Multi-Edge Server-Assisted Dynamic Federated Learning With an Optimized Floating Aggregation Point

Bhargav Ganguly, Seyyedali Hosseinalipour, *Member, IEEE*, Kwang Taik Kim, *Senior Member, IEEE*, Christopher G. Brinton, *Senior Member, IEEE*, Vaneet Aggarwal, *Senior Member, IEEE*, David J. Love, *Fellow, IEEE*, and Mung Chiang, *Fellow, IEEE*

***Abstract*— We propose cooperative edge-assisted dynamic federated learning (CE-FL). CE-FL introduces a distributed machine learning (ML) architecture, where data collection is carried out at the end devices, while the model training is conducted *cooperatively* at the end devices and the edge servers, enabled via data offloading from the end devices to the edge servers through base stations. CE-FL also introduces floating aggregation point, where the local models generated at the devices and the servers are aggregated at an edge server, which varies from one model training round to another to cope with the network evolution in terms of data distribution and users' mobility. CE-FL considers the heterogeneity of network elements in terms of communication/computation models and the proximity to one another. CE-FL further presumes a *dynamic* environment with online variation of data at the network devices which causes a *drift* at the ML model performance. We model the processes taken during CE-FL, and conduct analytical convergence analysis of its ML model training. We then formulate network-aware CE-FL which aims to adaptively optimize all the network elements via tuning their contribution to the learning process, which turns out to be a non-convex mixed integer problem. Motivated by the large scale of the system, we propose a *distributed optimization solver* to break down the computation of the solution across the network elements. We finally demonstrate the effectiveness of our framework with the data collected from a real-world testbed.**

***Index Terms*— Fog learning, federated learning, distributed machine learning, cooperative learning, network optimization.**

I. Introduction

RECENT advancements in smart devices (e.g., new chipsets in phones and smart cars) have made them capable of collecting large amounts of data in real-time. In the example case of smart cars, collection of more than 4 TB of data per day is predicted [1]. Utilizing this data for training a machine learning (ML) model (e.g., for driving assistance) is the main motivation for implementing distributed ML techniques over the network edge in 6G-and-beyond systems [2].

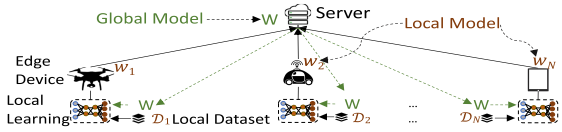


Fig. 1. Conventional architecture of federated learning.

Federated learning (FL) has been promoted as one of the key distributed ML techniques [3]. Its conventional model training architecture [4] is depicted in Fig. 1. In FL, each device trains a local model using its own dataset. The local models of devices are periodically transmitted to a server and aggregated together, e.g., via weighted averaging, to form a global model. The server broadcasts the global model among the devices to initiate the next round of local model training. FL keeps the dataset of the devices local, which is desired in applications with user privacy concerns, e.g., healthcare, and only uses the server as an *aggregator*. However, implementing FL using its conventional architecture over large-scale wireless networks with heterogeneous communication/computation capabilities faces major challenges discussed next.

A. Motivations and Challenges

As depicted in Fig. 2, we consider the implementation of distributed ML over a network of multiple user equipments (UEs), multiple edge data centers (DCs), and multiple base stations (BSs), which is a realistic model for the network edge [5]. Our foremost goal is to introduce a new ML model training architecture to execute distributed ML across the continuum of UEs to DCs. We further optimize the network-element orchestration, through a distributed algorithm, to have an efficient distributed ML model training procedure. Our system design is motivated by challenges faced with the implementation of FL model training architecture over our hierarchical network of interest summarized below:

(C1) Conventional FL neglects the computation power of the DCs and conducts ML model training solely at the UEs. Meanwhile, some of the UEs may have large volumes of data they are unable to process. At the same time, these resource-constrained UEs might be located near BSs with access to powerful DCs that can efficiently process data. Also, excessive computation

Manuscript received 25 March 2022; revised 17 September 2022 and 23 December 2022; accepted 9 March 2023; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Ioannidis. This work was supported in part by NSF under Grant CNS-2146171, Grant CCF1816013, and Grant EEC1941529; in part by NSC under Grant W15QKN-15-9-1004; in part by ONR under Grant N000142112472; in part by Cisco Inc.; and in part by ARL under Grant W911NF2020221. *(Bhargav Ganguly and Seyyedali Hosseinalipour contributed equally to this work.) (Corresponding author: Seyyedali Hosseinalipour.)*

Bhargav Ganguly and Vaneet Aggarwal are with the Department of Industrial Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: bganguly@purdue.edu; vaneet@purdue.edu).

Seyyedali Hosseinalipour is with the Department of Electrical Engineering, University at Buffalo-SUNY, Buffalo, NY 14228 USA (e-mail: alipour@buffalo.edu).

Kwang Taik Kim, Christopher G. Brinton, David J. Love, and Mung Chiang are with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: kimkt@purdue.edu; cgb@purdue.edu; djlove@purdue.edu; chiang@purdue.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNET.2023.3262482, provided by the authors.

Digital Object Identifier 10.1109/TNET.2023.3262482

power of DCs can out-weigh the added latency of data transfer (i.e., the latency caused by data transfer from the straggling UEs to the DCs can be compensated by faster data processing at DCs).

(C2) It is not clear which DC should be selected as the model aggregator since the distribution of data changes across time at the UEs. Also, UEs have different channels to the BSs, while different BSs have different delay of data/parameter transfer to different DCs.

(C3) There exists heterogeneity across the DCs and the UEs in terms of computation and communication capability. Thus, model training cannot be conducted arbitrary across the network elements. Also, it is not practical to consider the existence of a central controller with global knowledge about the characteristics of the UEs, BSs, and DCs that can solely obtain all the network orchestration decisions.

(C4) Although data offloading is not considered in the conventional FL architecture [6], it is a viable mechanism for many applications of FL. This is due to the fact that in many FL applications users do not have strict privacy concerns on their local data. For example, autonomous driving is one of the envisioned applications of FL, wherein data collected by the cars are processed for training [7]. In such a setting, with some form of incentivization (e.g., cashback points and gas credits), vehicle users may become willing to share their non-private data such as automobile sensor measurements and pictures of traffic signs. Furthermore, in privacy-sensitive applications, FL can be combined with research in *private representation learning* [8], which aims to tackle the privacy concerns associated with the transfer of raw data by obfuscating sensitive information attributes of the users. Also, encryption methods [9] can help to facilitate end-to-end data transfer between end devices and trusted servers.

To respond to the aforementioned challenges, we propose cooperative edge-assisted dynamic/online federated learning (CE−FL). CE−FL addresses (C1)&(C4) via exploiting the computation capability of the edge servers in ML model training, and considering *cooperative process of data points* between the end devices and the servers, where the devices offload a portion of their datasets to the BSs, which in turn disperse the received data across the servers. To address (C2), CE−FL introduces the idea of *floating aggregation point*, where the edge server in charge of aggregating the models varies from one model training round to another and is chosen efficiently based on the dynamics of data variations across the devices captured via *model/concept drift*, and the innate characteristics of the servers, BSs, and the end devices. To account for (C3), CE−FL considers heterogeneity of (i) the number of stochastic gradient descent (SGD) iterations used for model computation across the end devices and the edge servers, (ii) the mini-batch sizes of the SGDs, and (iii) the load and power consumption models across the devices and the servers. Finally, CE−FL introduces a *distributed network orchestration* technique, under which the contribution of all the network elements to ML model training (e.g., number of local SGD iterations, data offloading configuration and routing across the network) is optimized.

CE−FL conducts cooperative model training exploiting computation capability of both the devices and the servers, while achieving (a) a balanced computation load in the device layer and edge server layer; (b) an efficient data dispersion/routing from the device layer to the BS layer and from the BS layer to the edge server layer; and (iii) an efficient parameter aggregation via selecting a floating aggregation point.

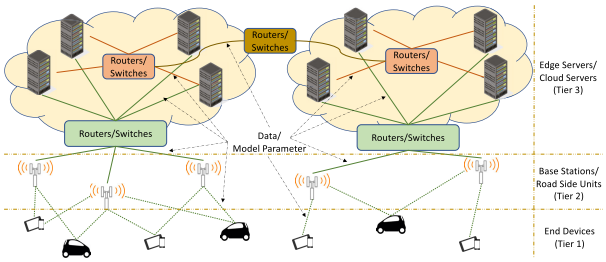


Fig. 2. A schematic of our network model. We consider a three tiered network consisting of end devices, base stations, and edge servers.

B. Related Work

Conventional FL. FL has attracted tremendous attention from both ML and wireless networking communities. In the former literature [10], [11], fundamental convergence of FL and a variety of new distributed learning techniques inspired by FL have been invented, e.g., fully decentralized learning [11]. In the networking literature [12], [13], [14], [15], [16], researchers have been mostly studying the performance of FL under communication and computation heterogeneity. For example, researchers have studied the model training performance under noisy channels [14], limited energy devices [12], limited bandwidth [13], and wireless aggregation of signals over the air [15]. Also, device sampling [16] has been topics of research. Furthermore, a part of literature focuses on adapting FL for new technologies, such as unmanned aerial vehicles [17], [18] and massive MIMO [19]. As compared to this literature, we consider a different network model focusing on the distribution of ML model training across the UE-BS-DC hierarchy.

New Network Architectures for FL. Some recent works promote migrating from FL to new distributed ML architectures. In [20], fog learning is proposed that incorporates the cooperation among the devices and multi-layer architecture of fog systems into ML. In [21], collaborative FL via device-to-device (D2D) communications for model relaying is introduced. In [22], [23], semi-decentralized/hybrid FL is proposed, which augments the global aggregations of FL with local aggregations conducted via D2D communications. In [24], democratized FL is studied to exploit the innate capability of heterogeneous devices to train an ML model. Finally, parallel successive learning has been developed in [25] as a dynamic/online cooperative learning method that optimizes the utilization of D2D communications according to device heterogeneity. We contribute to this literature via proposing a novel ML network architecture consisting of multiple UEs, BSs, DCs, for which we study efficient network-element orchestration.

Hierarchical FL. One of the recent architectures of FL has been hierarchical FL [26], [27], [28]. The literature in this area is focused on managing the multi-stage model aggregations across the end device-to-cloud continuum. In this literature, researchers aim to reduce the latency of model training via

conducting multiple rapid local aggregations at edge servers or lower layers of the network before conducting one global aggregation at the cloud server. However, they only use the edge/cloud servers as "model aggregators" which do not conduct any data processing as in conventional FL. This neglects the abundant computation capability of servers.

FL with Data Offloading. Although data sharing is not considered in the original FL architecture [6], [29], it has been promoted and investigated by the current literature. In the popular work [30], a data sharing mechanism is deployed at the edge devices to mitigate the performance drop encountered as a consequence of non-i.i.d. local datasets. In [20] and [16], device-to-device (D2D) data offloading is leveraged to improve the efficiency of FL over heterogeneous networks. In [7], a data offloading scheme is proposed for FL to assist straggler vehicles through offloading their data to edge servers. Compared to these works, we build a novel distributed network orchestration scheme, in which the devices offload a portion of their data to edge servers through base stations. In particular, we consider joint data processing at the end devices and edge servers. This leads us to consider joint load-balancing across the end devices and the edge servers to process the data for the ML task, and further consider optimal data routing to transfer data across the network hierarchy. We further formalize the floating aggregator notion into the learning architecture of FL, where the aggregator server is carefully selected to optimize a trade-off between model performance, energy consumption, and delay. We also develop a distributed network element orchestration scheme to optimize ML model training. These make our system model and analysis unique and different from existing works.

C. Outline and Summary of Contributions

Our contributions can be summarizes as follows:

- We introduce and develop cooperative edge-assisted multi-edge server FL (`CE-FL`). We analytically characterize the convergence characteristics of `CE-FL`, which leads to new convergence bounds in distributed ML.
- We model the processes taken through `CE-FL` and investigate intrinsic heterogeneities existing at different network layers. Our modeling gives insights on finding the optimized device and server orchestration scheme to train an ML model, which is carried out in conjunction with the floating aggregator selection, to optimize the performance of `CE-FL`.
- We propose network-aware `CE-FL` optimization problem that captures the trade offs between ML performance, energy consumption, and delay. The optimization problem aims to configure the *macro-decisions* across the network (e.g., data offloading and routing configuration across the network, and the floating aggregator), which leads to (i) *load balancing* across the devices and servers, and (ii) optimized data routing across the network layers. It also obtains the *micro-decisions*, which tunes each network element (e.g., the CPU frequency cycles of the devices, number of local SGD iterations and SGD min-batch sizes at both end devices and edge servers).
- We carefully investigate the characteristics of the optimization problem, which turns out to be a highly non-convex mixed-integer problem. We propose a distributed network optimization solver with drawing a connection between two methods of surrogate function, used in successive convex optimization, and consensus-based optimization. We further study the optimality of our optimization solver. To the best of our knowledge, our developed solver is among the first decentralized device orchestration mechanisms in the area of network-aware distributed machine learning.

In the following, we first describe the hierarchical network structure among the UEs, BSs, and DCs, as well as conducting distributed ML under a given network element orchestration scheme in Sec. II. In Sec. III, we present our theoretical results on the ML model convergence behavior of `CE-FL`. In Sec. IV, we aim to optimize the network orchestration scheme involving ML training, data and parameter offloading, and communication/computations overheads through formulating a network optimization problem. We then develop a decentralized solver to optimize device orchestration for distributed ML in Sec. V. In Sec. VI, we present our ablation study and proof-of-concept experiments to demonstrate how the tuned parameters manifested themselves in network costs and ML performance. Finally, in Sec. VII we conclude the paper.

II. System Model and Machine Learning Task

In this section, first, we describe the network structure of our interest (Sec. II-A). Second, we introduce the dynamic model tracking problem (Sec. II-B). Third, we provide an overview of our proposed cooperative distributed ML methodology (Sec. II-C). Fourth, we detail the ML model training (Sec. II-D). We finally model communications, computations, and the floating aggregator (Sec. II-E). A summary of the key mathematical symbols that we will introduce throughout the paper can be found in Appendix A in the supplementary material.

A. Network Architecture

We consider a hierarchical edge/fog computing network [5], [20], which consists of multiple user equipments (UEs), multiple base stations (BSs), and multiple edge server/data centers (DCs). A schematic of our network is depicted in Fig. 2. The UEs, BSs, and DCs are collected via the sets $\mathcal{N}$, $\mathcal{B}$, and $\mathcal{S}$, respectively. Each user UE $n \in \mathcal{N}$ can potentially upload/receive data to/from all of the BSs although connecting to some of the BSs may require prohibitively large transmit power. Also, each BS can potentially upload/receive data to/from all the DCs, although data transfers between some BS-DC pairs may impose large delay (e.g., when data needs to go through multiple switch/routers before reaching the destination).

Our proposed framework can be considered as *an interconnection between mobile edge computing [5] and distributed ML*, where the resource limited UEs aim to exploit the abundant resource of DCs to conduct an ML task with a high efficiency.

B. Dynamic/Online Model Tracking Problem

We consider a slotted time representation of the system dynamics, where $t \in \mathbb{N} \cup \{0\}$ denotes an arbitrary time index. In our network each t is associated with one global model training and aggregation round which will be described later. We assume distributed model training for an ML task, where the data is collected at the UEs at the bottom layer of our network hierarchy. At each time instant t, let $\overline{\mathcal{D}}_n^{(t)}$ denote the

dataset at UE $n \in \mathcal{N}$ consisting of $\overline{D}_n^{(t)} \triangleq |\overline{\mathcal{D}}_n^{(t)}|$ data points. In contrast to most of the existing literature on federated learning, we consider a *dynamic environment*, in which the number of data points and subsequently the distribution of data across the clients may vary over time [25]. Each data point $\xi \in \overline{\mathcal{D}}_n^{(t)}$, $\forall n$, is associated with a feature vector, e.g., RGB colors of a picture, and a label, e.g., the weather condition in the picture.

Let $f(\mathbf{x};\xi)$ denote the loss of the ML model (e.g., a neural network classifier) on data point ξ *parameterized* by model parameter $\mathbf{x} \in \mathbb{R}^P$, where P denotes the dimension of the model parameter. At each time instant t, for a model parameter $\mathbf{x}$, each device $n \in \mathcal{N}$ is associated with a *local loss* function

$$\overline{F}_n^{(t)}(\mathbf{x}) = \frac{1}{\overline{D}_n^{(t)}} \sum_{\xi \in \overline{\mathcal{D}}_n^{(t)}} f(\mathbf{x};\xi). \tag{1}$$

The instantaneous *global loss* function of the system is

$$F^{(t)}(\mathbf{x}) = \frac{1}{D^{(t)}} \sum_{n \in \mathcal{N}} D_n^{(t)} \overline{F}_n^{(t)}(\mathbf{x}), \tag{2}$$

where $D^{(t)} = \sum_{n \in \mathcal{N}} \overline{D}_n^{(t)}$ is the total number of data points.

The goal of the *dynamic/online model tracking* is to minimize the instantaneous global loss function, i.e., at each time instant t, it aims to find the model parameter $\mathbf{x}^{(t)^\star}$ such that

$$\mathbf{x}^{(t)^\star} = \arg\min_{\mathbf{x} \in \mathbb{R}^P} F^{(t)}(\mathbf{x}). \tag{3}$$

Let T denote the duration of ML model training and $[T] \triangleq \{0, \ldots, T-1\}$. It is evident that the sequence $\{\mathbf{x}^{(t)^\star}\}_{t \in [T]}$ is a function of devices local dataset dynamics. In our ML modeling, we quantify dynamics of data distributions at the devices via introducing *model/concept drift* in Definition 1 in Sec. III. *Model/concept drift* draws a mathematical connection between the ML model performance and ML model training delay.

The ML training formulation described by (3) aims to obtain optimal instantaneous models for conducting downstream tasks in real-time suitable for time-varying datasets at the devices [31]. We will later incorporate the performance of online model training obtained in Sec. III into our network element orchestration formulation in Sec. IV, wherein the inherent trade-off between instantaneous ML performance, energy consumption, and delay is optimized.

C. CE-FL Overview

We introduce *cooperative edge-assisted federated learning* (CE-FL) to conduct distributed ML over our hierarchical network. To solve (3), in CE-FL, part of the dataset of the UEs are offloaded to the DCs. CE-FL then simultaneously exploits the computation resources at the UEs (bottom layer) and the DCs (top layer) for ML model training, while the BSs (middle layer) are used for data and model parameter relaying. In particular, CE-FL encompasses four processes: (i) UEs data offloading to BSs, (ii) BSs dispersing data among DCs, (iii) UEs and DCs conducting model training, and (iv) model parameter transfer/pulling and aggregation at a floating aggregation DC, which are describe below.

1) *Data offloading from UEs to BSs*: At time instant t, each UE offloads/disperses part of its generated data across a subset of BSs. After the data offloading process, we denote the collected dataset at the each BS $b \in \mathcal{B}$ via $\mathcal{D}_b^{(t)}$ and the remaining dataset at each UE $n \in \mathcal{N}$ by $\mathcal{D}_n^{(t)} \subseteq \overline{\mathcal{D}}_n^{(t)}$.
2) *Data relaying from BSs to the DCs*: Each BS offloads/disperses its collected data among a subset of DCs. Since BSs do not have computation power, no data is kept at the BSs. We use $\mathcal{D}_s^{(t)}$ to denote the dataset collected at DC $s \in \mathcal{S}$.
3) *ML model training at UEs and DCs*: Each UE executes ML model training on the its remaining dataset. Also, ML model training begins at the DCs once BSs relay their data.
4) *Parameter transfer across the network and aggregation at a floating point*: After model training ends, the local ML models of UEs and DCs are aggregated at a *floating* DC, which varies from one model training round to another. The aggregator is adaptively chosen, while taking into account energy and delay of parameter transfer across the network.

We refer to the set of UEs and DCs as *data processing units* (DPU) and use index $i \in \mathcal{N} \cup \mathcal{S}$ to refer to an arbitrary UE/DC. Next, we will carefully model and formulate the four processes outlined above. We first describe the local model training for a given dataset at the UEs and DCs and then describe how the data and parameter offloading and relaying are carried out.

D. Distributed Model Training in CE-FL

As described above, in CE-FL, parts of the local datasets of the UEs gets transferred to the DCs through BSs and DCs possess a collected dataset at each time (model training round) t. We will describe how the data gets transferred in Sec. II-E and obtain the optimal data configuration and routing across the network in Sec. IV. In the following, we describe the procedure carried out across the UEs and DCs to conduct ML model training for an arbitrary data configuration across them.

To capture the performance of the ML model across UEs and DCs, we describe the *local ML loss* at each DPU (i.e., a UE or a DC) $i \in \mathcal{N} \cup \mathcal{S}$ at time t as[1]

$$F_i^{(t)}(\mathbf{x}) = \frac{1}{D_i^{(t)}} \sum_{\xi \in \mathcal{D}_i^{(t)}} f(\mathbf{x};\xi). \tag{4}$$

Since (2) measures the ML loss per *data point* and data points do not get generated and lost during the data transfer stage of CE-FL, it is straightforward to verify that $F^{(t)}(\mathbf{x})$ in (2) can be equivalently written as $F^{(t)}(\mathbf{x}) = \frac{1}{D^{(t)}} \sum_{i \in \mathcal{N} \cup \mathcal{S}} \mathcal{D}_i^{(t)} \overline{F}_i^{(t)}(\mathbf{x})$.

To solve (3), CE-FL orchestrates both UEs and DCs in local ML model training, where each DPU i aims to minimize its local loss $F_i^{(t)}(\mathbf{x})$ given by (4) via *local ML model training*. To conduct local ML model training we exploit the FedProx method [32], which consists of a series of local stochastic gradient descent (SGD) iterations at each DPU i on the *regularized local loss* using its local dataset $\mathcal{D}_i^{(t)}$. FedProx is shown to be particularly effective for handling non-iid data across the clients which is the case in our system of interest. To cope with the communication/computation heterogeneities of network elements, CE-FL uses FedProx with *different* number of local SGD iterations and mini-batch sizes across the network elements, which is among the first in literature.

In CE-FL, the ML model training starts with an initial model broadcast $\mathbf{x}^{(0)}$ from a predetermined DC across all the DPUs. Each subsequent global model training round $t \geq 1$ starts with broadcasting of a global model $\mathbf{x}^{(t)}$ from the respective elected floating aggregation DC. Given $\mathbf{x}^{(t)}$, each

[1] As compared to (1), the loss in (4) is defined on the actual dataset under which the ML model training is carried out in the index of the summation.

DPU i first initializes its local model parameter as $\mathbf{x}_i^{(t,0)} = \mathbf{x}^{(t)}$ and then conducts $\gamma_i^{(t)} \in \mathbb{N}$ local descent updates on its regularized local loss $g_i(\mathbf{x}, \mathbf{x}^{(t)}) = F_i^{(t)}(\mathbf{x}) + \frac{\mu}{2}\|\mathbf{x} - \mathbf{x}^{(t)}\|^2$, which ensures the closeness of the local model to the global model $\mathbf{x}^{(t)}$ controlled by regularization parameter μ. The evolution of the local model of each DPU i during the local descent iterations is given by

$$\mathbf{x}_i^{(t,k)} = \mathbf{x}_i^{(t,k-1)} - \eta \widetilde{\nabla} g_i(\mathbf{x}_i^{(t,k-1)}, \mathbf{x}^{(t)}),\ k = 1, \cdots, \gamma_i^{(t)}, \tag{5}$$

where η denotes the step size and $\widetilde{\nabla} g_i(\mathbf{x}_i^{(t,k-1)}, \mathbf{x}^{(t)})$ is the *stochastic* gradient of the regularized local loss given by

$$\widetilde{\nabla} g_i(\mathbf{x}_i^{(t,k-1)}, \mathbf{x}^{(t)}) = \widetilde{\nabla} F_i^{(t)}(\mathbf{x}_i^{(t,k-1)}) + \mu(\mathbf{x}_i^{(t,k-1)} - \mathbf{x}^{(t)}). \tag{6}$$

The stochastic gradient of local loss function $\widetilde{\nabla} F_i^{(t)}(\mathbf{x}_i^{(t,k-1)})$ is obtained via sampling a mini-batch (i.e., collection) of data points $\mathcal{D}_i^{(t,k-1)} \subseteq \mathcal{D}_i^{(t)}$ with size $D_i^{(t,k-1)} = m_i^{(t)} D_i^{(t)}$, where $m_i^{(t)} \in (0, 1]$ denotes the mini-batch fraction, as follows:

$$\widetilde{\nabla} F_i^{(t)}(\mathbf{x}_i^{(t,k-1)}) = \frac{1}{m_i^{(t)} D_i^{(t)}} \sum_{\xi \in \mathcal{D}_i^{(t,k-1)}} \nabla f(\mathbf{x}_i^{(t,k-1)}; \xi). \tag{7}$$

Replacing (6) in (5) and recursively expanding the result implies the following relationship between the instantaneous local model and the initial local model at each DPU i:

$$\mathbf{x}_i^{(t,k)} - \mathbf{x}^{(t)} = -\eta \sum_{\ell=0}^{k-1} a_{i,\ell}^{(t)} \widetilde{\nabla} F_i^{(t)}(\mathbf{x}_i^{(t,\ell)}), \tag{8}$$

where $a_{i,\ell}^{(t)} = (1 - \eta\mu)^{\gamma_i^{(t)} - 1 - \ell}$. We further define $\mathbf{a}_i^{(t)} = \left[a_{i,0}^{(t)}, \cdots, a_{i,\gamma_i^{(t)}-1}^{(t)}\right]$.

At the end of local model training period, each DPU i computes its accumulated gradient, which using (8), can be obtained based on the difference between the final local model parameter and the initial model parameter as

$$\sum_{\ell=0}^{\gamma_i^{(t)}-1} a_{i,\ell}^{(t)} \widetilde{\nabla} F_i^{(t)}(\mathbf{x}_i^{(t,\ell)}) = \left(\mathbf{x}^{(t)} - \mathbf{x}_i^{(t,\gamma_i^{(t)})}\right)/\eta, \tag{9}$$

and subsequently obtains its *normalized* accumulated gradient

$$\mathbf{d}_i^{(t)} = \frac{1}{\|\mathbf{a}_i^{(t)}\|_1} \sum_{\ell=0}^{\gamma_i^{(t)}-1} a_{i,\ell}^{(t)} \widetilde{\nabla} F_i^{(t)}(\mathbf{x}_i^{(t,\ell)}). \tag{10}$$

The normalization is a necessity to ensure that that the global ML model is not biased towards the DPUs that conduct more local SGD iterations [33]. Since DPUs possess different number of datapoints, each DPU i sends vector $D_i^{(t)}\mathbf{d}_i^{(t)}$, called *scaled accumulated gradient*, to the selected floating aggregation server. We assume that each BS can receive scaled accumulated gradients of multiple UEs,[2] in which case it sums all the received vectors together to keep the dimension of the transmitted vector to the aggregation server fixed (e.g., see Fig. 3 of [20]). The aggregation DC finally obtains the next global model parameter via first summing all the received vectors together to obtain $\sum_{i \in \mathcal{N} \cup \mathcal{S}} D_i^{(t)}\mathbf{d}_i^{(t)}$ and then scaling the resulting vector to carry out the update

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \vartheta\eta \frac{1}{D^{(t)}} \sum_{i \in \mathcal{N} \cup \mathcal{S}} D_i^{(t)}\mathbf{d}_i^{(t)}, \tag{11}$$

where ϑ is a scaling factor introduced to compensate for the normalization introduced in (10).

Remark 1: In this work, we use a single global model for the downstream task at the devices, which is a common approach in FL literature [10], [11], [12], [13]. It is worth mentioning that personalized FL, which aims to learn a global ML model from which local models are obtained specific to the local distribution of data at the devices [34], would be another potential approach. We leave the extension of the framework of `CE-FL` *to support personalized FL as future work.*

E. Communication, Computation, and Floating Aggregation

We next describe the models for data/parameter communications, SGD local computations, and floating aggregation.

1) UEs-BSs Communications: UE-to-BS communications are carried out through wireless channels. For the uplink communications, we consider that the data rate between UE $n \in \mathcal{N}$ and BS $b \in \mathcal{B}$ at time instant t is given by

$$R_{n,b}^{(t)} = V_{n,b}^{(t)} \log_2\left(1 + P_{n,b}^{(t)} |h_{n,b}^{(t)}|^2 / (\phi_{n,b}^{(t)})^2\right), \tag{12}$$

where $V_{n,b}^{(t)}$ is the bandwidth, $(\phi_{n,b}^{(t)})^2 = N_0 V_{n,b}^{(t)}$ is the noise power with N_0 denoting the noise spectral density, $P_{n,b}^{(t)}$ is the transmit power, and $h_{n,b}^{(t)}$ is the channel gain between the respective UE-BS pair.

Remark 2: Interference caused by concurrent UE transmissions in shared spectrum bands can also impact the data rate expression in (12), which would add a coupling among transmit powers in the optimization in Sec. IV. We do not incorporate interference management in our analysis, similar to works on computation offloading in multi-BS systems, multi-DC mobile edge computing systems [5], and related work in the FL literature focused on MAC layer design [13]. In Appendix K in the supplementary material, we discuss how our methodology can be complemented with a network interference management framework.

Similarly, for the downlink, the data rate between BS b and UE n is given by

$$R_{b,n}^{(t)} = V_b^{(t)} \log_2\left(1 + P_b^{(t)} |h_{b,n}^{(t)}|^2 / (\phi_b^{(t)})^2\right), \tag{13}$$

where $V_b^{(t)}$ denotes the downlink channel bandwidth, $P_b{}^{(t)}$ is the BS b transmit power, $h_{b,n}^{(t)}$ is the channel gain between the respective BS and UE, and $(\phi_b^{(t)})^2 = N_0 V_b^{(t)}$ is the noise power. Note that the BS-to-UEs communications are performed in *broadcast* mode and thus the BS does not use different transmit powers to transmit signals to different UEs.

2) BSs-DCs Communications: BS-to-DC communications are mostly enabled via wireline communications. We consider the deployed data rate $R_{b,s}^{(t)}$ for communication from BS $b \in \mathcal{B}$ to DC $s \in \mathcal{S}$, which is later optimized. Furthermore, for the uplink communication channel between BSs-DCs we denote the maximum rate over the link between an arbitrary (b, s) pair as $R_{b,s}^{\max}$. Also, the maximum capacity, in terms of the arriving data rate, of an arbitrary server is denoted by $R_s^{\max}$. Therefore, we have the following constraints pertaining to the uplink rates at global aggregation round t:

$$0 \leq R_{b,s}^{(t)} \leq R_{b,s}^{\max},\ \forall b \in \mathcal{B},\ \forall s \in \mathcal{S},\ \forall t \in [T], \tag{14}$$

$$\sum_{b \in \mathcal{B}} R_{b,s}^{(t)} \leq R_s^{\max}\ \forall s \in \mathcal{S},\ \forall t \in [T]. \tag{15}$$

We further let $R_{s,b}^{(t)}$ denote the data rate for communication from DC $s \in \mathcal{S}$ to BS $b \in \mathcal{B}$ in downlink. Since the downlink

[2]Our modeling in Sec. II-E ensures that each UE is associated with one BS during scaled accumulated gradient transfer to avoid reception of multiple replicas of the gradient of the same UE at the aggregation server.

is only used for the broadcast of ML model parameter, the capacity constraints are not considered.

3) DCs-DCs Communications: Similar to BS-to-DC communications, DC-to-DC information exchange mostly occur over wirelines. We let $R^{(t)}_{s,s'}$ denote the data rate between DC s and DC s' at time instant t, which is time varying due to the congestion over the links. In general, $R^{(t)}_{s,s'} \neq R^{(t)}_{s',s}$, $\forall s, s' \in \mathcal{S}$. Similarly, since DC-to-DC communications are only used for ML parameter aggregations and broadcasting, the capacity constraints are not considered.

4) Data Configuration at the UE Devices, BSs, and DCs: To capture the data offloading at each UE n, we introduce a continues variable $\rho^{(t)}_{n,b} \in [0,1]$ to denote the fraction of dataset of that UE which is offloaded to BS b at time t, where $\sum_{b\in\mathcal{B}} \rho^{(t)}_{n,b} \leq 1$, $\forall n \in \mathcal{N}$ since some part of the dataset might be kept local. Given the initial dataset at UE n (i.e., $\overline{\mathcal{D}}^{(t)}_n$ with size $\overline{D}^{(t)}_n$) and the remaining dataset at UE n (i.e., $\mathcal{D}^{(t)}_n \subseteq \overline{\mathcal{D}}^{(t)}_n$ with size $D^{(t)}_n \triangleq |\mathcal{D}^{(t)}_n|$), let $\mathcal{D}^{(t)}_{n,b} \subseteq \mathcal{D}^{(t)}_n$ with size $D^{(t)}_{n,b} \triangleq |\mathcal{D}^{(t)}_{n,b}|$ denote the offloaded dataset from UE n to BS b, where

$$D^{(t)}_{n,b} = \overline{D}^{(t)}_n \rho^{(t)}_{n,b}, \quad D^{(t)}_n = \left(1 - \sum_{b\in\mathcal{B}} \rho^{(t)}_{n,b}\right) \overline{D}^{(t)}_n. \tag{16}$$

The dataset collected at each BS $b \in \mathcal{B}$ denoted by $\mathcal{D}^{(t)}_b$ is the union of the received data points from UEs with size

$$D^{(t)}_b \triangleq |\mathcal{D}^{(t)}_b| = \sum_{n\in\mathcal{N}} \rho^{(t)}_{n,b} \overline{D}^{(t)}_n. \tag{17}$$

The BS then disperses all of its collected data across a subset of DCs since it is assumed to have no data processing power. At time t, let $\rho^{(t)}_{b,s} \in [0,1]$ denote the fraction of dataset of BS $b \in \mathcal{B}$ offloaded to DC $s \in \mathcal{S}$, where $\sum_{s\in\mathcal{S}} \rho^{(t)}_{b,s} = 1$. Let $\mathcal{D}^{(t)}_{b,s} \subseteq \mathcal{D}^{(t)}_b$ denote the portion of dataset transferred from BS b to DC s and $\mathcal{D}^{(t)}_s$ the dataset collected at DC s. We have

$$D^{(t)}_{b,s} \triangleq |\mathcal{D}^{(t)}_{b,s}| = \rho^{(t)}_{b,s} D^{(t)}_b, \;\; D^{(t)}_s \triangleq |\mathcal{D}^{(t)}_s| = \sum_{b\in\mathcal{B}} \rho^{(t)}_{b,s} D^{(t)}_b. \tag{18}$$

Each DPU $i \in \mathcal{N} \cup \mathcal{S}$ (i.e., a UE or a DC) subsequently uses its local dataset $\mathcal{D}^{(t)}_i$ (i.e., $\mathcal{D}^{(t)}_n$ in case of UE n and $\mathcal{D}^{(t)}_s$ in case of DC s) to conduct ML model training.

Remark 3: We note that some recent works have introduced GANs in federated learning [35], [36]. However, GAN architectures, given their long training duration, do not produce favorable results upon being implemented for time-varying data distributions. Consequently, since we consider dynamic/time-varying data distributions at the devices, it is not straightforward to replace the data offloading with GAN architectures at the DCs to reproduce the data of the UEs. Nevertheless, designing of GAN models that require short training time at the DCs and their effectiveness on reconstruction of online data at the devices are interesting future directions.

5) Energy Consumption and Delay Modeling: Energy consumption and delay are incurred by two mechanisms: (i) data/parameter transfer across the network, and (ii) data processing for ML model training across the network. We model these two mechanisms in the following.

Data and Gradient Transfer Across the Network. Let β^{D} and β^{M} denote the number of bits used to represent a data point and vector of scaled accumulated gradient, respectively. The delay of data and gradient transfer from UE n to BS b denoted by $\delta^{\mathsf{D},(t)}_{n,b}$ and $\delta^{\mathsf{M},(t)}_{n,b}$, respectively, are

$$\delta^{\mathsf{D},(t)}_{n,b} = \beta^{\mathsf{D}} \overline{D}^{(t)}_n \rho^{(t)}_{n,b} / R^{(t)}_{n,b}, \quad \delta^{\mathsf{M},(t)}_{n,b} = \beta^{\mathsf{M}} / R^{(t)}_{n,b}. \tag{19}$$

Also, the energy consumption of data and model transfer from UE n to BS b denoted by $E^{\mathsf{D}}_{n,b}$ and $E^{\mathsf{M}}_{n,b}$, respectively, are

$$E^{\mathsf{D},(t)}_{n,b} = \delta^{\mathsf{D},(t)}_{n,b} P^{(t)}_{n,b}, \quad E^{\mathsf{M},(t)}_{n,b} = \delta^{\mathsf{M},(t)}_{n,b} P^{(t)}_{n,b}. \tag{20}$$

Similarly, the delay of data and parameter transfer from BS b to DC s denoted by $\delta^{\mathsf{D},(t)}_{b,s}$ and $\delta^{\mathsf{M},(t)}_{b,s}$, respectively, are

$$\delta^{\mathsf{D},(t)}_{b,s} = \beta^{\mathsf{D}} D^{(t)}_b \rho^{(t)}_{b,s} / R^{(t)}_{b,s}, \quad \delta^{\mathsf{M},(t)}_{b,s} = \beta^{\mathsf{M}} / R^{(t)}_{b,s}. \tag{21}$$

Thus, assuming that the data transfer from the BSs to the DCs starts after reception of all the datapoints at all the BSs,[3] the delay of data collection at each DC $s \in \mathcal{S}$ is given by

$$\delta^{\mathsf{D},(t)}_s = \max_{b\in\mathcal{B}} \{\delta^{\mathsf{D},(t)}_{b,s}\} + \max_{n\in\mathcal{N}, b\in\mathcal{B}} \{\delta^{\mathsf{D},(t)}_{n,b}\}. \tag{22}$$

The energy consumption of data and parameter transfer from BS b to DC s denoted by $E^{\mathsf{D},(t)}_{b,s}$ and $E^{\mathsf{M},(t)}_{b,s}$, respectively, are

$$E^{\mathsf{D},(t)}_{b,s} = \delta^{\mathsf{D},(t)}_{b,s} P^{(t)}_{b,s}, \quad E^{\mathsf{M},(t)}_{b,s} = \delta^{\mathsf{M},(t)}_{b,s} P^{(t)}_{b,s}, \tag{23}$$

where $P^{(t)}_{b,s}$ is the transmit power consumption (during one second period) over the outgoing link from BS b to DC s.

Communications between two DCs are only conducted for parameter transfer and aggregation in our system. The delay and energy associated with parameter transfer between two DCs $s, s' \in \mathcal{S}$ denoted by $\delta^{\mathsf{M},(t)}_{s,s'}$ and $E^{\mathsf{M},(t)}_{s,s'}$, respectively, are

$$\delta^{\mathsf{M},(t)}_{s,s'} = \beta^{\mathsf{M}} / R^{(t)}_{s,s'}, \quad E^{\mathsf{M},(t)}_{s,s'} = \delta^{\mathsf{M},(t)}_{s,s'} P^{(t)}_{s,s'}, \tag{24}$$

where $P^{(t)}_{s,s'}$ is the transmit power consumption over the outgoing wirelines from DC s to DC s' ($\delta^{\mathsf{M},(t)}_{s,s'} = 0$ if $s = s'$.).

Data Processing Across the Network. For UE n, let $f^{(t)}_n \in [f^{\mathsf{min}}_n, f^{\mathsf{max}}_n]$ denote the CPU frequency used to process the datapoints at time t and c_n denote the number of required CPU cycles to process one data point. Then, the delay and energy consumption of data processing at UE n are [13]

$$\delta^{\mathsf{P},(t)}_n = c_n \gamma^{(t)}_n m^{(t)}_n D^{(t)}_n / f^{(t)}_n, \tag{26}$$

$$E^{\mathsf{P},(t)}_n = c_n \gamma^{(t)}_n m^{(t)}_n \mathcal{D}^{(t)}_n (f^{(t)}_n)^2 \alpha_n / 2. \tag{27}$$

In (26) and (27), $\gamma^{(t)}_n$ is the number of SGD iterations at the device, $m^{(t)}_n \in [0,1]$ is the mini-batch ratio (i.e., the fraction of $D^{(t)}_n$ data points processed at the device in each SGD iterations), and $\alpha_n/2$ is the device effective chip-set capacitance.

Also, for each DC s, we adopt the DC data processing model [37], [38], where each DC s consists of M_s identical server machines. Each machine is assumed to operate at the speed of processing $z^{(t)}_s$ data points per second. Let C_s denote the processing capacity of the machines at DC s (i.e., $z^{(t)}_s \leq C_s$). The delay and energy of data processing at DC s are

$$\delta^{\mathsf{P},(t)}_s = \frac{\gamma^{(t)}_s m^{(t)}_s D^{(t)}_s}{M_s z^{(t)}_s}, \tag{28}$$

$$E^{\mathsf{P},(t)}_s = \delta^{\mathsf{P},(t)}_s \times \left[\varrho \left(\frac{z^{(t)}_s}{C_s}\right)^2 \overline{P}_s M_s + (1-\varrho) \overline{P}_s M_s\right], \tag{29}$$

where $\gamma^{(t)}_s$ is the number of SGD iterations, $m_s(t) \in [0,1]$ is the mini-batch ratio, $(1-\varrho)$ is the fraction of power consumed in idle state (typically around 0.4), and $\overline{P}_s$ is the peak energy consumption of a server belonging to DC s.

[3]This assumption is imposed to make the formulation more tractable.

Modeling the Floating Aggregation. To capture the model aggregation at the floating aggregation DC, we let the binary indicator $I_s^{(t)} \in \{0,1\}$ identify whether DC s aggregates the ML model parameters at t ($I_s^{(t)}=1$) or not ($I_s^{(t)}=0$). We assume that upon completion of the local model training, each UE n offloads its scaled accumulated gradient to a BS, who then aggregates/sums the received gradients and relays the result to the aggregation DC. For the uplink, let indicator $I_{n,b}^{(t)}$ take the value of 1 if UE n offloads its gradient to BS b after the model training, and 0 otherwise. Also, for the downlink, let indicator $I_{b,n}^{(t)}$ take the value of 1 if UE n receives the aggregated model parameters from BS b and 0 otherwise.[4]

We then express the delay and energy consumption of transferring model parameters from UE n to the aggregation server, which are denoted by $\delta_n^{\mathsf{A},(t)}$ and $E_n^{\mathsf{A},(t)}$, respectively, as

$$\delta_n^{\mathsf{A},(t)} = \sum_{b\in\mathcal{B}} \delta_{n,b}^{\mathsf{M},(t)} I_{n,b}^{(t)} + \sum_{b\in\mathcal{B}}\sum_{s\in\mathcal{S}} \delta_{b,s}^{\mathsf{M},(t)} I_s^{(t)} I_{n,b}^{(t)}, \tag{30}$$

$$E_n^{\mathsf{A},(t)} = \sum_{b\in\mathcal{B}} E_{n,b}^{\mathsf{M},(t)} I_{n,b}^{(t)} + \sum_{b\in\mathcal{B}}\sum_{s\in\mathcal{S}} E_{b,s}^{\mathsf{M},(t)} I_s^{(t)} I_{n,b}^{(t)}. \tag{31}$$

Also, the delay and energy consumption of transferring model parameters from DC s to the aggregator server are given by

$$\delta_s^{\mathsf{A},(t)} = \sum_{s'\in\mathcal{S}} \delta_{s,s'}^{\mathsf{M},(t)} I_{s'}^{(t)}, \quad E_s^{\mathsf{A},(t)} = \sum_{s'\in\mathcal{S}} E_{s,s'}^{\mathsf{M},(t)} I_{s'}^{(t)}. \tag{32}$$

Since transferring of gradient across the network occurs in parallel, the delay and energy of gradient aggregation at the aggregator DC (i.e., obtaining $\sum_{i\in\mathcal{N}\cup\mathcal{S}} D_i^{(t)}\mathbf{d}_i^{(t)}$ and conducting (11)) can be written as

$$\delta^{\mathsf{A},(t)} = \max\Big\{ \underbrace{\max_{n\in\mathcal{N}}\{\delta_n^{\mathsf{A},(t)} + \delta_n^{\mathsf{P},(t)}\}}_{(a)}, \underbrace{\max_{s\in\mathcal{S}}\{\delta_s^{\mathsf{D},(t)} + \delta_s^{\mathsf{P},(t)} + \delta_s^{\mathsf{A},(t)}\}}_{(b)} \Big\}, \tag{34}$$

$$E^{\mathsf{A},(t)} = \sum_{n\in\mathcal{N}} E_n^{\mathsf{A},(t)} + \sum_{s\in\mathcal{S}} E_s^{\mathsf{A},(t)}. \tag{35}$$

[4]Note that BSs broadcast the received aggregated global model from the floating aggregator DC. Thus, each UE may receive the aggregated global model from multiple BSs. Thus, introducing $I_{b,n}^{(t)}$ will ensure the association of a UE to at least one BS for global parameter reception.

In (34), term (a) corresponds to the time that it takes for the reception of all the gradients of the UEs, which consists of local processing time and uploading the resulting gradients to the aggregating DC from the UEs. Also, term (b) captures the time required for the reception of the gradients of the DCs at the aggregator DC, which consists of data reception time at the DCs, local processing time at the DCs, and uploading the resulting gradient to the aggregation DC. Note that data offloading to the BSs from the UEs and local processing at the UEs can be conducted in parallel, which is the reason behind the existence of the outer max function in (34) (e.g., UEs can process their local data while their offloaded data is transferred across the network hierarchy to reach the designated DCs).

After aggregation DC aggregates the models via (11), it propagates the resulting model to the BSs and the DCs. The delay and energy consumption of global parameter reception at BS b from the floating aggregator DC are given by

$$\delta_b^{\mathsf{R},(t)} = \sum_{s\in\mathcal{S}} \delta_{s,b}^{\mathsf{M},(t)} I_s^{(t)}, \quad E_b^{\mathsf{R},(t)} = \sum_{s\in\mathcal{S}} E_{s,b}^{\mathsf{M},(t)} I_s^{(t)}, \tag{36}$$

respectively. After arriving at BSs, the global model is broadcast by each BS with the delay and energy consumption of

$$\delta_b^{\mathsf{B},(t)} = \max_{n\in\mathcal{N}} \Big\{ \delta_{b,n}^{\mathsf{M},(t)} I_{b,n}^{(t)} \Big\}, \quad E_b^{\mathsf{B},(t)} = \delta_b^{\mathsf{B},(t)} P_b^{(t)}, \tag{37}$$

respectively. Also, the delay and energy consumption of parameter reception at DC s from the aggregation DC are

$$\delta_s^{\mathsf{R},(t)} = \sum_{s'\in\mathcal{S}} \delta_{s',s}^{\mathsf{M},(t)} I_{s'}^{(t)}, \quad E_s^{\mathsf{R},(t)} = \sum_{s'\in\mathcal{S}} E_{s',s}^{\mathsf{M},(t)} I_{s'}^{(t)}, \tag{38}$$

respectively. The overall delay and energy consumption of parameter reception from the aggregation DC are

$$\delta^{\mathsf{R},(t)} = \max\Big\{ \max_{b\in\mathcal{B}}\{\delta_b^{\mathsf{R},(t)} + \delta_b^{\mathsf{B},(t)}\}, \max_{s\in\mathcal{S}}\{\delta_s^{\mathsf{B},(t)}\} \Big\}, \tag{39}$$

$$E^{\mathsf{R},(t)} = \sum_{b\in\mathcal{B}} \big[E_b^{\mathsf{R},(t)} + E_b^{\mathsf{B},(t)}\big] + \sum_{s\in\mathcal{S}} E_s^{\mathsf{R},(t)}, \tag{40}$$

respectively.

III. ML Convergence Analysis of CE-FL

We next investigate the ML model performance obtained via CE-FL. In our analysis, we make the following assumptions:

Assumption 1 (Smoothness): For each DPU $i \in \mathcal{N}\cup\mathcal{S}$, the local loss function is L-smooth, i.e., $\|\nabla F_i^{(t)}(\mathbf{x}) - \nabla F_i^{(t)}(\mathbf{y})\| \leq L\|\mathbf{x}-\mathbf{y}\|$, $\forall \mathbf{x},\mathbf{y}\in\mathbb{R}^p$, where $\|.\|$ denotes the 2-norm.

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\Big[\big\|\nabla F^{(t)}(\mathbf{x}^{(t)})\big\|^2\Big] \leq \underbrace{\frac{4(F^{(0)}(\mathbf{x}^0)-F^\star)}{\vartheta\eta T}}_{(a)} + \frac{4}{\vartheta\eta T}\underbrace{\sum_{t=0}^{T-1}\sum_{i\in\mathcal{N}\cup\mathcal{S}}\tau^{(t)}\Delta_i^{(t)}}_{(b)}$$
$$+ 16\eta L\vartheta\underbrace{\left[\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i\in\mathcal{N}\cup\mathcal{S}}\frac{(p_i^{(t)})^2(1-m_i^{(t)})\Theta_i(\tilde{\sigma}_i^{(t)})^2}{m_i^{(t)}D_i^{(t)}}\frac{\|\mathbf{a}_i^{(t)}\|_2^2}{\|\mathbf{a}_i^{(t)}\|_1^2}\right]}_{(c)}$$
$$+ 12\eta^2L^2\zeta_2\underbrace{\left(\max_{t\in[T]}\max_{i\in\mathcal{N}\cup\mathcal{S}}\frac{(\gamma_i^{(t)})^2(\|\mathbf{a}_i^{(t)}\|_1 - [a_{i,-1}^{(t)}])}{\|\mathbf{a}_i^{(t)}\|_1}\right)}_{(d)}$$
$$+ 12\eta^2L^2\underbrace{\left[\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i\in\mathcal{N}\cup\mathcal{S}}\frac{(1-m_i^{(t)})(D_i^{(t)}-1)\Theta_i(\tilde{\sigma}_i^{(t)})^2p_i^{(t)}\gamma_i^{(t)}}{m_i^{(t)}\|\mathbf{a}_i^{(t)}\|_1(D_i^{(t)})^2}(\|\mathbf{a}_i^{(t)}\|_2^2 - [a_{i,-1}^{(t)}]^2)\right]}_{(e)} \tag{25}$$

Assumption 2 (Local Data Variability): The local data variability at each DPU $i \in \mathcal{N} \cup \mathcal{S}$ is measured via a finite constant $\Theta_i \geq 0$ that satisfies the following inequality $\forall \mathbf{x} \in \mathbb{R}^p$:

$$\|\nabla f(\mathbf{x};\xi) - \nabla f(\mathbf{x};\xi')\| \leq \Theta_i \|\xi - \xi'\|, \forall \xi, \xi' \in \mathcal{D}_i^{(t)}, \forall t. \tag{41}$$

Assumption (Bounded Dissimilarity of Local Loss Functions): For a set of arbitrarily normalized coefficients $\{p_i\}$, where $i \in \mathcal{N} \cup \mathcal{S}$, $p_i \geq 0$ and $\sum_{i \in \mathcal{N} \cup \mathcal{S}} p_i = 1$, there exist two finite constants $\zeta_1 \geq 1$ and $\zeta_2 \geq 0$ such that the following inequality holds for any choice of model parameter $\mathbf{x} \in \mathbb{R}^p$:

$$\sum_{i \in \mathcal{N} \cup \mathcal{S}} p_i \left\| \nabla F_i^{(t)}(\mathbf{x}) \right\|^2 \leq \zeta_1 \left\| \sum_{i \in \mathcal{N} \cup \mathcal{S}} p_i \nabla F_i^{(t)}(\mathbf{x}) \right\|^2 + \zeta_2, \forall t. \tag{42}$$

It can be seen that ζ_1 and ζ_2 in the above definition measure the level of data heterogeneity across the clients. If the data is completely homogeneous (i.e., i.i.d.) across the DPUs we will have $\zeta_1 = 1$ and $\zeta_2 = 0$ and these parameters will increase as the data across the DPUs become more heterogeneous. In Appendix I in the supplementary material, we introduce a methodology to estimate the parameters introduced in the above three assumptions.

We next define *model/concept drift*, which was first characterized by us in [25] for dynamic ML. This quantity measures the shift in the local loss imposed by the variation of the local data. We slightly modify the definition in [25] and incorporate the duration of global aggregation into the definition:

Definition 1 (Model/Concept Drift): For DPU $i \in \mathcal{N} \cup \mathcal{S}$, the online model/concept drift for two consecutive rounds of global aggregation t and $t+1$ is measured by $\Delta_i^{(t)} \in \mathbb{R}$, which captures the maximum variation of the fractional loss function for any $\mathbf{x} \in \mathbb{R}^p$ per unit time according to

$$\frac{D_i^{(t+1)}}{D^{(t+1)}} F_i^{(t+1)}(\mathbf{x}) - \frac{D_i^{(t)}}{D^{(t)}} F_i^{(t)}(\mathbf{x}) \leq \tau^{(t)} \Delta_i^{(t)}, \tag{43}$$

where $\tau^{(t)}$ is the duration between global aggregation t and $t+1$, during which model training is not conducted.

Letting data variance $\sigma_i^{(t)}$ denote the sampled variance of data at DPU i at time t, we next obtain the general convergence behavior of CE-FL (proof provided in Appendix D in the supplementary material):

Theorem 1 (General Convergence of CE-FL): Assume that η satisfies $4\eta^2 L^2 \max_{t \in [T]} \max_{i \in \mathcal{N} \cup \mathcal{S}} \frac{\gamma_i^2(t)(\|\mathbf{a}_i^{(t)}\|_1 - [a_{i,-1}^{(t)}])}{\|\mathbf{a}_i^{(t)}\|_1} \leq \frac{1}{2\zeta_1^2+1}$. Under CE-FL, the cumulative average of the global loss satisfies (25), as shown at the bottom of the previous page, where $F^* \triangleq \min_{t \in [T]} \min_{\mathbf{x} \in \mathbb{R}^p} F^{(t)}(\mathbf{x})$.

The bound in (25) demonstrates convergence characteristics of CE-FL. It reveals the relationship between the loss gap imposed based on the choice of initial model parameter (term (a)). It reveals that as data variability $\{\Theta_i\}$ and data variance $\{\sigma_i\}$ (in terms (c) and (e)) increase across the devices, the system experiences a worse convergence performance. This is due to the fact that, given a fixed mini-batch size across the devices, larger values of $\{\Theta_i\}$ and $\{\sigma_i\}$ would imply a higher noise in estimation of true gradient via SGD. Finally, the effect of increasing the local SGD iterations can be particularly seen in term (d), where as $\gamma_i^{(t)}$ increases the convergence bound increases proportionally to the data heterogeneity across the devices (ζ_2 in the coefficient of term (d)). Note that ζ_1, ζ_2 quantity the extent of spatial data heterogeneity in terms of ML model loss across the network. Thus, as the data heterogeneity increases, the ML convergence bound favors lesser locally conducted SGD iterations to avoid local model bias. Furthermore, the bound imposed on the step size η in the statement of Theorem 1 implies that as the data heterogeneity measure ζ_1 or number of local SGD iterations $\{\gamma_i\}$ increase, the smaller values of step size are tolerable to avoid local model bias.

Given the general convergence behavior obtained in Theorem 1, we next obtain a specific choice of step size, conditions on the noise of SGD, and a condition on the model drift, under which CE-FL converges (proof provided in Appendix E in the supplementary material):

Corollary 1 (Convergence of CE-FL): Consider the conditions stated in Theorem 1. Further assume that η is small enough such that $\eta = \sqrt{d/(\bar{\gamma}T)}$, where $d = |\mathcal{N} \cup \mathcal{S}|$ and $\bar{\gamma} = \sum_{t=1}^{t=T} \sum_{i \in \mathcal{N} \cup \mathcal{S}} \gamma_i^{(t)}$. If (i) the local data variability satisfy $\Theta_i \leq \Theta_{\max}$, $\forall i$, for some positive $\Theta_{\max}$, (ii) the variances of datasets satisfy $\tilde{\sigma}_i^{(t)} \leq \tilde{\sigma}_{\max}$, $\forall i$, for some positive $\tilde{\sigma}_{\max}$, (iii) the mini-batch ratios satisfy $m_i^{(t)} \geq m_{\min}$, $\forall i$, for some positive $m_{\min}$, (iv) the number of SGD iterations satisfy $\gamma_i^{(t)} \leq \gamma_{\max}$, $\forall i$, for some positive $\gamma_{\max}$, and (v) the duration of global aggregations is bounded as $\tau^{(t)} \leq \max\{\frac{\tilde{\tau}}{T \sum_{i \in \mathcal{N} \cup \mathcal{S}} \Delta_i^{(t)}}, 0\}$, for some positive $\tilde{\tau}$, then the cumulative average of the global loss satisfies (33), as shown at the bottom of the page, implying $\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\left\| \nabla F^{(t)}(\mathbf{x}^{(t)}) \right\|^2 \right] = \mathcal{O}(1/\sqrt{T})$.

Remark 4: Since time varying local loss functions are of our interest, we used the cumulative average of the online global loss as a performance metric. If data is static across the DPUs, i.e., zero model drift, the global loss function becomes time-invariant and the above corollary implies $\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\left\| \nabla F(\mathbf{x}^{(t)}) \right\|^2 \right] = \mathcal{O}(1/\sqrt{T})$, revealing that CE-FL approaches a stationary point of the global loss function.

Considering Corollary 1, guaranteeing the convergence requires sufficiently small step size, bounded noise of SGD, and reasonably fast global aggregations. In particular, the condition between $(\tau^{(t)})$ and model/concept drift $(\Delta_i^{(t)})$ implies that the speed of triggering new global aggregations should be *inversely* proportional to the model/concept drift: higher

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[\|\nabla F^{(t)}(\mathbf{x}^{(t)})\|^2 \right] \leq \frac{4\sqrt{\bar{\gamma}}}{\vartheta \sqrt{dT}} \left[F^{(0)}(\mathbf{x}^{(0)}) - F^* \right] + \frac{4\tilde{\tau}\sqrt{\bar{\gamma}}}{\vartheta \sqrt{dT}} + 16 \frac{L \vartheta \Theta_{\max} \tilde{\sigma}_{\max}^2}{m_{\min}} \sqrt{\frac{d}{\bar{\gamma} T}} + \frac{12 L^2 d \Theta_{\max} \tilde{\sigma}_{\max}^2 \gamma_{\max}}{\bar{\gamma} m_{\min} T} + \frac{12 L^2 \zeta_2 d \gamma_{\max}^2}{\bar{\gamma} T} \tag{33}$$

concept drift should be met with rapid global aggregations to ensure that the global model can *track* the variations of the UEs' datasets.

It is worth mentioning that the theoretical convergence results obtained in Theorem 1 and Corollary 1 describe the ML model performance over the online datasets at the devices used for *training*, which is a common approach in the literature of FL. In practice, the generalization of the trained model to an unseen *test* dataset is facilitated via regularization methods and proper split of test vs. train datasets [39].

IV. Network-Aware Model Training Through CE-FL

We will next tie the ML performance of CE-FL to the characteristics of the network elements. In particular, to formulate our problem, in our proposed ML model training paradigm, *heterogeneity* at four levels should be considered: (i) UEs with different computation/communication capabilities; (ii) DCs with different number of servers and power consumption profiles; (iii) BSs with different access to the DCs and data transfer rates; (iv) data distribution across the DPUs. To jointly consider all of these heterogeneities, we formulate the network-aware CE-FL:

$$
\begin{aligned}
(\boldsymbol{\mathcal{P}}) :\min\ & \pi_1 \underbrace{\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla F^{(t)}(\mathbf{x}^{(t)})\|^2\right]}_{(a)} \\
&+ \pi_2 \sum_{t=0}^{T-1} \underbrace{\left[\delta^{\mathsf{A},(t)}+\delta^{\mathsf{R},(t)}\right]}_{(b)} \\
&+ \pi_3 \sum_{t=0}^{T-1} \underbrace{\left[\pi_{3,1}\sum_{n\in\mathcal{N}}\sum_{b\in\mathcal{B}} E_{n,b}^{\mathsf{D},(t)} + \pi_{3,2}\sum_{b\in\mathcal{B}}\sum_{s\in\mathcal{S}} E_{b,s}^{\mathsf{D},(t)}\right]}_{(c)} \\
&+ \pi_3 \sum_{t=0}^{T-1} \underbrace{\left[\pi_{3,3}\sum_{n\in\mathcal{N}} E_n^{\mathsf{P},(t)} + \pi_{3,4}\sum_{s\in\mathcal{S}} E_s^{\mathsf{P},(t)}\right]}_{(d)} \\
&+ \pi_3 \sum_{t=0}^{T-1} \underbrace{\left[\pi_{3,5}E^{\mathsf{A},(t)} + \pi_{3,6}E^{\mathsf{R},(t)}\right]}_{(e)}
\end{aligned} \tag{44}
$$

$$
\begin{aligned}
\text{s.t. } & (14),(15),(16),(18),(20),(22),(23),\\
& (26),(27),(28),(29),(35),(40),\\
& \sum_{b\in\mathcal{B}}\rho_{n,b}^{(t)} \le 1, \quad n\in\mathcal{N} && \text{(44a)}\\
& \sum_{s\in\mathcal{S}}\rho_{b,s}^{(t)} = 1, \quad b\in\mathcal{B} && \text{(44b)}\\
& \sum_{s\in\mathcal{S}} I_{\mathsf{s}}^{(t)} = 1 && \text{(44c)}\\
& \sum_{b\in\mathcal{B}} I_{n,b}^{(t)} = 1, \ \forall n\in\mathcal{N} && \text{(44d)}\\
& \sum_{b\in\mathcal{B}} I_{b,n}^{(t)} = 1, \ \forall n\in\mathcal{N} && \text{(44e)}\\
& \delta_n^{\mathsf{A},(t)} + \delta_n^{\mathsf{P},(t)} \le \delta^{\mathsf{A},(t)}, \ \forall n\in\mathcal{N} && \text{(44f)}\\
& \delta_s^{\mathsf{D},(t)} + \delta_s^{\mathsf{P},(t)} + \delta_s^{\mathsf{A},(t)} \le \delta^{\mathsf{A},(t)}, \ \forall s\in\mathcal{S} && \text{(44g)}\\
& \delta_b^{\mathsf{R},(t)} + \delta_b^{\mathsf{B},(t)} \le \delta^{\mathsf{R},(t)}, \ \forall b\in\mathcal{B} && \text{(44h)}\\
& \delta_s^{\mathsf{B},(t)} \le \delta^{\mathsf{R},(t)}, \ \forall s\in\mathcal{S} && \text{(44i)}\\
& 0 \le z_s^{(t)} \le C_{\mathsf{s}}, \ s\in\mathcal{S} && \text{(44j)}\\
& 0 \le \rho_{n,b}^{(t)} \le 1, \quad n\in\mathcal{N}, b\in\mathcal{B} && \text{(44k)}\\
& 0 \le \rho_{b,s}^{(t)} \le 1, \quad b\in\mathcal{B}, s\in\mathcal{S} && \text{(44l)}\\
& f_n^{\min} \le f_n^{(t)} \le f_n^{\max}, \quad n\in\mathcal{N} && \text{(44m)}\\
& 0 \le m_i^{(t)} \le 1, \quad i\in\mathcal{N}\cup\mathcal{S} && \text{(44n)}\\
& \gamma_i^{(t)} \ge 0, \quad i\in\mathcal{N}\cup\mathcal{S} && \text{(44o)}\\
& \delta^{\mathsf{A},(t)} \ge 0, \ \delta^{\mathsf{R},(t)} \ge 0 && \text{(44p)}\\
& I_{n,b}^{(t)} \in \{0,1\}, I_{b,n}^{(t)} \in \{0,1\}, \ n\in\mathcal{N}, \ b\in\mathcal{B} && \text{(44q)}\\
& I_s^{(t)} \in \{0,1\}, \quad s\in\mathcal{S} && \text{(44r)}
\end{aligned}
$$

Variables:

$$
\begin{aligned}
&\boldsymbol{w} : \{[\rho_{n,b}^{(t)}]_{n\in\mathcal{N},b\in\mathcal{B}}, [\rho_{b,s}^{(t)}]_{b\in\mathcal{B},s\in\mathcal{S}},\\
&[f_n^{(t)}]_{n\in\mathcal{N}}, [z_s^{(t)}]_{s\in\mathcal{S}},\\
&[\gamma_i^{(t)}]_{i\in\mathcal{N}\cup\mathcal{S}}, [m_i^{(t)}]_{i\in\mathcal{N}\cup\mathcal{S}}, [I_s^{(t)}]_{s\in\mathcal{S}}, [I_{n,b}^{(t)}]_{n\in\mathcal{N},b\in\mathcal{B}},\\
&[I_{b,n}^{(t)}]_{b\in\mathcal{B},n\in\mathcal{N}}, \delta^{\mathsf{A},(t)}, \delta^{\mathsf{R},(t)}, [R_{b,s}^{(t)}]_{b\in\mathcal{B},s\in\mathcal{S}}\}_{t=1}^{T}
\end{aligned}
$$

1) Objective and Variables: The objective of $\boldsymbol{\mathcal{P}}$ captures a trade-off between ML model performance (term (a)), which is replaced with the right hand side of the bound in (33), the delay of obtaining new global parameters (term (b)), and the energy consumption of model training (term $(c),(d),(e)$). In bound (33), we replace $\tau^{(t)}$ with $\delta^{\mathsf{A},(t)}+\delta^{\mathsf{R},(t)}$, which is an upper bound on it, for the tractability of the solution. Constants π_1,π_2,π_3 weigh these (possibly competing) objectives.

Also, the constants $\pi_{3,1}-\pi_{3,6}$ in terms $(c),(d),(e)$ weigh the impact of consumed energy for data transfer (term (c)), local model computation (term (d)), and model aggregation (term (e)). The value of these coefficients may vary in different applications. The optimization variables of our problem are the UE-BS data offloading ratios $[\rho_{n,b}^{(t)}]_{n\in\mathcal{N},b\in\mathcal{B}}$, BS-DC data offloading rations $[\rho_{b,s}^{(t)}]_{b\in\mathcal{B},s\in\mathcal{S}}$, CPU frequency at the devices $[f_n^{(t)}]_{n\in\mathcal{N}}$, speed of data processing at the DCs $[z_s^{(t)}]_{s\in\mathcal{S}}$, number of SGD iterations at DPUs $[\gamma_i^{(t)}]_{i\in\mathcal{N}\cup\mathcal{S}}$, mini-batch size of SGD $[m_i^{(t)}]_{i\in\mathcal{N}\cup\mathcal{S}}$, the index of the floating aggregation DC captured via $[I_s(t)]_{s\in\mathcal{S}}$, UE-to-BS association for offloading the final UE's local gradient $[I_{n,b}^{(t)}]_{n\in\mathcal{N},b\in\mathcal{B}}$, BS-to-UE association for receiving the aggregated global model $[I_{b,n}^{(t)}]_{b\in\mathcal{B},n\in\mathcal{N}}$, $\forall t$.

Constraints (20),(23),(27),(29),(34),(35),(39), and (40), describe the terms used in the objective function. Also, constraints (44a) and (44b) guarantee a feasible data dispersion in UE-BS and BS-DC communications. Also, (44c) and (44r) ensure the existence of only one floating aggregation DC at each global aggregation. Similarly, (44d) and (44e) along with (44q) guarantee proper BS-UE communications. To help with the decomposition of the problem, we revisited (34) and considered $\delta^{\mathsf{A},(t)}$ as an optimization variable accompanied with constraints (44f) and (44g). Using a similar argument for (39), we made $\delta^{\mathsf{R},(t)}$ an optimization variable and incorporated (44h) and (44i). Finally, (44j)-(44o) ensure the feasibility of the solution.

Roughly speaking, $\boldsymbol{\mathcal{P}}$ aims to achieve the lowest model loss, while minimizing the delay and energy consumption. This will result in (i) a *simultaneous load balancing* across

the UEs and DCs for data processing, and (ii) an *efficient data/parameter routing* across the network hierarchy, and (iii) optimized floating aggregation DC that causes minimal delay and energy of parameter aggregation and reception.

2) Challenges Faced in Solving the Problem: There are three challenges faced in solving $\boldsymbol{\mathcal{P}}$ discussed below.

1) $\boldsymbol{\mathcal{P}}$ belongs to the category of mixed integer optimization problems, which are in general highly non-trivial to solve. This is due the existence of discrete/binary variables used to denote the floating server selection (i.e., $I_s^{(t)}$, $\forall s \in \mathcal{S}$) and UE-BS association in uplink and downlink communications (i.e., $I_{n,b}^{(t)}$ and $I_{b,n}^{(t)}$, $\forall b \in \mathcal{B}, n \in \mathcal{N}$) in conjunction with the rest of the continuous optimization variables.
2) The objective of $\boldsymbol{\mathcal{P}}$ given by (44) is highly non-convex with respect to the continuous optimization variables. In particular, in (44), the ML loss term (a) given by (33) is non-convex with respect to local SGD iteration counts (i.e, $\gamma_i^{(t)}$, $\forall i \in \mathcal{N} \cup \mathcal{S}$ and the offloading parameters (i.e., $\rho_{n,b}^{(t)}$ and $\rho_{b,s}^{(t)}$, $\forall n \in \mathcal{N}, b \in \mathcal{B}, s \in \mathcal{S}$, which are incorporated in the number of data points $D_i^{(t)}$ in the bound in (33) through (16) and (18)). Furthermore, term (b) in (44) is non-convex with respect to the optimization variables due to the multiplication between the optimization variables in (26) and (28), which are encapsulated in the processing delay (i.e., $\delta_s^{\mathsf{P},(t)}$ and $\delta_n^{\mathsf{P},(t)}$) in $\delta^{\mathsf{A},(t)}$ as described by (34). Similarly, the computation energy expressions given by (27) and (29) are non-convex.
3) In a large-scale network, there is no central entity to solve $\boldsymbol{\mathcal{P}}$. In particular, it is impractical to consider a central DC with the knowledge of all the DPUs capabilities and link data rates, which are prerequisites to solve the problem.

We are thus motivated to develop a unique network element orchestration scheme via (i) effective *relaxation* of the integer variables, (ii) *successive convexification* of the problem, and (iii) *distributing the solution computation* across the network elements. Nevertheless, achieving this goal is not trivial and requires a careful investigation of $\boldsymbol{\mathcal{P}}$, which is carried out next.

V. Distributed Network Orchestration in CE-FL

We next develop a distributed solution for $\boldsymbol{\mathcal{P}}$, where the computation burden of obtaining the solution is spread across the network elements (i.e., UEs, BSs, and DCs). It is worth mentioning that our methodology is among the first distributed network element orchestration schemes in the broad area of *network-aware distributed machine learning*, where we show how distributed optimization techniques can be exploited to orchestrate the devices for a distributed ML task. In our methodology, each network element will solve a reduced/truncated version of $\boldsymbol{\mathcal{P}}$ to obtain its associated optimization variables (e.g., mini-batch size and number of SGD iterations at the UEs), while forming a consensus with other network elements on the rest of the variables (e.g., the floating aggregator DC). We also study the optimality of the obtained solution.

The design of the distributed solution framework addresses the three challenges associated with $\boldsymbol{\mathcal{P}}$ (Sec. IV-.2). In the following, we discuss our approach to addressing them.

Relaxing the Integer Variables. We first relax the integer variables to be continuous (i.e., $I_s^{(t)} \in [0,1]$, $\forall s \in \mathcal{S}$, and $I_{n,b}^{(t)}, I_{b,n}^{(t)} \in [0,1]$, $\forall b \in \mathcal{B}, n \in \mathcal{N}$). Then, we *force* them to take binary values via incorporating the following constraints:

$$\sum_{s\in\mathcal{S}} I_s^{(t)}\left(1-I_s^{(t)}\right) \leq 0, \tag{45}$$

$$\sum_{b\in\mathcal{B}} I_{n,b}^{(t)}\left(1-I_{n,b}^{(t)}\right) \leq 0,\ n\in\mathcal{N}, \tag{46}$$

$$\sum_{b\in\mathcal{B}} I_{b,n}^{(t)}\left(1-I_{b,n}^{(t)}\right) \leq 0,\ n\in\mathcal{N}, \tag{47}$$

$$\sum_{s\in\mathcal{S}} I_s^{(t)} = 1, \quad \sum_{b\in\mathcal{B}} I_{n,b}^{(t)} = \sum_{b\in\mathcal{B}} I_{b,n}^{(t)} = 1,\ n\in\mathcal{N}, \tag{48}$$

$$I_s^{(t)} \in [0,1],\ s\in\mathcal{S}, \tag{49}$$

$$I_{n,b}^{(t)}, I_{b,n}^{(t)} \in [0,1],\ b\in\mathcal{B}, n\in\mathcal{N}. \tag{50}$$

The above constraints ensure that the indicated continuous variables would take binary values under any feasible solution. Also, they guarantee that *only one* DC will be selected as the aggregator and *only one* BS will be associated with each UE during uplink/downlink parameter transfer. Note that the introduced constraints in (45)-(47) are non-convex.

Distribution/Decomposition of Variables of $\boldsymbol{\mathcal{P}}$. We break down the optimization variables in $\boldsymbol{\mathcal{P}}$ into two categories: (i) *local variables*, which are obtained locally at each network element, and (ii) *shared variables*, which are first optimized locally and then synchronized via a consensus mechanism across the adjacent network elements (e.g., UE-BS or BS-DC pairs). The UE-BS offloading parameters denoted by $\{[\rho_{n,b}^{(t)}]_{n\in\mathcal{N},b\in\mathcal{B}}\}_{t=1}^T$ determine the number of datapoints accumulated at the BSs, and subsequently dictate the dataset gathered at the DCs. Thus, $\rho_{n,b}^{(t)}$ for a given UE and BS $n\in\mathcal{N}, b\in\mathcal{B}$ are shared variables. A similar justification leads to $\rho_{b,s}^{(t)}$ being shared by the constituent BS and DC $b\in\mathcal{B}, s\in\mathcal{S}$. The floating server selection indicators $\{[I_s^{(t)}]_{s\in\mathcal{S}}\}_{t=1}^T$ are also shared variables among all the network elements (i.e., $\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}$) as it impacts the delay of parameter transfer. Furthermore, it is evident from (44f)-(44g) that aggregation delay $\{[\delta^{\mathsf{A},(t)}]\}_{t=1}^T$ are shared variables for set of nodes in $\mathcal{N}\cup\mathcal{S}$. Also, (44h)-(44i) imply that the reception delay $\{[\delta^{\mathsf{R},(t)}]\}_{t=1}^T$ are shared variables between the nodes in $\mathcal{B}\cup\mathcal{S}$. Rest of the variables associated with $\boldsymbol{\mathcal{P}}$ are local variables assigned to their respective individual nodes. For instance, the variables associated with ML model training and data processing at each UE $n\in\mathcal{N}$ (i.e., SGD mini-batch ratio $\{[m_n^{(t)}]\}_{t=1}^T$, the number of SGD iterations $\{[\gamma_n^{(t)}]\}_{t=1}^T$, and CPU frequency $\{[f_n^{(t)}]\}_{t=1}^T$) are local variables.

To compute the shared variables, we first *expand* the solution vector $\boldsymbol{w}$ of problem $\boldsymbol{\mathcal{P}}$ by creating *local copies* of the shared variables at their constituent nodes and introducing equality constraints to enforce agreement among the local copies. We summarize the variables computed at each node below:

1) Each UE n: $\boldsymbol{w}_n^{\mathsf{Local}} = \left\{[f_n^{(t)}], [m_n^{(t)}], [\gamma_n^{(t)}], [I_{n,b}^{(t)}]_{b\in\mathcal{B}}\right\}_{t=1}^T$, $\boldsymbol{w}_n^{\mathsf{Shared}} = \left\{[\rho_{n,b}^{(t),n}]_{b\in\mathcal{B}}, [I_s^{(t),n}]_{s\in\mathcal{S}}, [\delta^{\mathsf{A},(t),n}]\right\}_{t=1}^T$, where $\rho_{n,b}^{(t),n}, I_s^{(t),n}, \delta^{\mathsf{A},(t),n}$ denote the local copies of the shared variables (i.e., $\rho_{n,b}^{(t)}, I_s^{(t)}, \delta^{\mathsf{A},(t)}$) at node n.
2) Each BS b: $\boldsymbol{w}_b^{\mathsf{Local}} = \left\{[I_{b,n}^{(t)}]_{n\in\mathcal{N}}\right\}_{t=1}^T$, $\boldsymbol{w}_b^{\mathsf{Shared}} = \left\{[\rho_{n,b}^{(t),b}]_{n\in\mathcal{N}}, [\rho_{b,s}^{(t),b}]_{s\in\mathcal{S}}, [I_s^{(t),b}]_{s\in\mathcal{S}}, [\delta^{\mathsf{A},(t),b}], [\delta^{\mathsf{R},(t),b}], [R_{b,s}^{(t),b}]_{s\in\mathcal{S}}\right\}_{t=1}^T$ where $\rho_{n,b}^{(t),b}, \rho_{b,s}^{(t),b}, I_s^{(t),b}, \delta^{\mathsf{A},(t),b}, \delta^{\mathsf{R},(t),b}$,

$R_{b,s}^{(t),b}$ denote the local copies of the respective shared variables (i.e., $\rho_{n,b}^{(t)}, \rho_{b,s}^{(t)}, I_s^{(t)}, \delta^{\mathsf{A},(t)}, \delta^{\mathsf{R},(t)}, R_{b,s}^{(t)}$) at node b.

3) Each DC s: $\boldsymbol{w}_s^{\mathsf{Local}} = \big\{[z_s^{(t)}], [\gamma_s^{(t)}], [m_s^{(t)}]\big\}_{t=1}^{T}, \boldsymbol{w}_s^{\mathsf{Shared}} = \big\{[\rho_{n,b}^{(t),s}]_{n\in\mathcal{N},b\in\mathcal{B}}, [\rho_{b,s}^{(t),s}]_{b\in\mathcal{B}}, [I_s^{(t),s}]_{s\in\mathcal{S}}, [\delta^{\mathsf{A},(t),s}], [\delta^{\mathsf{R},(t),s}], [R_{b,s}^{(t),s}]_{b\in\mathcal{B}}\big\}_{t=1}^{T}$ where $\rho_{n,b}^{(t),s}, \rho_{b,s}^{(t),s}, \delta^{\mathsf{A},(t),s}, I_s^{(t),s}, \delta^{\mathsf{R},(t),s}$ denote the local copies of the respective shared variables (i.e., $\rho_{n,b}^{(t)}, \rho_{b,s}^{(t)}, \delta^{\mathsf{A},(t)}, I_s^{(t),s}, \delta^{\mathsf{R},(t)}, R_{b,s}^{(t)}$) at node s.

4) For each network element $d \in \mathcal{N}\cup\mathcal{S}\cup\mathcal{B}$, we let $\boldsymbol{w}_d$ encompass all the associated variables: $\boldsymbol{w}_d = \boldsymbol{w}_d^{\mathsf{Local}} \cup \boldsymbol{w}_d^{\mathsf{Shared}}$. Thus, with some abuse of notation we denote the extended variable space of $\boldsymbol{\mathcal{P}}$ via $\boldsymbol{w}$ defined as

$$\boldsymbol{w} = [\boldsymbol{w}_d]_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}} \triangleq \bigcup_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}} \boldsymbol{w}_d. \tag{51}$$

Additionally, equality constraints introduced to $\boldsymbol{\mathcal{P}}$ to enforce the equality of the local copies of the shared variables are

$$\rho_{n,b}^{(t),n} - \rho_{n,b}^{(t),b} = 0,\ \forall n\in\mathcal{N}, b\in\mathcal{B},\ \forall t, \tag{52}$$

$$\rho_{n,b}^{(t),n} - \rho_{n,b}^{(t),s} = 0,\ \forall n\in\mathcal{N}, s\in\mathcal{S},\ b\in\mathcal{B},\ \forall t, \tag{53}$$

$$\rho_{b,s}^{(t),b} - \rho_{b,s}^{(t),s} = 0,\ \forall b\in\mathcal{B}, s\in\mathcal{S},\ \forall t, \tag{54}$$

$$I_s^{(t),d} - I_s^{(t),d'} = 0,\ \forall d,d'\in\mathcal{N}\cup\mathcal{B}\cup\mathcal{S},\ s\in\mathcal{S},\ \forall t, \tag{55}$$

$$\delta^{\mathsf{A},(t),d} - \delta^{\mathsf{A},(t),d'} = 0,\ \forall d,d'\in\mathcal{N}\cup\mathcal{S},\ \forall t, \tag{56}$$

$$\delta^{\mathsf{R},(t),d} - \delta^{\mathsf{R},(t),d'} = 0,\ \forall d,d'\in\mathcal{B}\cup\mathcal{S},\ \forall t, \tag{57}$$

$$R_{b,s}^{(t),b} - R_{b,s}^{(t),s} = 0,\ \forall b\in\mathcal{B}, s\in\mathcal{S},\ \forall t. \tag{58}$$

Imposing (52)-(58) leads to *separability* of the optimization problem while ensuring agreement on the shared variables, allowing the development of a distributed solution later. We first highlight some characteristics of $\boldsymbol{\mathcal{P}}$, which are exploited in our distributed solution framework.

Structure of $\boldsymbol{\mathcal{P}}$. Let us denote the objective of $\boldsymbol{\mathcal{P}}$ given by (44) as $\mathcal{J}$. We note that the convex constraints of $\boldsymbol{\mathcal{P}}$ comprise linear summations in (44a)-(44e) and variable ranges in (44j)-(44p), (49)-(50), defined independently across nodes. We thus combine these convex constraints for each node $d \in \mathcal{N}\cup\mathcal{B}\cup\mathcal{S}$ as constraint vector $\boldsymbol{\mathcal{D}}_d(\boldsymbol{w}_d) \leq \mathbf{0}$ The constraints (16),(18),(20),(22),(23),(26),(27),(28),(29),(35),(40),(44f),(44g),(44h),(44i),(45)-(47) associated with $\boldsymbol{\mathcal{P}}$ are non-convex and can be jointly denoted by vector of constraints $\boldsymbol{\mathcal{C}}(\boldsymbol{w}) \leq \mathbf{0}$. Furthermore, we observe that (52)-(58) are linear equality constraints involving one or more network elements, thus can be equivalently written as an equality constraint in vector form, i.e., $\sum_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}} \boldsymbol{\mathcal{G}}(\boldsymbol{w}_d) = \mathbf{0}$. Then, the augmented version of $\boldsymbol{\mathcal{P}}$ denoted by $\widehat{\boldsymbol{\mathcal{P}}}$ which encompasses all the aforementioned constraints can be written as

$$\widehat{\boldsymbol{\mathcal{P}}}: \min_{\boldsymbol{w}} \quad \mathcal{J}(\boldsymbol{w}) \tag{59}$$

$$\text{s.t.} \quad \boldsymbol{\mathcal{C}}(\boldsymbol{w}) \leq \mathbf{0} \tag{59a}$$

$$\sum_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}} \boldsymbol{\mathcal{G}}(\boldsymbol{w}_d) = \mathbf{0} \tag{59b}$$

$$\boldsymbol{\mathcal{D}}_d(\boldsymbol{w}_d) \leq \mathbf{0},\ d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}. \tag{59c}$$

Next we describe our successive convex methodology, which is an *iterative procedure*, wherein $\widehat{\boldsymbol{\mathcal{P}}}$ is solved.

Successive Convex Solver. The psudo-code of our successive convex solver is given in Algorithm 1. The algorithm starts with an initial feasible point $\boldsymbol{w}^{(0)}$ satisfying the constraints of $\boldsymbol{\mathcal{P}}$. During each iteration ℓ of the algorithm,

Algorithm 1 Successive Convex Solver Wrapper

1: **Input:** Initialize $\boldsymbol{w}^{(0)}$, step size ζ
2: **Output:** Final iterate $\boldsymbol{w}$
3: Initialize iteration count $\ell = 0$.
4: **while** $\boldsymbol{w}^{(\ell)}$ has not converged **do**
5: Compute $\hat{\boldsymbol{w}}(\boldsymbol{w}^{(\ell)})$, the distributed parallel solution of $\widehat{\boldsymbol{\mathcal{P}}}_{\boldsymbol{w}^{(\ell)}}$ using PD CE−FL (Algorithm 2)
6: Obtain $\boldsymbol{w}^{(\ell+1)}$ using update rule (60)
7: $\ell \longleftarrow \ell + 1$
8: **end while**

we convexify $\widehat{\boldsymbol{\mathcal{P}}}$ at the given feasible point $\boldsymbol{w}^{(\ell)}$ to obtain *surrogate problem* $\widehat{\boldsymbol{\mathcal{P}}}_{\boldsymbol{w}^{(\ell)}}$ which can be further distributed and solved across the network elements $d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}$ in parallel. We denote the distributed solution of the surrogate problem $\widehat{\boldsymbol{\mathcal{P}}}_{\boldsymbol{w}^{(\ell)}}$ by $\hat{\boldsymbol{w}}(\boldsymbol{w}^{(\ell)})$. Subsequently, we update the variables as ($\psi < 1$)

$$\boldsymbol{w}^{(\ell+1)} = \boldsymbol{w}^{(\ell)} + \psi(\hat{\boldsymbol{w}}(\boldsymbol{w}^{(\ell)}) - \boldsymbol{w}^{(\ell)}). \tag{60}$$

The key aspects of Algorithm 1 are thus (i) obtaining the convex approximation, i.e., $\widehat{\boldsymbol{\mathcal{P}}}_{\boldsymbol{w}^{(\ell)}}$, and (ii) the design of the distributed solution. Henceforth, we first describe the convex approximation technique used to relax our original problem, and then build our parallel distributed solver.

Convexification of $\widehat{\boldsymbol{\mathcal{P}}}$. We use a proximal gradient method to relax the objective $\mathcal{J}(\boldsymbol{w})$. The non-convex constraints $\boldsymbol{\mathcal{C}}(\boldsymbol{w})$ are also convexified such that the approximations upper-bound the original constraints. In particular, at iteration ℓ of our successive convex solver (Algorithm 1), given the current solution of $\widehat{\boldsymbol{\mathcal{P}}}$, i.e., $\boldsymbol{w}^{(\ell)}$, we obtain convex approximations of objective $\mathcal{J}$ and non-convex constraints $\boldsymbol{\mathcal{C}}$ denoted by $\widetilde{\mathcal{J}}$ and $\widetilde{\boldsymbol{\mathcal{C}}}$, respectively, as

$$\widetilde{\mathcal{J}}(\boldsymbol{w};\boldsymbol{w}^{(\ell)}) = \sum_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}} \widetilde{\mathcal{J}}_d(\boldsymbol{w}_d;\boldsymbol{w}^{(\ell)}), \tag{61}$$

$$\begin{aligned}\widetilde{\mathcal{J}}_d(\boldsymbol{w}_d;\boldsymbol{w}^{(\ell)}) = {}& \frac{1}{|\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}|}\mathcal{J}(\boldsymbol{w}^{(\ell)}) \\ & + \nabla_{\boldsymbol{w}_d}\mathcal{J}(\boldsymbol{w}^{(\ell)})^{\top}(\boldsymbol{w}_d - \boldsymbol{w}_d^{(\ell)}) \\ & + \frac{\lambda_1}{2}\|\boldsymbol{w}_d - \boldsymbol{w}_d^{(\ell)}\|^2.\end{aligned} \tag{62}$$

$$\widetilde{\boldsymbol{\mathcal{C}}}(\boldsymbol{w};\boldsymbol{w}^{(\ell)}) = \sum_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}} \widetilde{\boldsymbol{\mathcal{C}}}_d(\boldsymbol{w}_d;\boldsymbol{w}^{(\ell)}), \tag{63}$$

$$\begin{aligned}\widetilde{\boldsymbol{\mathcal{C}}}_d(\boldsymbol{w}_d;\boldsymbol{w}^{(\ell)}) = {}& \frac{1}{|\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}|}\boldsymbol{\mathcal{C}}(\boldsymbol{w}^{(\ell)}) \\ & + \nabla_{\boldsymbol{w}_d}\boldsymbol{\mathcal{C}}(\boldsymbol{w}^{(\ell)})^{\top}(\boldsymbol{w}_d - \boldsymbol{w}_d^{(\ell)}) \\ & + \frac{L_{\nabla\boldsymbol{\mathcal{C}}}}{2}\|\boldsymbol{w}_d - \boldsymbol{w}_d^{(\ell)}\|^2.\end{aligned} \tag{64}$$

In (63)-(64), $L_{\nabla\boldsymbol{\mathcal{C}}}$ is the Lipschitz constant which is a characteristic of the constraint function $\boldsymbol{\mathcal{C}}$. The above formulation implies that $\widetilde{\boldsymbol{\mathcal{C}}}(\boldsymbol{w};\boldsymbol{w}^{(\ell)}) \geq \boldsymbol{\mathcal{C}}(\boldsymbol{w})$ [40], [41]. With $\lambda_1 > 0$, the proximal gradient-based relaxation in (61)-(62) ensures strong convexity of the *surrogate objective function* $\widetilde{\mathcal{J}}$. Thus, at each iteration ℓ of Algorithm 1, we formulate the relaxed convex approximation of $\widehat{\boldsymbol{\mathcal{P}}}$ at current iterate $\boldsymbol{w}^{(\ell)}$, i.e., $\widehat{\boldsymbol{\mathcal{P}}}_{\boldsymbol{w}^{(\ell)}}$, as

$$\widehat{\boldsymbol{\mathcal{P}}}_{\boldsymbol{w}^{(\ell)}}: \min_{\boldsymbol{w}} \quad \sum_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}} \widetilde{\mathcal{J}}_d(\boldsymbol{w}_d;\boldsymbol{w}^{(\ell)}) \tag{65}$$

$$\text{s.t.} \quad \sum_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}} \widetilde{\boldsymbol{\mathcal{C}}}_{\mathsf{d}}(\boldsymbol{w}_d;\boldsymbol{w}^{(\ell)}) \leq 0 \tag{65a}$$

$$\sum_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}} \boldsymbol{\mathcal{G}}(\boldsymbol{w}_d) = 0 \tag{65b}$$

$$\boldsymbol{\mathcal{D}}_d(\boldsymbol{w}_d) \leq 0,\ d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}. \tag{65c}$$

The convex constraint (65c) is separable across the nodes. However, (65a)-(65b) are in the form of summation over nodes, which does not let the problem to be trivially distributed. To facilitate a parallel distributed solution to $\widehat{\boldsymbol{\mathcal{P}}}_{\boldsymbol{w}^{(\ell)}}$, we perform Lagrangian-based dualization for constraints (65a)-(65b) as

$$\begin{aligned}\mathcal{L}(\boldsymbol{w},\boldsymbol{\Lambda},\boldsymbol{\Omega};\boldsymbol{w}^{(\ell)}) = &\sum_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}} \widetilde{\mathcal{J}}_d(\boldsymbol{w}_d;\boldsymbol{w}^{(\ell)})\\ &+\boldsymbol{\Lambda}^\top\big(\sum_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}} \widetilde{\boldsymbol{\mathcal{C}}}_d(\boldsymbol{w}_d;\boldsymbol{w}^{(\ell)})\big)\\ &+\boldsymbol{\Omega}^\top\big(\sum_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}} \boldsymbol{\mathcal{G}}(\boldsymbol{w}_d)\big).\end{aligned} \tag{66}$$

In (66), $\boldsymbol{w}$ and $\{\boldsymbol{\Lambda},\boldsymbol{\Omega}\}$ are the *primal* and *dual* variables, respectively. Also, $\boldsymbol{\Lambda}$ and $\boldsymbol{\Omega}$ are the Lagrangian multipliers associated with the convexified inequality constraints $\widetilde{\boldsymbol{\mathcal{C}}}_.(.,\boldsymbol{w}^{(\ell)})$ and linear equality constraints $\boldsymbol{\mathcal{G}}(.)$, respectively.

Based on the above formulation, we next define the following max-min optimization problem:

$$\begin{aligned}&\max_{\boldsymbol{\Lambda}\geq\mathbf{0}}\min_{\boldsymbol{w}}\ \mathcal{L}(\boldsymbol{w},\boldsymbol{\Lambda},\boldsymbol{\Omega};\boldsymbol{w}^{(\ell)})\\ &\text{s.t.}\ \ \boldsymbol{\mathcal{D}}_\mathrm{d}(\boldsymbol{w}_d)\leq 0,\ d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}.\end{aligned} \tag{67}$$

Note that solution to the above max-min problem is optimal for $\widehat{\boldsymbol{\mathcal{P}}}_{\boldsymbol{w}^{(\ell)}}$ due to its strongly-convex objective and convex constraints. In (67), for fixed values of dual parameters $\{\boldsymbol{\Lambda},\boldsymbol{\Omega}\}$, the inner minimization problem can be distributed across the network elements, where each node d aims to solve the partial Lagrangian minimization problem of the following format:

$$\begin{aligned}&\min_{\boldsymbol{w}_d}\widetilde{\mathcal{J}}_d(\boldsymbol{w}_d;\boldsymbol{w}^{(\ell)})+\boldsymbol{\Lambda}^\top\ \widetilde{\boldsymbol{\mathcal{C}}}_d(\boldsymbol{w}_d;\boldsymbol{w}^{(\ell)})+\boldsymbol{\Omega}^\top\boldsymbol{\mathcal{G}}(\boldsymbol{w}_d)\\ &\text{s.t.}\ \ \boldsymbol{\mathcal{D}}_d(\boldsymbol{w}_d)\leq 0.\end{aligned} \tag{68}$$

With the above max-min formulation and the division of the problem into sub-problems given by (67)-(68), in the following we construct a solution framework which enables parallel updates of the primal variables followed by a *decentralized consensus scheme* to update the dual-variables alternately.

Iterative Distributed Primal-Dual Algorithm with Decentralized Consensus. The pseudo-code of our *iterative* distributed primal-dual method is given in Algorithm 2. We first initialize the dual variables as $\{\boldsymbol{\Lambda}^{[0]},\boldsymbol{\Omega}^{[0]}\}$. Then, during each iteration i, for current estimates of dual variables $\{\boldsymbol{\Lambda}^{[i-1]},\boldsymbol{\Omega}^{[i-1]}\}$, we first highlight that the partial Lagrangian based minimization subproblem, i.e., (68), has a convex objective function. Also, the constraint $\boldsymbol{\mathcal{D}}_d(\boldsymbol{w}_d)\leq 0$ consists of box and polyhedrons constraints given by (44a)-(44p), (49)-(50), which are closed convex projection sets. Hence, we leverage gradient projection algorithm (GPA) [42] and obtain the solution of the primal variable minimization subproblems given by (68) for each individual node d at each iteration i as

$$\hat{\boldsymbol{w}}_d^{[i]}(\boldsymbol{w}^{(\ell)}) = \underset{\boldsymbol{w}_d:\boldsymbol{\mathcal{D}}_d(\boldsymbol{w}_d)\leq\mathbf{0}}{\arg\min}\ \mathcal{L}(\boldsymbol{w}_d,\boldsymbol{\Lambda}_d^{[i-1]},\boldsymbol{\Omega}_d^{[i-1]};\boldsymbol{w}^{(\ell)}). \tag{69}$$

Upon obtaining $\{\hat{\boldsymbol{w}}_d^{[i]}(\boldsymbol{w}^{(\ell)})\}_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}}$, we perform gradient ascent updates on dual variables $\{\boldsymbol{\Lambda},\boldsymbol{\Omega}\}$. These correspond to the outer maximization in (67), and thus involve computation of gradient of the Lagrangian function $\mathcal{L}(\boldsymbol{w},\boldsymbol{\Lambda},\boldsymbol{\Omega};\boldsymbol{w}^{(\ell)})$ defined in (66) at the current primal variable estimates

Algorithm 2 Iterative Distributed Primal Dual Method (PD CE−FL)

Input: Initialize $\boldsymbol{\Lambda}^{[0]}\geq\mathbf{0},\boldsymbol{\Omega}^{[0]}$
Output: Final iterates of $\{\boldsymbol{\Lambda},\boldsymbol{\Omega}\}$
Iteration count $i=0$
while $\{\boldsymbol{\Lambda}_d,\boldsymbol{\Omega}_d\}$ not converged $\forall d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}$ **do**
  for $d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}$ parallely **do**
    Obtain $\hat{\boldsymbol{w}}_d^{[i]}(\boldsymbol{w}^{(\ell)})$ via gradient projection method on (69) %% primal descent
    Obtain $\{\boldsymbol{\Lambda}_d^{[i]},\boldsymbol{\Omega}_d^{[i]}\}$ using (72)-(74) %% dual ascent
  end for
  $i\longleftarrow i+1$
end while

Algorithm 3 Iterative Decentralized Consensus Method (Consensus CE−FL)

Input: Initialization of variables $\boldsymbol{\Gamma}_d^{\{0\}}=[\boldsymbol{\Lambda}_d^{[i]},\boldsymbol{\Omega}_d^{[i]}]$ at each node d, maximum number of consensus rounds J
Output: Final local iterates $\{\boldsymbol{\Gamma}_d^{\{J\}}\}_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}}$ at convergence
Initialize iteration count iteration count $j=0$
while $j\leq J$ **do**
  for $d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}$ parallely **do**
    Receive $\{\boldsymbol{\Gamma}_{d'}^{\{j-1\}}\}_{(d,d')\in\mathcal{E}}$
    Communicate $\boldsymbol{\Gamma}_d^{\{j-1\}}$ to neighbors $\{d':(d,d')\in\mathcal{E}\}$
    Obtain $\boldsymbol{\Gamma}_d^{\{j\}}$ using consensus update rule (75)
  end for
  $j\longleftarrow j+1$
end while

$\{\hat{\boldsymbol{w}}_d^{[i]}(\boldsymbol{w}^{(\ell)})\}_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}}$ solved distributedly via (69). During each iteration i of Algorithm 2, this update is described as

$$\begin{aligned}&\boldsymbol{\Lambda}^{[i]}\\ &=\left[\boldsymbol{\Lambda}^{[i-1]}+\frac{\kappa\ \nabla_{\boldsymbol{\Lambda}}\mathcal{L}\big(\hat{\boldsymbol{w}}_d^{[i]}(\boldsymbol{w}^{(\ell)}),\boldsymbol{\Lambda}^{[i-1]},\boldsymbol{\Omega}^{[i-1]};\boldsymbol{w}^{(\ell)}\big)}{|\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}|}\right]^+\\ &=\Bigg[\boldsymbol{\Lambda}^{[i-1]}+\frac{\kappa}{|\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}|}\underbrace{\sum_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}}\widetilde{\boldsymbol{\mathcal{C}}}_d\big(\hat{\boldsymbol{w}}_d^{[i]}(\boldsymbol{w}^{(\ell)});\boldsymbol{w}^{(\ell)}\big)}_{\text{(a)}}\Bigg]^+,\end{aligned} \tag{70}$$

$$\begin{aligned}&\boldsymbol{\Omega}^{[i]}\\ &=\boldsymbol{\Omega}^{[i-1]}+\frac{\varepsilon\ \nabla_{\boldsymbol{\Omega}}\mathcal{L}(\hat{\boldsymbol{w}}_d^{[i]}(\boldsymbol{w}^{(\ell)}),\boldsymbol{\Lambda}^{[i-1]},\boldsymbol{\Omega}^{[i-1]};\boldsymbol{w}^{(\ell)})}{|\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}|}\\ &=\boldsymbol{\Omega}^{[i-1]}+\frac{\varepsilon}{|\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}|}\underbrace{\sum_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}}\boldsymbol{\mathcal{G}}\big(\hat{\boldsymbol{w}}_d^{[i]}(\boldsymbol{w}^{(\ell)})\big)}_{\text{(b)}},\end{aligned} \tag{71}$$

where κ and ε are the step sizes. (a) and (b) in (70) and (71) cannot be computed directly due to the need for a central processor; however, we desire to update $\{\boldsymbol{\Lambda},\boldsymbol{\Omega}\}$ locally to obtain a distributed solution. To this end, we conduct updates of dual variables via a *decentralized consensus scheme*, where the dual-ascent update consists of two steps. First, local copies of the dual variables get updated at each node d of the network

$$\boldsymbol{\Lambda}_d^{[i]}=\boldsymbol{\Lambda}_d^{[i-1]}+\kappa\ \tilde{\boldsymbol{\mathcal{C}}}_d(\hat{\boldsymbol{w}}_d^{[i]}(\boldsymbol{w}^{(\ell)});\boldsymbol{w}^{(\ell)}), \tag{72}$$

$$\boldsymbol{\Omega}_d^{[i]} = \boldsymbol{\Omega}_d^{[i-1]} + \varepsilon\, \boldsymbol{\mathcal{G}}_d(\hat{\boldsymbol{w}}_d^{[i]}(\boldsymbol{w}^{(\ell)})). \tag{73}$$

Then, the nodes exchange their local copies with neighboring nodes to form a consensus on the dual variables as follows:

$$[\boldsymbol{\Lambda}_d^{[i]}, \boldsymbol{\Omega}_d^{[i]}] = \text{Consensus } \texttt{CE-FL}(d, [\boldsymbol{\Lambda}_d^{[i]}, \boldsymbol{\Omega}_d^{[i]}]),$$
$$\boldsymbol{\Lambda}_d^{[i]} = \big[\boldsymbol{\Lambda}_d^{[i]}\big]^+. \tag{74}$$

The alternating optimization of primal and dual variables described by (69), (72)-(74) continues until convergence of sequence $\{\boldsymbol{\Lambda}_d, \boldsymbol{\Omega}_d\}_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}}$. The Consensus CE−FL in (74) is described in Algorithm 3, which relies on local message exchange across neighboring nodes, which we discuss next.

Decentralized Network-Wide Consensus (Algorithm 3). We consider a 0-1 edge connection between the nodes, where two nodes either engage in sharing optimization variables or do not communicate. We consider a bi-level hierarchical structure for this communication graph (see Fig. 2) wherein the UEs can possibly perform D2D communications with other UEs in their vicinity as well as uplink-downlink communications with at least one BSs (no direct communication link between DCs and UEs is assumed). We also assume that each BS is capable of communication to at least a DC, but it may or may not be communicating to other BSs.

Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ denote this communication graph with vertex set $\mathcal{V} = \mathcal{N} \cup \mathcal{B} \cup \mathcal{S}$ and edge set $\mathcal{E}$, where $(d, d') \in \mathcal{E}$ implies the communication between two nodes d, d' during the computation of solution of the network optimization. Also, let $\boldsymbol{A} = [A_{d,d'}]_{d,d'\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}}$ denote its adjacency matrix, where

$A_{d,d'} = 1,\ \forall (d,d') \in \mathcal{E}, A_{d,d'} = 0\ \forall (d,d') \notin \mathcal{E}$. Since there is no connection between the UEs and DCs, we have $A_{n,s} = 0,\ \forall s \in \mathcal{S}, n \in \mathcal{N}$. Also, since each UE is connected to at least one BS, we have $\exists b \in \mathcal{B}:\ A_{n,b} = 1,\ \forall n \in \mathcal{N}$. Finally, since each BS engages in variable transfer to at least one DC: $\exists s \in \mathcal{S}: A_{b,s} = 1,\ \forall b \in \mathcal{B}$. Also, we assume that each DC is at least connected to another DC: $\exists s' \in \mathcal{S}: A_{s,s'} = 1,\ \forall s \in \mathcal{S}$.

We next describe the consensus procedure performed over $\mathcal{H}$ to locally update the dual variables in (70)-(71). Our procedure is described in Algorithm 3. For notation simplicity, we represent the dual variables via a vector $\boldsymbol{\Gamma} = [\boldsymbol{\Lambda}, \boldsymbol{\Omega}]$. At iteration i of PD CE−FL (Algorithm 2) the call of Consensus CE−FL subroutine (Algorithm 3) via (74) is triggered with initialization $\boldsymbol{\Gamma}_d^{\{0\}} = [\boldsymbol{\Lambda}_d^{(i)}, \boldsymbol{\Omega}_d^{(i)}]$. Afterward, at each iteration j of Algorithm 3, each network node $d \in \mathcal{N} \cup \mathcal{S} \cup \mathcal{B}$ sends its current local value of dual variables, i.e., $\boldsymbol{\Gamma}_d^{\{j-1\}}$, and in turn receives the value of which from its neighbors, i.e., $\{\boldsymbol{\Gamma}_{d'}^{\{j-1\}}\}_{(d,d')\in\mathcal{E}}$. Subsequently, the following update is executed at each network node d:

$$\boldsymbol{\Gamma}_d^{\{j\}} = \boldsymbol{\Gamma}_d^{\{j-1\}} W_{d,d} + \sum_{d':(d,d')\in\mathcal{E}} \boldsymbol{\Gamma}_{d'}^{\{j-1\}} W_{d,d'}, \tag{75}$$

where $W_{d,d'}$ is the *weight* that node d assigns to its neighbor d'. It is important to construct $W_{d,d'}$ for all pairs of neighboring nodes (d, d') so that the local estimates of the dual variables asymptotically attain their global values. Letting degree(d) denote the degree of node d, we consider the consensus weights as $W_{d,d} = 1 - \chi \times \text{degree}(d)$, and $W_{d,d'} = \chi\ \forall (d,d') \in \mathcal{E}$, where χ is a constant satisfying $z < \frac{1}{\max_{d\in\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}}\ \text{degree}(d)}$, which is proven to show fast convergence for consensus [43]. With the knowledge of z (e.g., trivially chosen as $\chi = \frac{1}{|\mathcal{N}\cup\mathcal{S}\cup\mathcal{B}|} - \hat{\chi}$ for a small $\hat{\chi} > 0$), the consensus weights can be distributedly obtained. We next study the convergence of our optimization solver (proof provided in Appendix F in the supplementary material).

TABLE I
ENERGY CONSUMPTION COMPARISON ACROSS VARYING TARGET ACCURACIES

	F-MNIST (Target Acc.)			CIFAR-10 (Target Acc.)		
	60 %	**70 %**	**80 %**	**40 %**	**50 %**	**60 %**
CE−FL (in KJ)	19.5	32.8	47.3	17.3	26.6	42.7
FedNova (in KJ)	23.4	49.2	78.7	24.7	35.1	60.3
FedAvg (in KJ)	27.9	57.1	83.2	29.1	39.0	67.8
vs FedNova Savings (%)	16.7	33.3	39.9	30	24.2	29.2
vs FedAvg Savings (%)	30.1	42.6	43.1	40.5	31.8	37.0

Theorem 2 (Convergence of the Optimization Solver): If $J \to \infty$ (see Algorithm 3), the sequence $\{\boldsymbol{w}^{(\ell)}\}$ generated by Algorithm 1 is feasible for $\boldsymbol{\mathcal{P}}$ and non-increasing, which asymptotically reaches a stationary solution of $\boldsymbol{\mathcal{P}}$.

In Appendix J in the supplementary material, we provide the complexity analysis for our CE−FL methodology.

VI. NUMERICAL EVALUATIONS

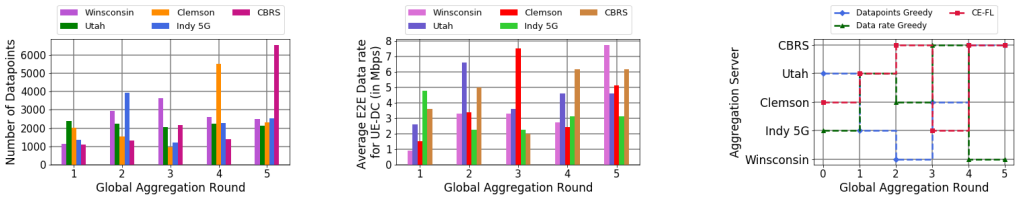
A. Simulations Setup and Testbed Configuration

We acquire realistic models of communication models and the units' power consumption through data gathering from 5G/4G and CBRS network testbeds that include BSs, UEs, and DCs. The data collection technique is explained in Appendix G in the supplementary material. The DCs for the 5G/4G data collection are located at the Indy 5G Zone [44], Discovery Park District [45], Wisconsin, Utah, and Clemson, respectively [46]. Following that, a larger default network setting was generated for the numerical evaluations by post-processing the measured data (see Appendix G–D in the supplementary material). The capacity of DCs is chosen $R_s^{\max} \in [40, 50]$Gbps, $\forall s$, and capacity of each BS-DC link is chosen as $R_{b,s}^{\max} \in [3, 4]$Gbps, $\forall b, s$. The created dataset consists of 20 UEs, 10 BSs, and 5 DCs, with each server along with 2 BSs and 4 UEs comprising a *sub-network*. Each sub-network is characterized by high intra-network and low inter-network data transfer rates.

B. Results and Discussion

We perform an *ablation study* on $\boldsymbol{\mathcal{P}}$ by isolating different sets of optimization variables to show the behavior of $\boldsymbol{\mathcal{P}}$ under different network settings. The default network setting is described in Appendix H in the supplementary material.

1) Performance of CE−FL for Dynamic ML Model Training: We compare the ML model performance of CE−FL in terms of energy consumption and delay against FedNova [33] in terms of classification accuracy obtained on Fashion-MNIST [47] and CIFAR-10 [48] datasets in Table I and II respectively (See Appendix H in the supplementary material for the description of the datasets). We consider time-varying datasets at the UEs, where at each global aggregation round, UEs acquire datasets with sizes distributed according to normal distribution with mean 2000 and variance 200. We note that ML training under FedNova was performed with average CPU/data-processing frequencies, mini-batch sizes and number of SGD iterations at the DPUs. Tables I, II demonstrate the energy and delay savings that CE−FL obtains against FedNova and AedAvg upon reaching different classification accuracies. We observe that CE−FL outperforms FedNova, which in turn beats FedAvg, in terms of network delay and energy costs.



(a): Data distribution (b): E2E data rate distribution (c): Aggregator switching

Fig. 3. Datapoint distribution, E2E data rate distribution, and CE−FL aggregation DC switching pattern against baseline methods.

TABLE II
MODEL TRAINING DELAY COMPARISON ACROSS VARYING TARGET ACCURACIES

	F-MNIST (Target Acc.)			CIFAR-10 (Target Acc.)		
	60 %	**70 %**	**80 %**	**40 %**	**50 %**	**60 %**
CE−FL (in sec)	2721.2	3521.4	4577.8	2850.7	3988.6	4912.5
FedNova (in sec)	3035.2	4359.9	5432.6	3271.5	4729.9	5873.4
FedAvg (in sec)	3721.6	4882.7	6156.3	4022.1	5623.9	6433.8
vs FedNova Savings (%)	10.3	19.2	15.7	12.9	15.7	16.4
vs FedAvg Savings (%)	26.9	27.9	25.6	29.1	29.1	23.7

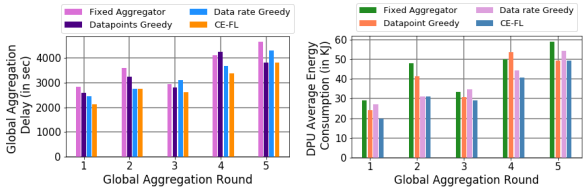
2) Floating Aggregation Point in CE−FL*:* We examine how time varying and unequal data at DPUs control the aggregation DC throughout the ML training in Fig. 3.

For the experiments, we consider two greedy baselines: (i) datapoint greedy, and (ii) data rate greedy. The datapoint greedy strategy chooses the DC whose *subnetwork* has the highest concentration of datapoints at each global aggregation as the floating aggregator. Also, data rate greedy method designs a strategy based on the end-to-end (E2E) data transfer rates between UEs and the DCs. Mathematically, we define the E2E data-rate between arbitrary UE n and DC s as

$$R_{n,s}^{\mathsf{E2E},(t)} = \max_{b\in\mathcal{B}} \left\{ \frac{1}{\frac{1}{R_{n,b}^{(t)}} + \frac{1}{R_{b,s}^{\max}}} \right\}. \tag{76}$$

Then, at each round of global aggregation, the DC with the highest average E2E data rate across all the UEs is chosen as the floating aggregator DC. We depict the evolution of datapoint concentration and average E2E data rates at the DC across global aggregation rounds in Figs. 3a and 3b, and how the choice of the aggregator varies in the greedy strategies in comparison to CE−FL in Fig. 3c. Comparing the data distribution across the network depicted in Fig. 3a and the optimal aggregator selected in CE−FL in Fig. 3c reveals an interesting phenomenon. In particular, the optimal aggregator selection of CE−FL matches that of the datapoint greedy method (i.e., selecting the DC in the area with the highest data concentration) when data concentrations are extremely skewed across the network (i.e., $t = 5$). A similar takeaway can be obtained via comparing Fig. 3b and 3c, where CE−FL only matches that of the data rate greedy strategy (at $t = 2$) when the E2E data rate is skewed toward Utah DC, which also has a high data concentration.

However, Fig. 3c reveals that CE−FL does not always favor the DCs with the highest data concentration (i.e., $t \in \{1, 2, 3, 4\}$). This is due to the fact that congestion across the links and heterogeneity of the network elements in terms of computation and proximity are further considered in active aggregation DC selection in $\boldsymbol{\mathcal{P}}$.



(a): Delay incurred (b): Energy consumption

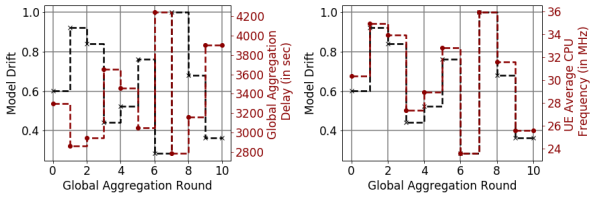
Fig. 4. Delay and energy comparison between CE−FL and the baselines.

We then investigate the energy and delay savings obtained under CE−FL active aggregator selection against fixed aggregator strategy (averaged over the 5 DCs) and the greedy method. Fig. 4a, 4b confirms the efficacy of the active selection paradigm in CE−FL in terms of network resource savings, highlighting the need to jointly taking into account the heterogeneities of network elements, congestion of the links, and uneven data concentrations as in CE−FL to select the aggregation DC.

3) Impact of Model Drift on the Behavior of CE−FL*:* Fig. 5 reveals how time varying model drift dictates the delay of conducting global aggregation rounds and the frequency of data processing at the UEs. It can be seen that increasing the model drift results in reduced global aggregation delays and faster data processing. This implies that when the temporal variations of the local datasets at the DPUs is large, to have an adequate ML model for the online datasets at the devices, CE−FL promotes rapid global aggregations and fast data processing.

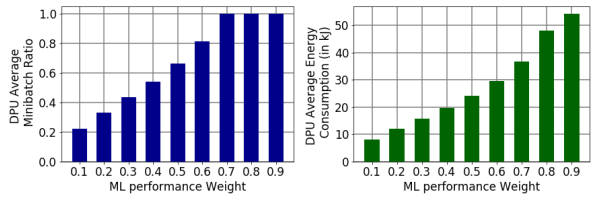
4) Impact of ML Performance Weight on Local Model Training: We characterize the impact of ML performance weight (i.e., π_1 in the objective of $\boldsymbol{\mathcal{P}}$) on the mini-batch ratios and the energy consumption at the DPUs in Fig. 6. As can be seen from the two subplots, increasing the ML performance importance results in an increase in the SGD mini-batch ratios to have more accurate local models and consequently increases the energy consumption at the DPUs. This implies that in applications where the accuracy of the ML model is of particular importance, CE−FL will sacrifice resource savings to obtain an ML model with a better accuracy.

5) Decentralized Network Optimization Solver: We first develop the centralized solver for $\boldsymbol{\mathcal{P}}$ and investigate the performance of our decentralized solver against it for a network with $|\mathcal{N}| = 20$, $|\mathcal{B}| = 10$ and $|\mathcal{S}| = 5$. The centralized solver solves $\boldsymbol{\mathcal{P}}$ via Algorithm 1 and Algorithm 2 while assuming the knowledge of all the intrinsic parameters of all the UEs, DCs and BSs used in $\boldsymbol{\mathcal{P}}$. Thus, the centralized solver performs global dual updates (70), (71) without the requirement Consensus CE−FL (Algorithm 3). The comparisons are depicted in Fig. 7. As can be seen from Fig. 7a, our distributed solver has a comparable performance to the centralized counterpart under various consensus rounds $J \in \{10, 50, 70\}$ (see Algorithm 3);



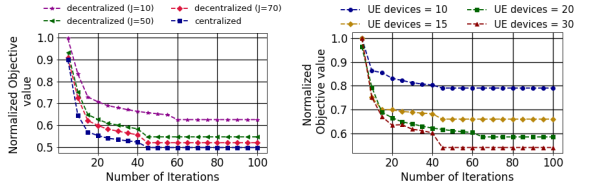
(a): Drift vs. Delay Trade-off (b): Drift vs. CPU frequency Trade-off

Fig. 5. Impact of model drift on system behavior.



(a): ML loss weight vs. average mini-batch size of SGD (b): ML loss weight vs. average energy consumption of DPUs

Fig. 6. Impact of ML loss importance on the system.



(a): Centralized vs. decentralized solution for different number of consensus rounds

(b): Decentralized algorithm convergence for different number of UEs in the system

Fig. 7. Performance of our decentralized optimization solver.

as the number of consensus rounds increases the gap between the performance of the two narrows. We next demonstrate the performance of our distributed solver under varying network sizes while keeping number of consensus rounds $J = 30$ in Fig. 7b. We vary the number of UEs as $|\mathcal{N}| \in \{10, 15, 20, 30\}$ while fixing $|\mathcal{B}| = 10$ and $|\mathcal{S}| = 5$. From Fig. 7b, it can be seen that increasing the size of the network indeed improves the solution of the solver due to processing larger number of datapoints across the DPUs leading to a better ML performances. However, 7b also highlights the shrinking gains of increasing the number of UE devices, where an initial increase in $|\mathcal{N}| = 10$ to $|\mathcal{N}| = 15$ results in significant performance enhancement; whereas, further increase in the number of UEs doesn't translate to notable improvements. This indicates that as the number of DPUs are increased beyond a certain limit, the ML performance gains are obscured by the larger cumulative network energy consumption and network delays attributed to data offloading, ML processing and aggregations.

VII. Conclusion and Future Work

We proposed CE−FL, which presumes a scenario in which the data processing for ML model training occurs simultaneously across the DCs and the UEs, which is enabled via offloading a part of the local datasets of the UEs to the DCs through the BSs. CE−FL further assumes a realistic scenario in which the number of datapoints and the data distribution across the UEs varies over time and incorporates the concept of floating aggregation DC to the distributed ML model training. We analytically characterized the ML performance of CE−FL and formulated network-aware CE−FL as an optimization problem, which will lead to a joint load balancing across the UEs and DCs and efficient data routing across the network hierarchies. We then developed a distributed optimization solver to solve our formulated problem. For the future work, studying the performance of CE−FL under device dropouts, link failures, and asynchronous model transfers can be considered. Also, we have done some experimental works in [49] to quantify the effect of varying aggregator in a decentralized setting under proof-of-work-based metrics, the extension of which to concretized formulations is open.

Acknowledgment

The authors would like to thank Hyoyoung Lim for her help in data collection from 5G testbed.

References

[1] R. Miller, "Autonomous cars could drive a deluge of data center demand," Data Center Frontier, 2017.

[2] S. Ali et al., "6G white paper on machine learning in wireless communication networks," 2020, *arXiv:2004.13875*.

[3] Z. Yang, M. Chen, K.-K. Wong, H. V. Poor, and S. Cui, "Federated learning for 6G: Applications, challenges, and opportunities," 2021, *arXiv:2101.01338*.

[4] J. Konečnỳ, H. McMahan, F. Yu, P. Richtárik, A. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Proc. NIPS WKSHP Private Multi-Party Mach. Learn.*, 2016, pp. 1–10.

[5] R. Kaewpuang, D. Niyato, P. Wang, and E. Hossain, "A framework for cooperative resource management in mobile cloud computing," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 2685–2700, Dec. 2013.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. FL, USA: PMLR, 2017, pp. 1273–1282.

[7] Z. Ji, L. Chen, N. Zhao, Y. Chen, G. Wei, and F. R. Yu, "Computation offloading for edge-assisted federated learning," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9330–9344, Sep. 2021.

[8] S. S. Azam, T. Kim, S. Hosseinalipour, C. Joe-Wong, S. Bagchi, and C. Brinton, "Can we generalize and distribute private representation learning?" in *Proc. Int. Conf. Artif. Intell. Stat.*, 2022, pp. 11320–11340.

[9] R. Arora, A. Parashar, and C. C. I. Transforming, "Secure user data in cloud computing using encryption algorithms," *Int. J. Eng. Res. Appl.*, vol. 3, no. 4, pp. 1922–1926, 2013.

[10] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–10.

[11] F. Haddadpour and M. Mahdavi, "On the convergence of local descent methods in federated learning," 2019, *arXiv:1910.14425*.

[12] S. Wang et al., "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Mar. 2019.

[13] 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.

[14] F. Ang, L. Chen, N. Zhao, Y. Chen, W. Wang, and F. R. Yu, "Robust federated learning with noisy communication," *IEEE Trans. Commun.*, vol. 68, no. 6, pp. 3452–3464, Jun. 2020.

[15] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2022–2035, Mar. 2020.

[16] S. Wang, M. Lee, S. Hosseinalipour, R. Morabito, M. Chiang, and C. G. Brinton, "Device sampling for heterogeneous federated learning: Theory, algorithms, and implementation," in *Proc. IEEE Conf. Comput. Commun.*, May 2021, pp. 1–10.

[17] B. Brik, A. Ksentini, and M. Bouaziz, "Federated learning for UAVs-enabled wireless networks: Use cases, challenges, and open problems," *IEEE Access*, vol. 8, pp. 53841–53849, 2020.

[18] S. Wang, S. Hosseinalipour, M. Gorlatova, C. G. Brinton, and M. Chiang, "UAV-assisted online machine learning over multi-tiered networks: A hierarchical nested personalized federated learning approach," *IEEE Trans. Netw. Service Manage.*, early access, Oct. 21, 2022.

[19] T. T. Vu, D. T. Ngo, N. H. Tran, H. Q. Ngo, M. N. Dao, and R. H. Middleton, "Cell-free massive MIMO for wireless federated learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6377–6392, Oct. 2020.
[20] 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.
[21] 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.
[22] F. P.-C. Lin, S. Hosseinalipour, S. S. Azam, C. G. Brinton, and N. Michelusi, "Semi-decentralized federated learning with cooperative D2D local model aggregations," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3851–3869, Dec. 2021.
[23] S. Hosseinalipour et al., "Multi-stage hybrid federated learning over large-scale D2D-enabled fog networks," *IEEE/ACM Trans. Netw.*, vol. 30, no. 4, pp. 1569–1584, Aug. 2022.
[24] M. N. H. Nguyen et al., "Self-organizing democratized learning: Towards large-scale distributed learning systems," 2020, *arXiv:2007.03278*.
[25] S. Hosseinalipour et al., "Parallel successive learning for dynamic distributed model training over heterogeneous wireless networks," 2022, *arXiv:2202.02947*.
[26] M. S. H. Abad, E. Ozfatura, D. GUndUz, and O. Ercetin, "Hierarchical federated learning ACROSS heterogeneous cellular networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 8866–8870.
[27] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.
[28] J. Wang, S. Wang, R.-R. Chen, and M. Ji, "Demystifying why local aggregation helps: Convergence analysis of hierarchical SGD," 2020, *arXiv:2010.12998*.
[29] J. Konečný, 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*.
[30] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*.
[31] E. Rizk, S. Vlaski, and A. H. Sayed, "Dynamic federated learning," in *Proc. IEEE 21st Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, May 2020, pp. 1–10.
[32] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, 2020, pp. 429–450.
[33] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 33, Dec. 2020, pp. 7611–7623.
[34] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 3557–3568.
[35] X. Cao, G. Sun, H. Yu, and M. Guizani, "PerFED-GAN: Personalized federated learning via generative adversarial networks," *IEEE Internet Things J.*, vol. 10, no. 5, pp. 3749–3762, Mar. 2023.
[36] J. Zhang, J. Chen, D. Wu, B. Chen, and S. Yu, "Poisoning attack in federated learning using generative adversarial nets," in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./13th IEEE Int. Conf. Big Data Sci. Eng.*, Aug. 2019, pp. 374–380.
[37] T. Chen, A. G. Marques, and G. B. Giannakis, "DGLB: Distributed stochastic geographical load balancing over cloud networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 7, pp. 1866–1880, Jul. 2017.
[38] S. Hosseinalipour, A. Nayak, and H. Dai, "Power-aware allocation of graph jobs in geo-distributed cloud networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 4, pp. 749–765, Apr. 2020.
[39] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, "Exploring generalization in deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–10.
[40] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.
[41] G. Scutari, F. Facchinei, and L. Lampariello, "Parallel and distributed methods for constrained nonconvex optimization—Part I: Theory," *IEEE Trans. Signal Process.*, vol. 65, no. 8, pp. 1929–1944, Apr. 2017.
[42] M. Su and H. Xu, "Remarks on the gradient-projection algorithm," *J. Nonlinear Anal. Optim.*, vol. 1, pp. 35–43, Jan. 2010.
[43] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. Control Lett.*, vol. 53, no. 1, pp. 65–78, 2004.
[44] NineTwelve. (2022). *Indy 5G Zone*. [Online]. Available: https://indiana5gzone.com
[45] Discovery Park District. (2022). *The Convergence Center for Innovation and Collaboration*. [Online]. Available: https://discoveryparkdistrict.com/the-convergence-center/
[46] D. Duplyakin et al., "The design and operation of CloudLab," in *Proc. USENIX ATC*, 2019, pp. 1–14.
[47] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.
[48] A. Krizhevsky, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2009.
[49] H. T. Nguyen, R. Morabito, K. T. Kim, and M. Chiang, "On-the-fly resource-aware model aggregation for federated learning in heterogeneous edge," in *Proc. IEEE Global Common. Conf. (GLOBECOM)*, Spain, 2021, pp. 1–6.
[50] A. O. Al-Abbasi, V. Aggarwal, and M.-R. Ra, "Multi-tier caching analysis in CDN-based over-the-top video streaming systems," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 835–847, Apr. 2019.
[51] B. Johansson, T. Keviczky, M. Johansson, and K. H. Johansson, "Subgradient methods and consensus algorithms for solving convex optimization problems," in *Proc. 47th IEEE Conf. Decis. Control*, Apr. 2008, pp. 4185–4190.
[52] D. Bienstock, G. Muñoz, and S. Pokutta, "Principled deep neural network training through linear programming," 2018, *arXiv:1810.03218*.
[53] X. Mao, A. Maaref, and K. H. Teo, "Adaptive soft frequency reuse for inter-cell interference coordination in SC-FDMA based 3GPP LTE uplinks," in *Proc. IEEE Global Telecommun. Conf.*, Jul. 2008, pp. 1–6.
[54] M. Qian, W. Hardjawana, Y. Li, B. Vucetic, X. Yang, and J. Shi, "Adaptive soft frequency reuse scheme for wireless cellular networks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 1, pp. 118–131, Jan. 2015.
[55] M. Qian, W. Hardjawana, Y. Li, B. Vucetic, J. Shi, and X. Yang, "Inter-cell interference coordination through adaptive soft frequency reuse in LTE networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2012, pp. 1618–1623.
[56] B. Gao, J.-M. J. Park, and Y. Yang, "Uplink soft frequency reuse for self-coexistence of cognitive radio networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 6, pp. 1366–1378, Jun. 2014.

Bhargav Ganguly received the dual B.Tech. and M.Tech. degrees in EE from IIT Kanpur in 2019. He is currently pursuing the Ph.D. degree with Purdue University.

Seyyedali Hosseinalipour (Member, IEEE) received the Ph.D. degree in EE from NCSU in 2020. He is currently an Assistant Professor with University at Buffalo (SUNY).

Kwang Taik Kim (Senior Member, IEEE) received the Ph.D. degree in ECE from Cornell University in 2008. He is currently a Research Assistant Professor with Purdue University.

Christopher G. Brinton (Senior Member, IEEE) received the Ph.D. degree in EE from Princeton University in 2016. He is currently an Assistant Professor of ECE with Purdue University.

Vaneet Aggarwal (Senior Member, IEEE) received the Ph.D. degree in EE from Princeton University in 2010. He is currently a Professor with Purdue University.

David J. Love (Fellow, IEEE) received the Ph.D. degree in EE from The University of Texas at Austin in 2004. He is currently a Nick Trbovich Professor of ECE with Purdue University.

Mung Chiang (Fellow, IEEE) received the Ph.D. degree from Stanford University in 2003. He is currently the President of Purdue University and a Roscoe H. George Distinguished Professor of ECE.