADCASTING FUNCTIONS

The system provides a programming paradigm to define personalized aggregating and broadcasting functions. It keeps a *buffer* placeholder for each agent to receive or send new data from/to other agents and a *memory* placeholder to memorize the *buffer* during each round. As a result, aggregating and broadcasting functions may only work within the activated agents in predefined network topology during each round to emulate the failure of transmissions in realistic wireless environments due to certain QoS constraints or powered off.

Once an agent receives new data sequentially from the others, the *buffer* will record the data and update its value according to the personalized update function. In addition, the *memory* keeps tracking the latest *buffer* value. Mathematically, the process can be formulated as follows:

$$\begin{cases} \texttt{buffer} \longleftarrow \texttt{Update}(\texttt{buffer}, \texttt{memory}) \\ \texttt{memory} \longleftarrow \texttt{buffer} \end{cases}, \quad (2)$$

where the function $\texttt{Update}(\cdot, \cdot)$ represents the user-defined buffer updating scheme. Taking FedAvg[10] as an example, $\texttt{Update}(\cdot, \cdot)$ is the weighted average function of the latest received data and its memory. The *buffer* for clients is the returned data at the end of each *round*. Then, the *buffer* updates itself with the latest received data and the memorized data from previous rounds.

Meanwhile, the synchronous and asynchronous settings can also be achieved by determining when the server sends the updated *buffer* to its adjacent client nodes. Specifically, when adopting asynchronous settings, the server immediately returns the updated *buffer* to its recently communicating client. In contrast, with synchronous settings, the server broadcasts the recently updated buffer only after receiving data from a required number of clients.

3) PARTITIONING DATASET AND BUILDING DL MODELS

The DAI tasks presume that the training dataset must be partitioned into multiple computing clients before training. We provide a paradigm to define the strategy of dataset partition. Each simulation process loads the identical raw dataset from shared memory and splits it according to the predefined partition ratio of each client. Subsequently, each client with a unique rank will be assigned the corresponding sub-dataset. If the partition ratio is not specified, the dataset will be, by default, partitioned into all clients in a uniform and random manner. After dataset partitioning, the definition of the DAI model is just the same as the centralized counterparts. The proposed system provides a Python wrapper function for the model to automatically aggregate and broadcast required values during the training process at each round while keeping users unaware of it unless users would like to customize the aggregate and broadcast functions. Users can perform the same for the other training settings as if there were only one client in centralized tasks.

B. SCALABILITY

The DAI tasks in natural environments usually involve many computing devices with limited computing power and storage space, such as intelligent IoT devices and wireless sensors. We implement two ingenious methods to emulate this characteristic of limited available computing resources, e.g., a powerful workstation with several computing cores or a small computing cluster.

First, the proposed system can run on multiple computing cores through distributed multi-processing interface (MPI) communication backends [26]. Before initiating simulations, the system automatically partitions the clients and the server into different computing cores and gives each core a unique rank identity. Each core maintains the identical wireless topology, in which the clients and server partition details are recorded. To distinguish multiple clients simulated in parallel but on different computing cores, we assign a unique address to each client as (*rank_id*, *node_id*), where *node_id* is the index of the agent in its corresponding core. During aggregating and broadcasting in each round, the clients send and receive data to/from the corresponding computing core where the server is located through communication backends. Also, the whole communication process is unaware to users.

Second, within each computing core, we propose and utilize the scheme called "series-tube", which provides a wrapper function and serially executes a list of objects defined in Python to enhance the capability of the simulator. The wrapper function replicates the original "objects" into a list according to the number of clients in a single computing core while maintaining its functions and values as a series-tube object. By calling the wrapped object, the simulator serially processes the functions of the replicated objects and returns the results into a list format. Therefore, it keeps the whole process user-unaware and makes the codes scalable with just a few modifications.

IV. WIRELESS ENVIRONMENTAL SETUPS AND CONVERGENCE ANALYSIS

As we introduced the system in the last section, the successful implementation of DAI in realistic transmission environments depends on the reliability of the wireless channels over which model parameters are transmitted. It is undoubtedly that training a model in an unreliable wireless environment will degrade the efficiency compared to that in a fully reliable environment. Therefore, it is worth investigating and quantifying the impacts of the randomness of wireless channels on the training procedure of DAI. As the simulator's core, we try to keep our design as generic as possible and expound on the wireless system setups. Then, based on the given wireless environmental configurations, we further analyze the convergence of a generic DAI algorithm.

A. EFFECTS OF WIRELESS ENVIRONMENTAL SETUPS

There are two kinds of wireless channels pertaining to the uplink and downlink. The former refers to the transmission links from the clients to the DAI server, while the latter refers to the links from the DAI server to the clients. Because the global model parameters transmitted from the DAI server are the same for all clients, we can easily adopt a broadcast protocol for the downlink transmission with sufficiently large transmit power and bandwidth. Therefore, its reliability can be guaranteed. On the contrary, a unicast protocol is adopted for uplink transmissions because all clients are required to send unique local model parameters. However, because clients generally have less transmission capability, uplink transmission reliability is problematic, and uplink communication efficiency is of paramount importance [10].

Furthermore, the unstable uplink transmission will result in a reduced number of clients' responses within a time window[2] ϵ, which could lead to inefficient aggregation at the DAI server and thereby a low training efficiency overall. Consequently, the wireless communication models of the uplink require special attention and are worth investigating. In the following, we analyze how the randomness of wireless uplink channels affects the number of clients' responses within a predetermined time window.

Temporarily neglecting packet transmission errors, whether or not a packet from a certain client can be received is directly related to the random event that whether the transmission latency of the packet from the nth client, denoted as L_n, is less than or equal to time window ϵ, $\forall\ n \in \{1, 2, \ldots, N\}$, where N is the total number of clients. Referring to the Shannon-Hartley theorem, the transmission latency L_n is dominated by four factors: 1) bandwidth B_n; 2) transmit power PT_n; 3) packet size S_n; 4) channel power gain G_n. To be explicit, we can also express the transmission latency as a function of these four factors: $L_n(B_n, PT_n, S_n, G_n)$.

The first three factors mentioned above are specified by communication and DAI computing protocols. They are determinate, while the last factor, i.e., the channel power gain G_n, is stochastic and randomly varies over time, frequency, and space. Statistically, channel power gain G_n is mainly affected by four wireless propagation phenomena: 1) path loss; 2) shadowing; 3) multi-path fading; 4) molecular absorption (applicable to millimeter-wave and terahertz radios). The joint impacts of these wireless propagation phenomena can be described and simulated by different channel models, e.g., Rayleigh, Rician, and Nakagami channel models, as well as a variety of compound channel models [23], [38]–[41], depending on the use of spectrum, node mobility, geographical and atmospheric conditions. To maintain generality, we do not specify the use of the channel model in this paper.

Meanwhile, considering that errors in the received packet might exist, error check and re-transmission are imperative in most modern communication protocols. Incorporating both mechanisms, the total transmission time of a client, denoted as $TL_n = L_n\Sigma_n$, depends on the transmission latency of a single transmission attempt L_n and the number of re-transmissions Σ_n. Note that the number of re-transmissions Σ_n is also a random variable related to the coding and modulation setups and characterized by packet error rate PER_n. For simplicity, we can adopt the geometric distribution with parameter PER_n to model the random number of re-transmissions Σ_n. Based on the formulation and explanation presented above, we can simply define the packet loss rate of the nth client in the physical layer to be $\rho_n = \hat{F}_{TL_n}(\epsilon) = \mathbb{P}\{TL_n > \epsilon\} = 1 - F_{TL_n}(\epsilon)$, where $F_{TL_n}(\epsilon)$ and $\hat{F}_{TL_n}(\epsilon)$ are the cumulative distribution function (CDF) and the complementary CDF (CCDF) of the total transmission time $TL_n(TL_n, \Sigma_n)$ considering packet errors and re-transmissions.

We can now characterize the number of clients' correct responses $\tilde{N}$ within the preset time window ϵ. Assuming only the correct responses received within ϵ will be recorded at the DAI server, the number of recorded correct responses from clients $\tilde{N}$ is a dependent random number on the total transmission time $\{TL_n\}_{n=1}^{N}$. Because the transmissions of all N clients are mutually independent, the randomness of $\tilde{N}$ can be characterized by the probability mass function (PMF) infra:

$$\begin{aligned}\Phi_{\tilde{N}}(\eta) &= \mathbb{P}\left\{\tilde{N} = \eta\right\} \\ &= \sum_{\tilde{\mathcal{N}}(\eta)\subseteq\mathcal{N}} \left(\prod_{n\in\tilde{\mathcal{N}}(\eta)} F_{TL_n}(\epsilon)\right)\left(\prod_{n\in\mathcal{N}\setminus\tilde{\mathcal{N}}(\eta)} \hat{F}_{TL_n}(\epsilon)\right),\end{aligned} \tag{3}$$

where $\mathcal{N}$ is the full set of N clients and $\tilde{\mathcal{N}}(\eta)$ is an arbitrary subset of η clients that transmit correct responses within the given time window ϵ; the summation operation is carried out over all $\binom{N}{\eta}$ subsets of η clients.

Assuming all clients are homogeneous, which implies all their channel distribution parameters and other wireless setups to be identical, we have $\rho = \rho_1 = \rho_2 = \cdots = \rho_N$. As a result, the number of clients' correct responses $\tilde{N}$ within the preset time window ϵ abides the binomial distribution with N dependent trials and success probability $r = 1 - \rho$. Therefore, we can reduce (3) to be $\Phi_{\tilde{N}}(\eta) = \binom{N}{\eta} r^{\eta}(1-r)^{N-\eta}$. When the total number of clients N is large, we can rely on the law of large numbers and have the following relation:

$$\tilde{N} \approx \mathbb{E}\{\tilde{N}\} = Nr. \tag{4}$$

Based on this simplification, although r is defined as the probability that a packet can be correctly received within the time window, it quantitatively equals the ratio of activated clients for large N. We denote both measures by r herein for notational simplicity unless otherwise specified.

B. ANALYSIS OF ALGORITHMIC CONVERGENCE OF DAI

In the previous subsection, we qualitatively analyzed that the time window can influence the ratio of activated agents and thus yields an effect on the algorithmic convergence of DAI. In this subsection, we present the quantitative analysis of the convergence rate concerning the ratio of activated agents. Although the internal processes can be understood from the abstraction given in (1), it can hardly help for analytical formulations and derivations. Hence, for facilitating the following analysis of convergence, we begin with

2. The time window is dynamically managed by pace steering techniques, depending on the number of clients and service requirements [4], [24]. For example, when the number of clients is small, the time window ϵ should be set to a relatively large value so that a sufficient number of responses from clients can be collected and aggregated at the DAI server. On the other hand, when the number of clients goes large, the time window ϵ should be reduced to reduce the computing burden at the DAI server. The time window ϵ is, in essence, a trade-off factor between computing and communication efficiencies.

re-defining the mathematical problem as follows:

$$\min_{\mathbf{w}} \left\{ F(\mathbf{w}) \triangleq \sum_{n=1}^{N} p_n F_n(\mathbf{w}) \right\}, \tag{5}$$

where p_n is the weight of the client n such that $p_n \geq 0$ and $\sum_{n=1}^{N} p_n = 1$. Suppose that the client n holds the s_n training data samples: $x_{n,1}, x_{n,2}, \ldots, x_{n,s_n}$; local objective function $F_n(\cdot)$ is defined as

$$F_n(\mathbf{w}) \triangleq \frac{1}{s_n} \sum_{j=1}^{s_n} \ell\big(\mathbf{w}; x_{n,j}\big), \tag{6}$$

where $\ell(\cdot;\cdot)$ is a user-specified loss function. The problem aims at minimizing the averaged loss value through minimizing the local objective function at each distributed device. Without losing of generality, we make some common assumptions for simplifying the analysis:

- F_n is L-smooth function, $\forall\ n \in \mathcal{N}$;
- F_n is μ-strong convex function, $\forall\ n \in \mathcal{N}$;
- The variance of stochastic gradients in each client is bounded σ^2;
- The expected squared norm of stochastic gradients is uniformly bounded by G^2.

Interested readers can refer to the Appendix for mathematical implications and the inherent rationality of these assumptions.

Taking the well-known FedAvg algorithm proposed in [16] as an example, we describe the process of its τth round by utilizing the abstraction given in (1). Firstly, the server broadcasts the latest model parameters $\mathbf{w}_\tau$, to all clients, and hence, the message $\mathcal{K}_\tau^n$ received at client n is $\mathbf{w}_\tau$ assuming a perfect downlink channel. Secondly, every client takes the received $\mathbf{w}_\tau$ as the update at beginning of the local round, i.e., $\mathbf{w}_t^n = \mathbf{w}_\tau$, and performs $E(\geq 1)$ local SGD updates based on its own dataset:

$$\mathbf{w}_{t+i+1}^n \longleftarrow \mathbf{w}_{t+i}^n - \eta_{t+i} \nabla F_n\big(\mathbf{w}_{t+i}^n, \xi_{t+i}^n\big), \tag{7}$$

for $i = 0, 1, \ldots, E-1$, where η_{t+i} is the learning rate, and ξ_{t+i}^n denotes the samples uniformly chosen from the local dataset at each SGD updating step. Thirdly, after locally updating through E steps, every client sends the latest model parameters to the central server. The message $\mathcal{J}_n^{\tau+1}$ sent out from the client n is represented by $\mathbf{w}_{t+E}^n$. Last, the central server aggregates the local models received from clients $\{\mathcal{J}_1^{\tau+1}, \ldots, \mathcal{J}_N^{\tau+1}\}$ to produce a new global model $\mathbf{w}_{\tau+1}$ for the next round.

Because of the non-iid data distribution and partial-client participation issue when applying DAI in realistic wireless environments, the aggregation steps can be various. Ideally, if the server receives messages from all clients (a.k.a. full-client participation) before broadcasting, the aggregation could be

$$\mathbf{w}_{\tau+1} \longleftarrow \sum_{n=1}^{N} p_n \mathbf{w}_{t+E}^n. \tag{8}$$

Otherwise, the partial-client participation issue rises, which can lead to low training efficiency without taking proper countermeasures. Specifically, the server receives the first K $(1 \leq K \leq N)$ messages and stops to wait for the rest. Let $\mathcal{S}_\tau (|\mathcal{S}_\tau| = K)$ be the set of the indices of the responded clients in the τth round. Then, the aggregation with partial clients' responses is performed according to

$$\mathbf{w}_{\tau+1} \longleftarrow \frac{N}{K} \sum_{n \in \mathcal{S}_\tau} p_n \mathbf{w}_{t+E}^n. \tag{9}$$

Comparing (9) with (7), it is evident that the partial-client participation issue slows down the algorithmic convergence of DAI by reducing the number of aggregated samples. The convergence rate of the FedAvg algorithm has been well studied when the required number of clients is constant in [42]–[44]. Therefore, we focus on the convergence when the number of required clients is changeable among communication rounds, which reflects the realistic scenario in wireless environments, especially when we set a small time window. Our analysis is based on the recent research of federated learning on Non-IID data [44].

Assume that the server receives $\tilde{N}_t$ (say the t-th communication round) activated clients within the preset time window, and assume that the total number of communication rounds is T. Let $\Delta_t \triangleq \mathbb{E}\|\overline{\mathbf{w}}_t - \mathbf{w}^\star\|^2$, defined as the expected distance to the optimum, where $\overline{\mathbf{w}}_t = \sum_{k=1}^{N} p_k \mathbf{w}_t^k$ is the weighted average of model parameters among all clients, and $\mathbf{w}^\star$ denotes the optimized model parameters.

Lemma 1: Assume that the central server received $\tilde{N}_t$ activated clients in the preset time window. Define $\Gamma = F^* - \sum_{k=1}^{N} p_k F_k^*$ to quantify the degree of heterogeneity of non-iid distributions. Letting $\Delta_t = \mathbb{E}\|\overline{\mathbf{w}}_{t+1} - \mathbf{w}^\star\|^2$, we have

$$\Delta_{t+1} \leq (1 - \eta_t \mu)\Delta_t + \eta_t^2 (B + C_t), \tag{10}$$

where $B = \sum_{k=1}^{N} p_k^2 \sigma_k^2 + 6L\Gamma + 8(E-1)^2 G^2$, and $C_t = \frac{N - \tilde{N}_t}{N-1} \frac{4}{\tilde{N}} E^2 G^2$.

Proof: Please refer the Appendix for details. ■

Apparently, $C_t = 0$ if and only if $\tilde{N}_t = N$. Because of this inequality, we are unable to obtain the optimal solution directly. Alternatively, we can find the bound on the solution by analyzing its supremum. We use $\sup(\Delta_t)$ to denote the supremum of Δ_t for $t = 1, 2, \ldots, T$, given η_{t-1} being the learning rate at the $(t-1)$th step. Besides, we let $\sup\sup(\Delta_t)$ denote the supremum of Δ_t for $t = 2, 3, \ldots, T$, given Δ_{t-1} reaching its supremum $\sup(\Delta_{t-1})$ at the $(t-1)$th step with η_{t-2} being the learning rate at $(t-2)$th step. With these denotations, it follows that

$$\begin{cases} \sup(\Delta_{t+1}) = \min\limits_{\eta_t} \big[(1 - \eta_t \mu)\Delta_t + \eta_t^2 (B + C_t)\big] \\ \sup\sup(\Delta_{t+1}) = \min\limits_{\eta_t} \big[(1 - \eta_t \mu)\sup(\Delta_t) + \eta_t^2 (B + C_t)\big], \end{cases} \tag{11}$$

$\forall\ t = 1, 2, \ldots, T-1$, by which we can determine the minimum by

$$\begin{cases} \sup(\Delta_{t+1}) = \Delta_t - \frac{\mu^2 \Delta_t^2}{4(B+C_t)} \\ \sup\sup(\Delta_{t+1}) = \sup(\Delta_t) - \frac{\mu^2 \sup(\Delta_t)^2}{4(B+C_t)}. \end{cases} \tag{12}$$

For the quadratic function $f(x) = x - \frac{\mu^2 x^2}{4(B+C)}$, we can obtain its maximum to be $\frac{B+C}{\mu^2}$ when $x = \frac{2(B+C)}{\mu^2}$ and derive $f(x_1) \leq f(x_2)$ when $x_1 \leq x_2 \leq \frac{2(B+C)}{\mu^2}$. As a result, letting $x = \Delta_{t-1}$, we know that $\sup(\Delta_t) \leq \frac{B+C_{t-1}}{\mu^2}$. Because of $B > C_t$, $\forall\ t = 1, 2, \ldots, T$, we can derive the inequality $\frac{B+C_{t-1}}{\mu^2} \leq \frac{2(B+C_t)}{\mu^2}$. Finally, we obtain $\sup(\Delta_{t+1}) \leq \sup\sup(\Delta_{t+1})$.

Recursively let

$$\tilde{\Delta}_{t+1} = \min_{\eta_t} \Big[(1 - \eta_t \mu)\tilde{\Delta}_t + \eta_t^2 (B + C_t) \Big], \tag{13}$$

for $t = 0, 1, \ldots, T-1$, and let $\tilde{\Delta}_0 = \Delta_0$. Given $t' < t$, it can be found that $\tilde{\Delta}_t$ is the supremum of Δ_t by setting all its previous $\Delta_{t'}$ being the corresponding supremum. With the analysis above, we know that the supremum converges fastest when $\eta_t = \frac{\mu \tilde{\Delta}_t}{2(B+C_t)}$. With the above analysis, we want to find the relations between the learning rates of partial device participation and full device participation conditions. The result is presented as follows.

Lemma 2: Denote $\bar{\eta}_t$ to be the learning rate at communication round t to guarantee the algorithm convergence when full devices are participated. Let $r_t = \frac{\tilde{N}_t}{N}$ be the device participation ratio at communication round t. The convergence of the algorithm when partial devices are participated can be guaranteed by setting $\eta_t = r_t \bar{\eta}_t$.

Proof: Hint: By analyzing the relation of learning rates between $C_t = 0$ and $C_t > 0$, we can find an equation to combine the two conditions. Please refer the Appendix for details. ■

Withe the analysis of Lemma 1 and 2, we can begin to analyze the convergence rate in the wireless environments as follows.

Theorem 1: Let the assumptions hold and L, μ, σ_k, G be defined therein. Choose $\kappa = \frac{L}{\mu}$, $\gamma = \max\{8\kappa, E\}$ and the learning rate $\eta_t = \frac{2r_t}{\mu(\gamma + t)}$. Then FedAvg algorithm in wireless environments satisfies

$$\mathbb{E}[F(\mathbf{w}_T)] - F^* \leq \frac{2\kappa}{\gamma + T} \left(\frac{B + D}{\mu} + 2L \left\| \mathbf{w}_0 - \mathbf{w}^* \right\|^2 \right),$$

where $B = \sum_{k=1}^{N} p_k^2 \sigma_k^2 + 6L\Gamma + 8(E-1)^2 G^2$, and $D = 4E^2 G^2$.

Proof: Hint: Assume $C_t = 0$ and from Lemma 1, find the bound of Δ_t by induction. Apply the assumptions on F, find the relations between $F(\mathbf{w}_t)$ and Δ_t. Combining with Lemma 2 to find the learning rate in wireless environments. Please refer the Appendix for details. ■

V. EXPERIMENTS

In this section, we take the well-known FedAvg algorithm as an example to validate the effectiveness of the proposed system. In particular, we systematically evaluate the performance of FedAvg with different parameter settings, which can be roughly split into model-related hyper-parameters and system-related parameters. The target of a series of experiments is to study the accuracy, efficiency, robustness, and fairness of a given algorithm based on our proposed system. Besides, we also validate the scalability of the system.

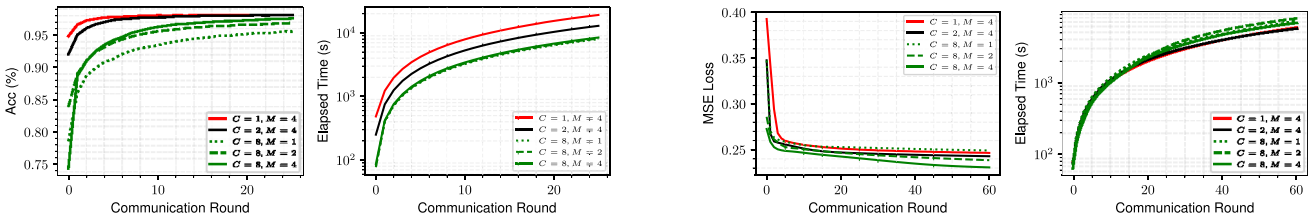
A. EXPERIMENT SETUPS

To demonstrate the generality of our proposed system, we consider two completely different tasks on the PyTorch platform. The first is a multi-class image classification problem for digital recognition, and the second is a regression problem for wireless traffic prediction [45]–[47]. We perform the first task on the MNIST dataset [48]. This dataset is one of the most classical ones in the ML/DL realm and has been widely applied in the literature. We attempt to predict which class the input image belongs to for the multi-class classification problem, and the prediction accuracy is adopted as the evaluation metric. In the experiment, the model architecture adopted for this task is described as follows: A CNN with two 5×5 convolution layers (the first layer with 10 channels and the second layer with 20 channels; each followed by a 2×2 max pooling and the rectified linear unit (ReLU) activation function), a fully connected layer with 50 units utilizing the ReLU activation function for neural computing, and a final softmax output layer [16]. The total number of the applied model parameters equals 1199882. The initial learning rate is set to unity, with an exponential decay rate at 0.9 every 5 local training steps.

We perform the second task on the Call Detail Record (CDR) dataset from 'Telecom Italy Open Big Data Challenge' [49]. The CDR dataset contains three kinds of wireless traffics from different cells: The number of text messages, the number of calls, and the number of Internet data packages. For this problem, we attempt to predict the future traffic volume of a cell, given the historical traffic volumes, and the mean square error (MSE) is adopted as the evaluation metric. In the experiment, the model architecture adopted includes a stacked long short-term memory (LSTM) structure with two LSTM layers (each layer with 64 hidden units) and a fully connected layer with a single output. The total number of the applied model parameters equals 12961. The initial learning rate is set to be[3] 0.05, with an exponential decay rate at 0.9 every 5 local epochs.

We assume that all clients connect to the APs through wireless links. In the following experiments, if without further annotations, we assume all computing clients are located randomly in several wireless cells. Each cell is simulated within one CPU core process, while the server is simulated in another independent core process. Within each cell,

3. Note that the models and (hyper-)parameters we adopted here are relatively straightforward since the design and optimization of network architecture and (hyper-)parameters are out of the scope of this paper.



(a) Image classification task.

(b) Traffic prediction task.

FIGURE 4. Accuracy and time of two real-world ML tasks versus epochs, given different number of WiFi cells and different number of activated clients in each cell.

TABLE 2. Suggested values and explanations of independent channel distribution and system configuration parameters.

Parameters	Suggested values and explanations
B_n	Bandwidth of wireless communications; 10-29 Mbps for 802.11g, 150 Mbps for 802.11n, 3-32 Mbps for 802.11a, 210 Mbps - 1 Gbps for 802.11ac; We choose 10 Mbps here.
PT_n	Wireless transmission power; 80-720 mW for WiFi modulem; We choose 720 mW here.
S_n	Communication packet size; usually less than 64K bits for UDP and TCP protocols; We choose 1K bits here.
G_n	The factor of logarithmic distance propagation loss models; 1.4125-2.2387 for LoS links, 2.1878-3.0549 for non-LoS links; We choose 2 here.
M	Number clients per cell; 1-6 clients per cell; We choose 4 clients per cell here.
data rate	Data rate of P2P channel between the server and Ap nodes; We choose 500 Mbps here.
delay	Delay of P2P channel between the server and Ap nodes; We choose 20 ms here.

we assume that a limited number of computing clients randomly walk in a squared area and communicate with the server through an AP node. We further consider the wireless channel model for each client to be the constant speed propagation delay model and logarithmic distance propagation loss model. We assume that each AP node connects to the server without losing generality through a virtual point-to-point link with a limited data rate and delay. For the sake of simplicity, we assume that all clients in every cell have the same system configurations and adopt the suggested channel and system parameters in [50], which are listed in Table 2.

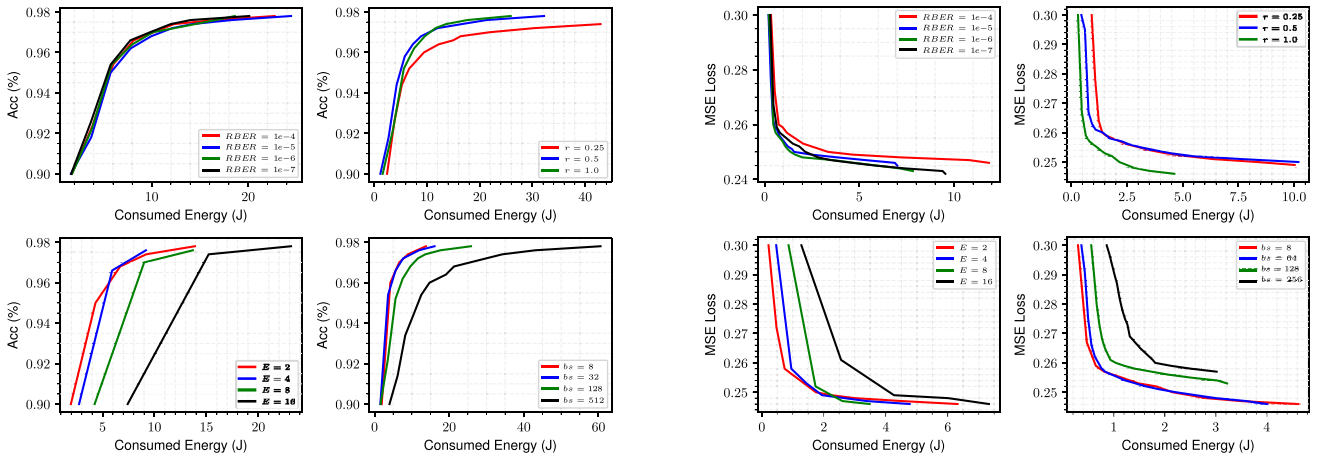
The computing time is closely related to the CPU frequency, IO throughputs, memory cache, and the existing tasks running on the agent's device and thereby hard to formulate mathematically. To precisely simulate the computing time of agents, we assume it to be ten times the computing time on our computational platform, which is a workstation with two physical CPUs, 20 core processes per CPU, and 256 GB memory cache. To avoid interference from the existing tasks running on the workstation, we simultaneously build the simulations for each experiment to keep the same operational conditions. These system configurations are fixed unless otherwise specified.

B. ACCURACY

In this subsection, we present the overall prediction performance of our simulator. The experiments are conducted as follows. We set the number of cells (C) to 1, 2, and 8, respectively. Each cell is simulated in a single CPU core. The number of activated agents (M) in each cell sets to be four by default. Besides, we consider the number of active agents per cell to be 1 and 2 when the number is 8. Thus, we have five scenarios in total. We assume that each agent has a sub-dataset with the same size and distribution for each scenario. Furthermore, we assume that the image classification task in different experiments has the same size as the whole dataset. However, we assume that the sub-dataset size for the traffic prediction task is constant, implying that the full dataset size increases with the number of clients. We also stipulate different learning rates according to Lemma 2 for different activated ratio scenarios. Specifically, we set the learning rate ratio as the sub-dataset size dividing the whole dataset size for each client.

We utilize the accuracy and MSE loss on an independent test dataset to represent the performance of both image classification and wireless traffic prediction tasks. We utilize different colors of red, black, and green in the figures to denote the cases corresponding to the number of cells of 1, 2, and 8, respectively. Besides, we utilize dot-line, dash-line, and solid-line to represent the cases with the number of active agents per cell of 1, 2, and 4, respectively. As the results presented in both sub-figures of Fig. 4, we draw two sets of lines to represent the performance and time versus the number of training rounds.

From Fig. 4, it is clear that the scenario with one cell and four active agents achieves the best performance among all settings in the image classification task. In contrast, the experiment with eight cells and four active agents per cell outperforms the others in the wireless traffic prediction task. However, for the scenarios with the same number of cells in both tasks, the more active agents per cell will lead to better performance. These phenomenons can be applied to explain both the weakness and strength of the FedAvg algorithm. When the number of active clients per cell equals four, no matter how many cells are utilized for training, the whole training dataset keeps unchanged for the image classification task. The mathematical theory has proved the convergence of



(a) Image classification task.

(b) Wireless traffic prediction task.

FIGURE 5. Test dataset performance of two real-world ML tasks versus energy with four variables: Error rate, active ratio, local epochs, and batch size.

FedAvg. However, it does not perform as well as a centralized algorithm in practice. At least, it converges slower than a centralized algorithm. Nevertheless, the conclusion is the opposite in the wireless traffic prediction task. The dataset is not static, and the more cells utilized in the training phase, the larger the training dataset. A more extensive training dataset generally yields better prediction performance. The FedAvg algorithm, as a result of this, works better with a large number of cells, reflecting the negative influence caused by increasing the number of cells.

As for the consumed time, the two tasks perform differently as usual. For the image classification task, the scenarios with a small number of cells spend a significant amount of time to finish the same number of rounds. In contrast, the conclusion is the opposite for the traffic prediction task. The reason is that the computing phase is dominant compared to the communication phase in the image classification task. Therefore, the scenarios with a small number of cells spend more time on computation than those with more cells. The computational time consumed in the traffic prediction task is almost the same for all cells, as they have the same sub-dataset size for training. Therefore, the scenarios with plenty of cells need more time for communication than those with a small number of cells, which causes the opposite results to the image classification task.

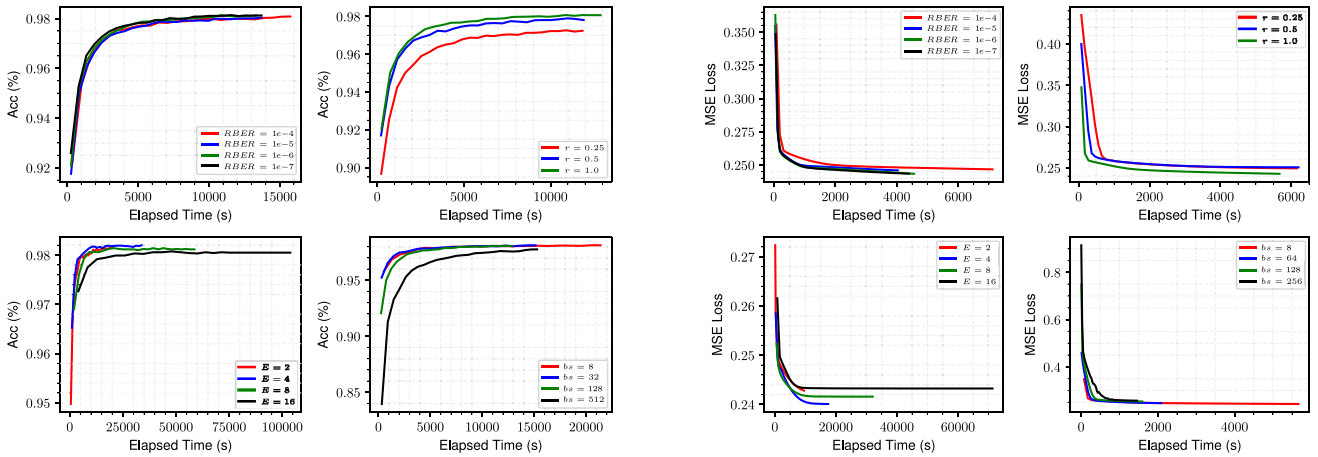
C. EFFICIENCY

In this subsection, we study the factors that affect the efficiency of the FedAvg algorithm in a single cell with four activated agents and other systematic settings in Table 2 by default. We define the efficiency of our system as the energy and time consumed for a task to reach the termination condition. We set the termination condition for our experiments when the FedAvg algorithm reaches an accuracy or a loss threshold. The studied variables include the received bytes error rate (RBER), the agent activated ratio (r), the number of local training epochs (E), and the training batch size (bs). Other variables may also affect the efficiency, but we only study the abovementioned variables due to their dominant and direct impacts.

In particular, we set the accuracy thresholds for the image classification task starting from 0.9 and ending at 0.98 with 0.002 as steps and the loss thresholds for the traffic prediction task starting from 0.3 and ending at 0.245 with -0.001 as steps. We accumulate each threshold's consumed energy and time to draw simulation curves. From previous experience, the simulation curves would present a ladder shape if the energy has not changed between two consecutive thresholds. Therefore, we only keep the first result if the energy value is constant among several successive thresholds. The performance of energy and time with different settings are presented in Fig. 5 and Fig. 6. We discuss the simulation results for two tasks affected by variable settings separately.

The RBERs are chosen from $[10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}]$. Notice that the server averages and aggregates the received bytes, whether correct or corrupted. The figures show that varying RBERs have no significant effects on the test dataset convergence, as they reach the same maximum accuracy or minimum MSE loss. While not the same as the conventional applications, which require the correct received packets or redundant error correction codes (ECC), AirDAI is inherently robust against noise and other channel imperfections, leading to new research on the protocol design of data transmissions. However, it makes sense and can also be observed that a significant RBER will considerably increase the energy and time to reach convergence.

The figures show that a sizeable active ratio performs better than a small one. The reason is that compared to a small active ratio, a larger one has more datasets involved in the training phase, making the test performance reach the same value while consuming less energy. As for the time consumed, the conclusion is not so clear. A more extensive training dataset generally converges faster than a smaller one. However, a large active ratio may increase the time for communications, increasing the total time consumed. Although the figures in our experiments present that a larger



(a) Image classification task.

(b) Wireless traffic prediction task.

FIGURE 6. Test dataset performance of two real-world ML tasks versus time with four variables: Error rate, active ratio, local epochs, and batch size.

active ratio consumes less energy and time to reach the same test performance, we cannot conclude that a sizeable active ratio will always be helpful.

The number of local epochs refers to the number of training epochs for each client during the training phase. A generally accepted common knowledge is that increasing the number of training epochs will significantly decrease the communication over computation ratio and require fewer communication rounds to complete the same number of epochs. A large number of epochs will lead to faster convergence than a small one. However, the results present a counter-intuitive conclusion. There might be two reasons for this phenomenon. First, the computation time takes a significant ratio of a complete round compared to the communication time. Second, it depends on the algorithm. The local training overfits when the number of local epochs reaches a threshold. A further step in increasing the number of epochs will not accelerate the convergence of corresponding tasks.

The simulation results also show that the batch size only affects the training phase. The optimal batch size to reduce energy and time cost for one round depends on the agents' specific tasks and computation power. As shown in the figures, in our experiments, 32 is the best choice for the image classification task among all other options, while 64 is the best for the traffic prediction task.

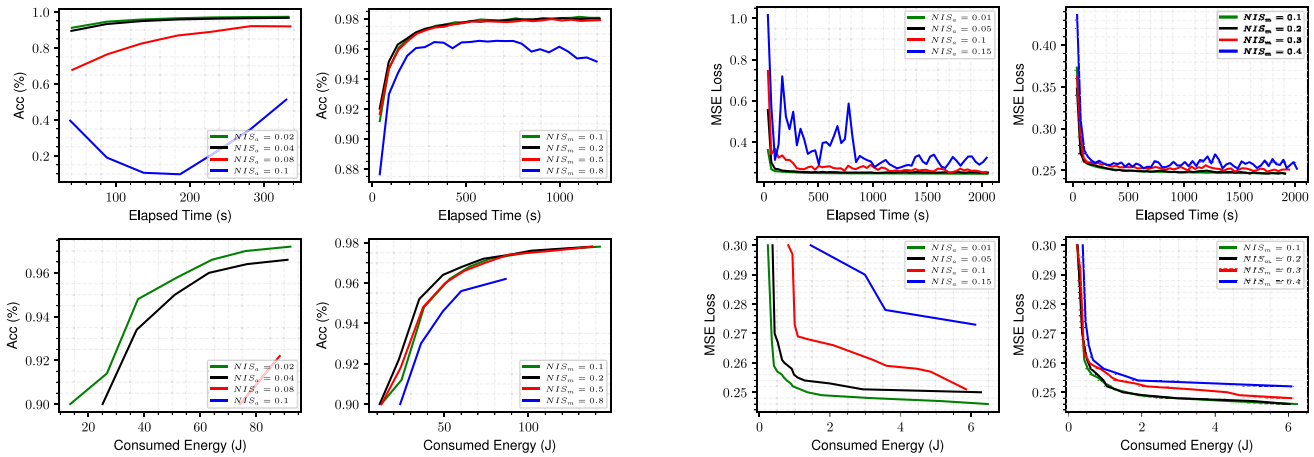
D. ROBUSTNESS

Any practically implementable algorithm must be robust to malicious users in reality [51], [52]. We carry out experiments on the FedAvg algorithm to validate its robustness to malicious agents based on our system. We assume that the agents are malicious and spam erroneous data to the central server. The erroneous data in the following simulations are produced by adding Gaussian noise to the original data. It is worth noting that the added noise strength must be less than a threshold. Otherwise, the central server can easily distinguish the malicious agent by comparing it with the average value and will reject the malicious data. We set up the experiments by considering two kinds of noise: additive and multiplicative. The additive noise is generated as $w_{\text{noise:a}} = w + \mathcal{N}(0, NIS_{\text{a}})$, and the multiplicative noise is generated as $w_{\text{noise:m}} = w \times (1 + \mathcal{N}(0, NIS_{\text{m}}))$, where w is a model parameter capturing the baseline of the correct data, and $\mathcal{N}(0, NIS)$ is a zero-mean and $NIS_{a/m}$-deviation Gaussian distributed random variable.

The simulation results regarding the robustness test are shown in Fig. 7, from which one straightforward observation is that the same noise will affect different tasks differently. For instance, the performance has been significantly degraded for the classification task when NIS_a of the additive noise equals 0.1. In contrast, the traffic prediction still has a competitive performance with the same additive noise. We can observe a similar phenomenon when applying the multiplicative noise. The classification task is more robust to the multiplicative noise than the prediction task. Moreover, the slight value noise has an in-distinctive impact on the accuracy or MSE loss performance. However, it will consume more energy and time than the benchmark without noise to reach the same performance. In conclusion, even applying the same FedAvg algorithm under identical experimental conditions, different tasks with different model parameters will vary from noise levels.

E. FAIRNESS

Fairness is also an important metric and should be evaluated when applying an algorithm in multi-agent environments. Some agents have more raw data than other agents and thus consume more energy during the local training phase. Such a situation could cause service imbalance and reduced training efficiency. We simulate this scenario with different dataset partitions and focus on the system consumed energy, time, training performance, and the consumed energy ratio between two agents when the system reaches the termination condition. The following experiments consider the configurations with one WiFi AP and four agents served by the WiFi AP. The



(a) Image classification task.

(b) Wireless traffic prediction task.

FIGURE 7. Training performance comparisons among different types of noise on the real-world two ML tasks.

TABLE 3. Energy and time consumed with different dataset partitions.

Image Task				
Partition	Energy	Time	Accuracy	Ratio
8 : 1 : 1 : 1	1.030	2.287	0.980	6.190
64 : 1 : 1 : 1	1.059	2.950	0.980	19.462
512 : 1 : 1 : 1	1.058	3.023	0.980	25.548
4096 : 1 : 1 : 1	1.074	3.064	0.980	25.259
512 : 512 : 512 : 1	0.996	1.233	0.980	14.031
4096 : 4096 : 4096 : 1	0.994	1.235	0.980	14.021
Traffic Task				
Partition	Energy	Time	Accuracy	Ratio
8 : 1 : 1 : 1	0.965	2.343	0.248	7.381
64 : 1 : 1 : 1	0.934	2.899	0.251	35.863
512 : 1 : 1 : 1	0.938	3.017	0.252	72.126
4096 : 1 : 1 : 1	0.904	2.914	0.257	77.928
512 : 512 : 512 : 1	1.002	1.257	0.250	53.895
4096 : 4096 : 4096 : 1	0.994	1.248	0.251	58.560

dataset is partitioned according to the partition ratio at the beginning of each experiment. We examine the system outputs when the number of rounds equals 10 for the classification task and 25 for the traffic prediction task.

The partition values given in the header of Table 3 denote the sub-dataset size ratio among four agents. The energy and time denote the total energy and time consumed when reaching the preset termination condition. Also, to significantly compare the energy and time consumed among different partitions, the values are normalized to that of a uniform dataset partition. The values in the ratio column mean the consumed energy proportion of the agents with the largest dataset size and the smallest. From Table 3, we can observe that the energy consumed and evaluation results with different partition scenarios stay unchanged within acceptable errors. Although the impact of unbalanced datasets is insignificant, we can still tell that the unbalanced dataset partitions will affect the training performance for the traffic prediction task. On the other hand, we can observe from Table 3 that the unbalanced dataset partitions significantly affect the total time consumed and energy ratio among different agents for both tasks. Besides comparing the time columns of two sub-tables, the normalized consumed time for the same partition keeps the same within acceptable errors. However, this observation is unsuitable for the values in the ratio columns because the consumed energy ratios between the computation module and communication module are different for these two tasks, affecting each agent's consumed energy while not involving the total consumed time.

F. SCALABILITY

Although the scalability of our proposed system is unrelated to the performance of an algorithm, we still would like to emphasize its importance for users when implemented in practice. We evaluate the scalability against the wall-clock (simulation running) time. The results for both image classification and traffic tasks are presented in Table 4. The cores and cells in each table header denote the numbers of computing cores and simulated cells utilized in each simulation. The simulated cells are uniformly distributed in all computing cores. We present each scenario's average running time per round in the first row. We offer the percent of the wall-clock time of different cores with one core in the second row. Due to the enormous wall-clock consumption of the image classification task, we only conduct the experiments with cells number no greater than 64.

Comparing the results of different cells within the same number of cores makes it straightforward to observe that the wall-clock increases almost linearly with the number of cells. By comparing the results of different cores within the same number of cells, the wall-clock time decreases as expected with the increase in cores number. However, it takes almost the same wall-clock time to simulate one round when the number of cores equals 16 and 32. It is caused by the limitations of the multi-processing scheme and our hardware platform. Compared to the consumed wall-clock time by the simulator for computing purposes, sharing messages among

TABLE 4. Wall-clock time in seconds consumed per round with different numbers of cells and cores.

Image Classification Task						
C \ Cr	1	2	4	8	16	32
32	5862	3329	1595	932	624	623
	100.0%	56.8%	27.2%	15.9%	10.6%	10.6%
64	12712	6073	3318	1763	1282	1404
	100.0%	47.8%	26.1%	13.9%	10.1%	11.0%
Traffic Prediction Task						
C \ Cr	1	2	4	8	16	32
32	54	28	17	10	6	6
	100.0%	52.8%	30.2%	17.0%	11.3%	11.3%
64	110	58	32	19	13	14
	100.0%	52.3%	28.4%	17.4%	11.0%	11.9%
128	225	135	64	39	29	33
	100.0%	59.8%	28.6%	17.0%	12.9%	14.3%
256	496	247	133	88	80	93
	100.0%	49.7%	26.9%	17.8%	16.2%	18.8%

multiple cores takes more time. As a result, increasing the number of computing cores in this situation will not help decrease the wall-clock time.

VI. CONCLUSION

In this paper, we virtualized the basics of DAI in wireless environments and proposed the AirDAI system, which can evaluate the training performance metrics and a set of system-related QoS metrics. In addition, we introduced a general wireless channel model and analyzed the impacts of operating DAI under different wireless setups on the convergence rate. The experimental results revealed how wireless transmission parameters and system configurations affect the training efficiency of the DAI algorithms. Based on the proposed AirDAI system, we designed a Python-built simulator that works on single and multiple computing cores and is compatible with existing ML systems. We took the well-known FedAvg algorithm as an example and conducted extensive experiments with the designed simulator. The experimental results pertaining to prediction accuracy and QoS metrics verified the effectiveness and efficiency of the proposed system and its associated simulator. With this generic system design and the simulator codes, the research progress on DAI in wireless communication systems is expected to be accelerated.

APPENDIX

The Appendix first introduces four general assumptions commonly applied in the SGD convergence analysis. Secondly, we define a new term to distinguish the scenarios of iid and non-iid dataset distributions. Then, we present the lemmas that give the limitation of one-step SGD update and the linear ratio relationship between learning rates. At last, we provide the proof of convergence based on the above two lemmas.

Assumption 1: $F_1, F_2, \ldots, F_N$ are all L -smooth: for all $\mathbf{v}$ and $\mathbf{w}$, leading to $F_k(\mathbf{v}) \leq F_k(\mathbf{w}) + (\mathbf{v}- \mathbf{w})^T \nabla F_k(\mathbf{w}) + \frac{L}{2}\|\mathbf{v}-\mathbf{w}\|_2^2$.

Assumption 2: $F_1, F_2, \ldots, F_N$ are all μ -strongly convex: for all $\mathbf{v}$ and $\mathbf{w}$, leading to $F_k(\mathbf{v}) \geq F_k(\mathbf{w}) + (\mathbf{v}-\mathbf{w})^T \nabla F_k(\mathbf{w}) + \frac{\mu}{2}\|\mathbf{v}-\mathbf{w}\|_2^2$.

Assumption 3: Letting ξ_t^k be randomly sampled from the kth device's local data in a uniform manner, the variance of stochastic gradients in each device is bounded by $\mathbb{E}\|\nabla F_k(\mathbf{w}_t^k, \xi_t^k) - \nabla F_k(\mathbf{w}_t^k)\|^2 \leq \sigma_k^2, \forall\ k = 1, 2, \ldots, N$.

Assumption 4: The expected squared norm of stochastic gradients is uniformly bounded, i.e., $\mathbb{E}\|\nabla F_k(\mathbf{w}_+^k, \xi_+^k)\|^2 \leq G^2, \forall\ k = 1, 2, \ldots, N$ and $\forall\ t = 0, 1, \ldots, T-1$ for $k = 1, \ldots, N$.

The assumptions mentioned above on functions $F_1, F_2, \ldots, F_N$ are general and necessary for the convergence analysis; typical examples include the ℓ_2 -norm regularized linear regression, logistic regression, and softmax classifier.

To extend the analysis on both the iid and non-iid dataset partition scenarios, we propose a new term to quantify the degree of non-iid. The definition is as follows.

Definition 1: Let F^* and F_k^* be the minimum values of F and F_k, respectively. We use the term $\Gamma = F^* - \sum_{k=1}^N p_k F_k^*$ to quantify the degree of heterogeneity of non-iid distributions. That is, if the data are iid, then Γ goes to zero as the number of samples grows. If the data are non-iid, then Γ is nonzero, and its magnitude signifies the heterogeneity of data distributions.

With the above assumptions and definition, we formally present Lemma 1, which limits the expected distance between the current value and the optimum with one-step SGD.

Lemma 3: Assume that the central server received $\tilde{N}_t$ activated clients in the preset time window. Letting $\Delta_t = \mathbb{E}\|\overline{\mathbf{w}}_{t+1} - \mathbf{w}^\star\|^2$, we have

$$\Delta_{t+1} \leq (1 - \eta_t \mu)\Delta_t + \eta_t^2 (B + C_t), \tag{14}$$

where $B = \sum_{k=1}^N p_k^2 \sigma_k^2 + 6L\Gamma + 8(E-1)^2 G^2$ and $C_t = \frac{N-\tilde{N}_t}{N-1}\frac{4}{\tilde{N}} E^2 G^2$.

Proof: The proof of the presented lemma can be found in [44]. ■

We present Lemma 2 as follows, in which we aim at finding the learning rate relations between the full device participation setting and the partial device participation setting caused due to limited time window.

Lemma 4: Denote $\bar{\eta}_t$ to be the learning rate at communication round t to guarantee the algorithm convergence when full devices are participated. Let $r_t = \frac{\tilde{N}_t}{N}$ be the device participation ratio at communication round t. The convergence of the algorithm when partial devices are participated can be guaranteed by setting $\eta_t = r_t \bar{\eta}_t$.

Proof: Let $\bar{\eta}_t = \frac{\mu \tilde{\Delta}_t}{2B}$, which implies that $C_t = 0$ and the number of clients are all activated, we can obtain the following relations:

$$\eta_t = \bar{\eta}_t \frac{B}{B + C_t} = \bar{\eta}_t \left[1 + \varepsilon\left(\frac{N - K_t}{K_t}\right)\right]^{-1}, \tag{15}$$

where $\varepsilon = \frac{C_t}{B} \times \frac{\tilde{N}_t}{N-\tilde{N}_t}$ is a $\tilde{N}_t$-irrelevant constant. For simplicity, ε could be stipulated to be unity, and hence, we obtain the following relation

$$\eta_t = \frac{\tilde{N}_t}{N}\bar{\eta}_t = r_t\bar{\eta}_t, \tag{16}$$

which indicates that we can adapt the learning rate linearly with respect to the number of activated clients. ■

With the lemmas and assumptions mentioned above, we are able to give the bound on the convergence of FedAvg algorithm in wireless environment settings as follows,

Theorem 2: Let the assumptions hold and L, μ, σ_k, G be defined therein. Choose $\kappa = \frac{L}{\mu}$, $\gamma = \max\{8\kappa, E\}$ and the learning rate $\eta_t = \frac{2r_t}{\mu(\gamma+t)}$. Then FedAvg algorithm in wireless environments satisfies

$$\mathbb{E}[F(\mathbf{w}_T)] - F^* \leq \frac{2\kappa}{\gamma + T}\left(\frac{B+D}{\mu} + 2L\left\|\mathbf{w}_0 - \mathbf{w}^*\right\|^2\right),$$

where $B = \sum_{k=1}^{N} p_k^2\sigma_k^2 + 6L\Gamma + 8(E-1)^2G^2$, and $D = 4E^2G^2$.

Proof: Our proof starts with the full device participation condition. Let $C_t = 0$, from Lemma 3 we obtain as follows,

$$\Delta_{t+1} \leq (1-\eta_t\mu)\Delta_t + \eta_t^2 B, \tag{17}$$

For a diminishing step size, $\eta_t = \frac{\beta}{t+\gamma}$ for some $\beta > \frac{1}{\mu}$ and $\gamma > 0$ such that $\eta_1 \leq \min\{\frac{1}{\mu}, \frac{1}{4L}\} = \frac{1}{4L}$ and $\eta_t \leq 2\eta_{t+E}$. We will prove $\Delta_t \leq \frac{v}{\gamma+t}$ where $v = \max\{\frac{\beta^2 B}{\beta\mu - 1}, (\gamma+1)\Delta_1\}$. We prove it by induction. Firstly, the definition of v ensures that it holds for $t = 1$. Assume the conclusion holds for some t, it follows that

$$\begin{aligned}
\Delta_{t+1} &\leq (1-\eta_t\mu)\Delta_t + \eta_t^2 B \\
&= \left(1 - \frac{\beta\mu}{t+\gamma}\right)\frac{v}{t+\gamma} + \frac{\beta^2 B}{(t+\gamma)^2} \\
&= \frac{t+\gamma-1}{(t+\gamma)^2}v + \left[\frac{\beta^2 B}{(t+\gamma)^2} - \frac{\beta\mu - 1}{(t+\gamma)^2}v\right] \\
&\leq \frac{v}{t+\gamma+1}.
\end{aligned} \tag{18}$$

Then by the strong convexity of $F(\cdot)$

$$\mathbb{E}[F(\overline{\mathbf{w}}_t)] - F^* \leq \frac{L}{2}\Delta_t \leq \frac{L}{2}\frac{v}{\gamma+t}. \tag{19}$$

Specifically, if we choose $\beta = \frac{2}{\mu}$, $\gamma = \max\{8\frac{L}{\mu} - 1, E\}$ and denote $\kappa = \frac{L}{\mu}$, then $\eta_t = \frac{2}{\mu}\frac{1}{\gamma+t}$ and

$$\mathbb{E}[F(\overline{\mathbf{w}}_t)] - F^* \leq \frac{2\kappa}{\gamma+t}\left(\frac{B}{\mu} + 2L\Delta_1\right). \tag{20}$$

For $C_t > 0$ (partial participation), from Lemma 2, we know that the convergence is guaranteed by setting $\eta_t = r_t\bar{\eta}_t$, where $\bar{\eta}_t$ is the learning rate in full participation condition. Therefore, let $\eta_t = \frac{2r_t}{\mu(\gamma+t)}$ and replace B with $B + C_t$, we have

$$\mathbb{E}[F(\overline{\mathbf{w}}_t)] - F^* \leq \frac{2\kappa}{\gamma+t}\left(\frac{B+C_t}{\mu} + 2L\Delta_1\right)$$

$$\leq \frac{2\kappa}{\gamma+t}\left(\frac{B+D}{\mu} + 2L\Delta_1\right), \tag{21}$$

where $D = 4E^2G^2$ is the upper bound of C_t. ■

REFERENCES

[1] S. Dang, O. Amin, B. Shihada, and M.-S. Alouini, "What should 6G be?" *Nat. Electron.*, vol. 3, no. 1, pp. 20–29, 2020.

[2] M. Abrams, J. Abrams, P. Cullen, and L. Goldstein, "Artificial intelligence, ethics, and enhanced data stewardship," *IEEE Security Privacy*, vol. 17, no. 2, pp. 17–30, Mar./Apr. 2019.

[3] S. Dang, C. Zhang, B. Shihada, and M.-S. Alouini, "Big communications: Connect the unconnected," 2021, *arXiv:2104.06131*.

[4] W. Y. B. Lim *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart., 2020.

[5] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Wireless communications for collaborative federated learning," *IEEE Commun. Mag.*, vol. 58, no. 12, pp. 48–54, Dec. 2020.

[6] M. Lesk, "Big data, big brother, big money," *IEEE Security Privacy*, vol. 11, no. 4, pp. 85–89, Jul./Aug. 2013.

[7] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Netw.*, vol. 33, no. 5, pp. 156–165, Sep./Oct. 2019.

[8] J. Dean *et al.*, "Large scale distributed deep networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, vol. 1, 2012, pp. 1223–1231.

[9] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade* (Lecture Notes in Computer Science, 7700), G. Montavon, G. B. Orr, and K. R. Müller, Eds. Berlin, Germany: Springer, 2012. [Online]. Available: https://doi.org/10.1007/978-3-642-35289-8_25

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

[11] A. H. Bond and L. Gasser, *Readings in Distributed Artificial Intelligence*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2014.

[12] J. Zhou, S. Dang, B. Shihada, and M.-S. Alouini, "Power allocation for relayed OFDM with index modulation assisted by artificial neural network," *IEEE Wireless Commun. Lett.*, vol. 10, no. 2, pp. 373–377, Feb. 2021.

[13] S. Dang, M. Wen, S. Mumtaz, J. Li, and C. Li, "Enabling multi-carrier relay selection by sensing fusion and cascaded ANN for intelligent vehicular communications," *IEEE Sensors J.*, vol. 21, no. 14, pp. 15614–15625, Jul. 2021.

[14] J. Li, S. Dang, M. Wen, Z. Zhang, and Q. Li, "Smart detection using the cascaded artificial neural network for OFDM with subcarrier number modulation," *IEEE Wireless Commun. Lett.*, vol. 10, no. 6, pp. 1227–1231, Jun. 2021.

[15] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[16] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Stat.*, 2017, pp. 1273–1282.

[17] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 72–80, Apr. 2020.

[18] S. Hosseinalipour, C. G. Brinton, V. Aggarwal, H. Dai, and M. Chiang, "From federated to fog learning: Distributed machine learning over heterogeneous wireless networks," *IEEE Commun. Mag.*, vol. 58, no. 12, pp. 41–47, Dec. 2020.

[19] Y. Liu, J. Peng, J. Kang, A. M. Iliyasu, D. Niyato, and A. A. A. El-Latif, "A secure federated learning framework for 5G networks," *IEEE Wireless Commun.*, vol. 27, no. 4, pp. 24–31, Aug. 2020.

[20] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities, and challenges," *IEEE Commun. Mag.*, vol. 58, no. 6, pp. 46–51, Jun. 2020.

[21] C. T. Dinh *et al.*, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 398–409, Feb. 2021.

[22] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. IEEE INFOCOM*, Paris, France, Apr. 2019, pp. 1387–1395.

[23] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 269–283, Jan. 2021.
[24] K. Bonawitz *et al.*, "Towards federated learning at scale: System design," 2019, *arXiv:1902.01046*.
[25] C. He *et al.*, "FEDML: A research library and benchmark for federated machine learning," 2020, *arXiv:2007.13518*.
[26] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc., 2019, pp. 8024–8035.
[27] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.
[28] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*.
[29] P. Han, S. Wang, and K. K. Leung, "Adaptive gradient sparsification for efficient federated learning: An online learning approach," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2020, pp. 300–310.
[30] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," 2017, *arXiv:1710.09854*.
[31] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Advances in Neural Information Processing Systems*, vol. 30. Red Hook, NY, USA: Curran Assoc., pp. 1709–1720, 2017.
[32] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time optimization for federated learning over wireless networks," 2020, *arXiv:2001.07845*.
[33] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive IoT networks," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4641–4654, May 2020.
[34] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6360–6368, Jul. 2020.
[35] L. Zhang, C. Zhang, and B. Shihada, "Efficient wireless traffic prediction at the edge: A federated meta-learning approach," *IEEE Commun. Lett.*, early access, Apr. 18, 2022, doi: 10.1109/LCOMM.2022.3167813.
[36] M. R. Sprague *et al.*, "Asynchronous federated learning for geospatial applications," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Disc. Databases*, 2018, pp. 21–28.
[37] G. F. Riley and T. R. Henderson, "The *ns* − 3 network simulator," in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Günes, and J. Gross, Eds. Berlin, Germany: Springer, 2010. [Online]. Available: https://doi.org/10.1007/978-3-642-12331-3_2
[38] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Noida, India: Pearson Educ., 2010.
[39] L. Rubio, J. Reig, and N. Cardona, "Evaluation of Nakagami fading behaviour based on measurements in urban scenarios," *AEU-Int. J. Electron. Commun.*, vol. 61, no. 2, pp. 135–138, 2007.
[40] B. Paolo, "Channel models for terrestrial wireless communications: A survey," ISTI, Pisa, Italy, Rep. oai:it.cnr:prodotti:160413, 2006.
[41] J. Proakis and M. Salehi, *Digital Communications*. New York, NY, USA: McGraw-Hill, 2008.
[42] S. U. Stich, "Local SGD converges fast and communicates little," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–17.
[43] J. Wang and G. Joshi, "Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms," in *Proc. ICML Workshop Coding Theory Mach. Learn.*, 2019, pp. 1–50.
[44] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-iid data," 2019, *arXiv:1907.02189*.
[45] C. Zhang, H. Zhang, D. Yuan, and M. Zhang, "Citywide cellular traffic prediction based on densely connected convolutional neural networks," *IEEE Commun. Lett.*, vol. 22, no. 8, pp. 1656–1659, Aug. 2018.
[46] C. Zhang, H. Zhang, J. Qiao, D. Yuan, and M. Zhang, "Deep transfer learning for intelligent cellular traffic prediction based on cross-domain big data," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1389–1401, Jun. 2019.
[47] C. Zhang, S. Dang, B. Shihada, and M.-S. Alouini, "Dual attention-based federated learning for wireless traffic prediction," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2021, pp. 1–10.
[48] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[49] G. Barlacchi *et al.*, "A multi-source dataset of urban life in the city of Milan and the province of Trentino," *Sci. Data*, vol. 2, Oct. 2015, Art. no. 150055.
[50] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proc. USENIX Annu. Tech. Conf.*, vol. 14. Boston, MA, USA, 2010, pp. 1–14.
[51] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 634–643.
[52] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2020, pp. 2938–2948.

GUOQING MA (Graduate Student Member, IEEE) received the B.Sc. degree from the South University of Science and Technology of China, Shenzhen, China, in 2017. He is currently pursuing the Ph.D. degree in computer science with the King Abdullah University of Science and Technology, Thuwal, Saudi Arabia. His main research interests include big data analysis and large-scale system design.

CHUANTING ZHANG (Member, IEEE) received the B.S. and M.S. degrees in computer science from the Inner Mongolia University of Science and Technology, Baotou, China, in 2011 and 2014, respectively, and the Ph.D. degree in communication and information systems from Shandong University, Jinan, China, in 2019. He is currently a Senior Research Associate with the University of Bristol, U.K. Previously, he was a Postdoctoral Fellow with the Computer, Electrical and Mathematical Science and Engineering Division, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia. His current research interests include spatial-temporal data analysis, federated learning, and graph mining.

SHUPING DANG (Member, IEEE) received the B.Eng. degree (First Class Hons.) in electrical and electronic engineering from the University of Manchester and the B.Eng. degree in electrical engineering and automation from Beijing Jiaotong University in 2014 via a joint "2+2" dual-degree program, and the D.Phil. degree in engineering science from the University of Oxford in 2018. He joined in the R&D Center, Huanan Communication Company Ltd., after graduating from the University of Oxford and worked as a Postdoctoral Fellow with the Computer, Electrical and Mathematical Science and Engineering Division, King Abdullah University of Science and Technology. He is currently a Lecturer with Department of Electrical and Electronic Engineering, University of Bristol. His research interests include 6G communications, wireless communications, wireless security, and machine learning for communications.

BASEM SHIHADA (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Waterloo. He is an Associate & Founding Professor with the Computer, Electrical and Mathematical Sciences and Engineering Division, King Abdullah University of Science and Technology. In 2009, he was appointed as a Visiting Faculty with the Department of Computer Science, Stanford University. His current research covers a range of topics in energy and resource allocation in wired and wireless networks, software defined networking, cloud/fog computing, Internet of Things, data networks, and underwater networks.