A Hybrid Architecture for Federated and Centralized Learning

Ahmet M. Elbir, *Senior Member IEEE*, Sinem Coleri, *Fellow IEEE*, Anastasios K. Papazafeiropoulos, *Senior Member IEEE*, Pandelis Kourtessis and Symeon Chatzinotas, *Senior Member IEEE*

***Abstract*—Many of the machine learning tasks rely on centralized learning (CL), which requires the transmission of local datasets from the clients to a parameter server (PS) entailing huge communication overhead. To overcome this, federated learning (FL) has been suggested as a promising tool, wherein the clients send only the model updates to the PS instead of the whole dataset. However, FL demands powerful computational resources from the clients. In practice, not all the clients have sufficient computational resources to participate in training. To address this common scenario, we propose a more efficient approach called hybrid federated and centralized learning (HFCL), wherein only the clients with sufficient resources employ FL, while the remaining ones send their datasets to the PS, which computes the model on behalf of them. Then, the model parameters are aggregated at the PS. To improve the efficiency of dataset transmission, we propose two different techniques: i) increased computation-per-client and ii) sequential data transmission. Notably, the HFCL frameworks outperform FL with up to 20% improvement in the learning accuracy when only half of the clients perform FL while having 50% less communication overhead than CL since all the clients collaborate on the learning process with their datasets.**

***Index Terms*—Machine learning, federated learning, centralized learning, edge intelligence, edge efficiency.**

I. Introduction

The ever growing increase in the number of connected devices in the last few years has led to a surge in the amount of data generated by mobile phones, connected vehicles, drones and internet of things (IoT) devices due to the rapid development of various emerging applications, such as artificial intelligence (AI), virtual and augmented reality (VAR), autonomous vehicles (AVs), and machine-to-machine communications [2, 3]. According to international telecommunication union (ITU), the global mobile traffic is expected to reach 607 EB by 2025. Moreover, AVs are expected to generate approximately 20 TB/day/vehicle [4]. In order to process and extract useful information from the huge amount of data, machine learning (ML) has been recognized as a promising tool for emerging technologies, such as IoT [5], AV [6], and next generation wireless communications [7–9] due to its success in image/speech recognition, natural language processing, etc [2]. These applications require huge amount of data to be processed and learned by a learning model, often an artificial neural network (ANN), by extracting the features from the raw data and providing a "meaning" to the input via constructing a model-free data mapping with huge number of learnable parameters [7]. The implementation of these learning models demands powerful computation resources, such as graphics processing units (GPUs). Therefore, huge learning models, massive amount of training data, and powerful computation infrastructure are the main driving factors defining the success of ML algorithms [3, 7].

Many of the ML tasks are based on centralized learning (CL) algorithms, which train powerful ANNs at a parameter server (PS) [2, 10, 11]. While CL inherently assumes the availability of datasets at the PS, this may not be possible for the wireless edge devices (clients), such as mobile phones, connected vehicles, and IoT devices, which need to send their datasets to the PS. Transmitting the collected datasets to the PS in a reliable manner may be too costly in terms of energy, latency, and bandwidth [12]. For example, in LTE (long term evolution), a single frame of 5 MHz bandwidth and 10 ms duration can carry only 6000 complex symbols [13], whereas the size of the whole dataset can be on the order of hundreds of thousands symbols [14]. As a result, CL-based techniques require huge bandwidth and communication overhead during training.

In order to provide a practically viable alternative to CL-based training, federated learning (FL) has been proposed to exploit the processing capability of the edge devices and the local datasets of the clients [11, 15, 16]. In FL, the clients compute and transmit the model parameters to the PS instead of their local datasets as in CL to collaboratively train the learning model. The collected model updates are aggregated at the PS and then broadcast to the clients to further update the learning parameters iteratively. Since FL does not access the whole dataset at once, it has slightly lower learning performance than that of CL. However, FL is communication-efficient and privacy-preserving since it keeps the datasets at the clients. Recently, FL has been applied to image classification [12, 15, 17], speech recognition [18] and wireless communications [19, 20]. In particular, various wireless network architectures exploiting FL have been investigated, such as cellular networks [13, 21, 22], vehicular net-

This work was supported in part by the ERC project AGNOSTIC, and by the Scientific and Technological Research Council of Turkey with European CHIST-ERA grant 119E350.

A preliminary work of this paper was presented in 2021 European Signal Processing Conference (EUSIPCO) [1].

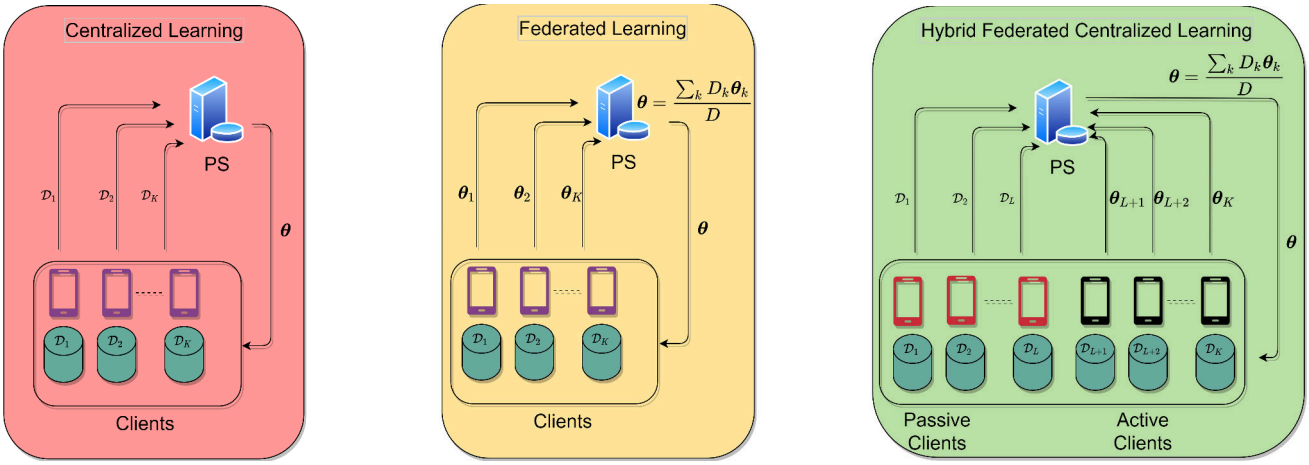
A. M. Elbir is with Duzce University, Duzce, Turkey, and with SnT, University of Luxembourg, Luxembourg (e-mail: ahmetmelbir@gmail.com).

S. Coleri is with the Department of Electrical and Electronics Engineering, Koc University, Istanbul, Turkey (e-mail: scoleri@ku.edu.tr).

A. K. Papazafeiropoulos is with the CIS Research Group, University of Hertfordshire, Hatfield, U. K. and with SnT at the University of Luxembourg, Luxembourg. (e-mail: tapapazaf@gmail.com).

P. Kourtessis is with the CIS Research Group, University of Hertfordshire, Hatfield, U. K. E-mail: p.kourtessis@herts.ac.uk

S. Chatzinotas is with the SnT at the University of Luxembourg, Luxembourg. (e-mail:symeon.chatzinotas@uni.lu).



(a) (b) (c)

Fig. 1. CL, FL and HFCL frameworks. (a) In CL, all clients transmit their datasets to the PS. (b) In FL, the datasets are preserved at the clients while model parameters are sent to the PS. (c) In HFCL, the clients are designated as active and inactive depending on their computational capability to either perform CL or FL.

works [6], unmanned aerial vehicles [23], IoT networks [24], and for the design of physical layer applications [25]. In these works, the FL architectures rely on the fact that all of the clients are capable of model computation, which may require powerful parallel processing units, such as GPUs. However, this may not always be possible in practice due to the diversity of the devices with different computation capabilities, such as mobile phones, vehicular components and IoT devices. When the edge devices lack of sufficient computational power, they cannot perform model computation, and thus become unable to participate in the learning process. To address this problem, client selection algorithms have been developed. In [26] and [27], trusted clients and the ones with sufficient computational resources are participated in FL-based training. However, these studies do not allow the clients that are not selected to participate in the training process, regardless of their computational capability.

In this work, we introduce a hybrid FL and CL (HFCL) framework for which the main motivation is to effectively train a learning model regardless of the computational capability of the clients. By exploiting their computational resources, the clients that are capable of model computation perform FL while the remaining clients resort to CL and send their datasets to the PS, which computes the corresponding model parameters on behalf of them (see Fig. 1). At the beginning of the training, the clients are designated as *inactive* (i.e., CL is applied) or *active* (i.e., FL is employed) depending on their computational capabilities. That is, the clients are called active if they have adequate computational resources, e.g., GPUs, to compute the model parameters, whereas the clients are called inactive otherwise in such a case, hence they send their datasets to the PS, which computes the model parameters on behalf of them. The PS, then, optimizes the bandwidth allocation by minimizing the maximum communication delay of the clients. During model training, both active and inactive clients are synchronized and the training process is conducted iteratively as in FL, where the model parameters are aggregated in each communication round between the PS and the active clients. Prior to the model training, the inactive clients transmit the datasets to the PS so that the PS can compute model updates on behalf of them during training. In the meantime, the active clients perform model computation. Once the model parameters corresponding to active (computed on device) and inactive (computed on PS) clients are collected at the PS, they are aggregated and broadcast to the active clients only so that they can compute the model parameters in the next iteration. Since the PS computes the model parameters on behalf of the inactive clients, it is not necessary to broadcast model updates to them. As a result, the proposed HFCL approach provides a trade-off among learning performance, communication overhead, privacy, and the clients' computational resources (see, e.g., Table I). While the usage of CL enables higher learning performance and lower dependence on the clients' computational resources, FL enjoys enhanced privacy and less communication overhead. In contrast, HFCL jointly employs CL and FL. Therefore, it enjoys less communication overhead than CL; higher performance and more flexibility than FL, while it is less privacy-preserving and lower communication-efficient than FL. Thus, HFCL can be particularly useful when the dataset does not include privacy-sensitive content, e.g., physical layer data [25].

Although the integration of FL and CL may seem straightforward, there are two main challenges in HFCL. First, training time of HFCL is longer than FL since the inactive clients need to transmit their datasets to the PS prior to the training. In other words, the active clients should wait for the inactive clients at the beginning of the training until the dataset transmission for inactive clients is completed so that they can synchronously participate in model training. To circumvent this problem and provide efficient model training, we propose sequential dataset transmission (HFCL-SDT) where the inactive clients' datasets are divided into smaller blocks to complete the dataset transmission quicker. Second, the learning accuracy of HFCL is lower than CL since the

model parameters corresponding to the active clients are noisy due to wireless transmission. To improve the learning accuracy, we propose increased computation-per-client (HFCL-ICpC) approach where the training continues at the active clients during the transmission of inactive clients' datasets instead of waiting idle for the inactive clients as in basic HFCL, in which the active clients wait for the completion of dataset transmission. Thus, HFCL-ICpC improves the learning performance as compared to HFCL without increasing the delay due to dataset generation. In summary, HFCL-ICpC improves the learning performance while HFCL-SDT reduces the dataset transmission time per communication round. Inspired from [15], where the model parameters are infrequently aggregated at the clients, we develop a new approach to first aggregate the model parameters at the clients until the transmission of the inactive clients' datasets is completed, then continue aggregating the models at the PS. Compared to [15], our technique achieves a better learning performance as well as flexibility on the hardware requirements since the learning model can be trained on the whole dataset even if a part of the clients do not have computational resources. Compared to our preliminary work [1], this paper introduces HFCL-ICpC approach to improve learning accuracy, theoretical analysis on the convergence of HFCL, and the implementation of the 3-D object detection scenario. While the term *hybrid* is used in some recent FL-based works in terms of client scheduling [28], data partitioning [29] and data security [30], this is the first work employing a hybrid architecture of FL and CL by exploiting the hardware capability of the edge devices. The main contributions of this paper are as follows:

- We propose a hybrid FL/CL approach to manage the diverse computational capabilities of the edge clients. In the proposed approach, only the clients, which have adequate computational resources perform FL while the remaining clients perform CL.
- We show that the performance of HFCL converges to that of FL (CL) when the number of active clients increases (decreases). Specifically when the active and inactive clients are partitioned equally, the proposed HFCL approach exhibits up to 20% improvement in the learning accuracy compared to FL. HFCL also enjoys 50% less communication overhead with only 2% loss in the learning accuracy as compared to CL.
- To efficiently transmit the datasets of the inactive clients to the PS, we propose the HFCL-SDT and HFCL-ICpC techniques to achieve no interruption of the learning process. Compared to FL, the proposed techniques are effective due to faster convergence rates and higher learning accuracy together with less communication delay due to less dataset transmission.

Notation: Throughout the paper, the identity matrix of size $N \times N$ is denoted by $\mathbf{I}_N$. The notation $(\cdot)^\mathsf{T}$ denotes the transpose operation. The notations $[\mathbf{A}]_{i,j}$ and $[\mathbf{a}]_i$ denote the (i,j)th element of matrix $\mathbf{A}$ and the ith element of vector $\mathbf{a}$, respectively. The function $\mathbb{E}\{\cdot\}$ expresses the statistical expectation of its argument while $\|\mathbf{a}\|$ denotes the l_2-norm and ∇ represents the gradient of vector quantity. A convolutional layer with N $D\times D$ 2-D kernels is represented by $N@$ $D\times D$.

TABLE I
COMPARISON OF CL, FL AND HFCL

Framework / Property	CL	FL	HFCL
Communication Overhead	High	Low	Moderate
Learning Accuracy	High	Moderate	Moderate
Clients' Hardware Requirement	Low	High	Flexible
Privacy-Preserving	Low	High	Moderate

TABLE II
SUMMARY OF MAIN NOTATIONS

Notation	Definition
$\mathcal{D}$	The whole dataset
$\mathcal{X}$	Input data
$\mathcal{Y}$	Output data
$\boldsymbol{\theta}$	Model parameters
$\mathcal{F}(\boldsymbol{\theta})$	Loss function (2), (3)
$\mathbf{g}(\boldsymbol{\theta})$	Gradient vector (5)
$\bar{\mathcal{L}}$	Set of inactive clients
$\mathcal{L}$	Set of active clients
$\tilde{\boldsymbol{\theta}}$	Noisy model parameters (9)
$\tilde{\boldsymbol{\theta}}_k$	Noisy model received at the kth active client (11)
$\mathcal{F}_k(\boldsymbol{\theta})$	Regularized loss function for active clients (12)
$\bar{\mathcal{F}}_k(\boldsymbol{\theta})$	Regularized loss function for inactive clients (14)
η	Learning rate
Q	Data block size during dataset transmission
N	Number of local model updates
d_k	Number of transmitted symbols
$\tilde{\sigma}^2$	Noise variance of $\widetilde{\Delta\boldsymbol{\theta}}$ (10)
σ_k^2	Noise variance of $\widetilde{\Delta\boldsymbol{\theta}}_k$ (11)

The rest of the paper is organized as follows. In Section II, CL and FL frameworks are presented. Then, we introduce the proposed HFCL, HFCL-ICpC, and HFCL-SDT approach in Section III, IV, and V. In Section VI, we investigate the convergence and the communication overhead of the HFCL approach. Section VII presents the numerical simulation results, and we summarize the paper in Section VIII with concluding remarks.

II. PRELIMINARIES: CENTRALIZED AND FEDERATED LEARNING

In ML, we are interested in constructing a learning model that forms a non-linear relationship between the two data pairs: the input and the label. Let $\mathcal{D}^{(i)} = (\mathcal{X}^{(i)}, \mathcal{Y}^{(i)})$ be the ith tuple of the dataset $\mathcal{D}$ for $i = 1, 2, \ldots, D$, where $D = |\mathcal{D}|$ denotes the number of data instances in $\mathcal{D}$. Here, $\mathcal{X}^{(i)} \in \mathbb{R}^{U_x \times V_x}$ and $\mathcal{Y}^{(i)} \in \mathbb{R}^{U_y \times V_y}$ denote the ith input and label pairs of $\mathcal{D}$, respectively. Eventually, the non-linear function, representing the ML model, can be given by $f(\mathcal{X}^{(i)}|\boldsymbol{\theta}) = \mathcal{Y}^{(i)}$, for which $\boldsymbol{\theta} \in \mathbb{R}^P$ denotes the vector of learnable model parameters. The training of the ML model takes place by focusing on a scenario, wherein K clients collaborate on optimizing $\boldsymbol{\theta}$ for the ML task.

A. Centralized Learning

In CL, the model has access to the whole dataset $\mathcal{D}$, which is collected by the PS from the clients (see Fig. 1a). Let $\mathcal{D}_k$ be the

local dataset of the kth client with D_k being its size such that $\mathcal{D} = \bigcup_{k\in\mathcal{K}} \mathcal{D}_k$ and $D = \sum_{k\in\mathcal{K}} D_k$, where $\mathcal{K} = \{1, \ldots, K\}$. In that event, the CL-based model training can be performed at the PS by solving the following optimization problem over the learnable parameters $\boldsymbol{\theta}$ as

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \mathcal{F}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \frac{D_k}{D} \mathcal{F}_k(\boldsymbol{\theta}),$$
$$\text{subject to: } f(\mathcal{X}_k^{(i)}|\boldsymbol{\theta}) = \mathcal{Y}_k^{(i)}, \quad i = 1, \ldots, D_k, \tag{1}$$

where $\boldsymbol{\theta}^\star$ denotes the model parameters after training. $\mathcal{X}_k^{(i)}$ and $\mathcal{Y}_k^{(i)}$ denote respectively the input and output of the ith element of $\mathcal{D}_k$ as $\mathcal{D}_k^{(i)} = (\mathcal{X}_k^{(i)}, \mathcal{Y}_k^{(i)})$. $\mathcal{F}_k(\boldsymbol{\theta})$ is the loss function, and it can be defined for regression and classification tasks, respectively, as

$$\mathcal{F}_k(\boldsymbol{\theta}) = \frac{1}{D_k} \sum_{i=1}^{D_k} \|f(\mathcal{X}_k^{(i)}|\boldsymbol{\theta}) - \mathcal{Y}_k^{(i)}\|^2, \tag{2}$$

and

$$\mathcal{F}_k(\boldsymbol{\theta}) = -\frac{1}{D_k} \sum_{i=1}^{D_k} [\mathcal{Y}_k^{(i)} \ln f(\mathcal{X}_k^{(i)}|\boldsymbol{\theta}) + (1 - \mathcal{Y}_k^{(i)}) \ln(1 - f(\mathcal{X}_k^{(i)}|\boldsymbol{\theta}))]. \tag{3}$$

B. Federated Learning

Compared to CL, FL does not involve the transmission of datasets to the PS. Instead, the model training is performed at the clients while the model parameters produced by the clients are aggregated at the PS, as shown in Fig. 1b. Consequently, the solution of the optimization problem in FL settings takes place at the client and the PS side as follows

$$\text{Client}: \quad \boldsymbol{\theta}_k = \arg\min_{\boldsymbol{\theta}} \mathcal{F}_k(\boldsymbol{\theta}), \quad k \in \mathcal{K}, \tag{4a}$$
$$\text{PS}: \quad \boldsymbol{\theta} = \frac{\sum_{k\in\mathcal{K}} D_k \boldsymbol{\theta}_k}{D}, \tag{4b}$$

in which, each client optimizes its model parameters $\boldsymbol{\theta}_k$ based on $\mathcal{F}_k(\boldsymbol{\theta})$ as in (4a) and transmits $\boldsymbol{\theta}_k$ to the PS, where the the model parameters are aggregated as in (4b). For an effective solution of (4a), gradient descent (GD) algorithm is used iteratively such that an optimal local solution is obtained for each iteration. The PS and the clients exchange the updated model parameters until convergence [11, 15, 31]. In particular, the parameter update at the tth iteration is performed as

$$\boldsymbol{\theta}_k^{(t+1)} = \boldsymbol{\theta}_k^{(t)} - \eta \mathbf{g}(\boldsymbol{\theta}_k^{(t)}), \tag{5}$$

where $\boldsymbol{\theta}_k^{(t)}$ denotes the model parameters at the kth client for the tth communication round/iteration with $t = 1, \ldots, T$, T is the total number of iterations, η is the learning rate, and $\mathbf{g}(\boldsymbol{\theta}_k^{(t)}) = \nabla\mathcal{F}_k(\boldsymbol{\theta}_k^{(t)})$ denotes the $P \times 1$ gradient vector, computed at the kth client, based on $\mathcal{D}_k$ and $\boldsymbol{\theta}_k^{(t)}$. Consequently, we can rewrite (4) as

$$\text{Client}: \quad \boldsymbol{\theta}_k^{(t+1)} = \boldsymbol{\theta}_k^{(t)} - \eta \mathbf{g}(\boldsymbol{\theta}_k^{(t)}), \quad k \in \mathcal{K}, \tag{6a}$$
$$\text{PS}: \quad \boldsymbol{\theta}^{(t+1)} = \frac{\sum_{k\in\mathcal{K}} D_k \boldsymbol{\theta}_k^{(t+1)}}{D}, \tag{6b}$$

which performs GD and iteratively reaches convergence.

III. Hybrid Federated and Centralized Learning

In this section, we introduce the proposed HFCL framework by taking into account the computational capability of the clients so that all clients can contribute to the learning task with their datasets regardless of their ability to compute the model parameters. The motivation is based on the following inadequacy observed in practice. In ML tasks, training a model requires huge computational power to compute the model parameters. This requirement cannot always be satisfied by the computational capabilities of the client devices. For this reason, an effective training of the ML model, we propose a hybrid training framework that accounts for the computational capabilities of the clients. Specifically, we assume that only a portion of the clients with sufficient computational power performs FL, while the remaining clients, which suffer from inadequate computational capability, send their datasets to the PS for model computation, as illustrated in Fig. 1c.

Let us define the set of clients who perform CL, i.e., sending datasets to the PS, and the ones who perform FL, i.e., transmitting the model parameters to the PS as $\mathcal{L} = \{1, \ldots, L\}$ and $\bar{\mathcal{L}} = \{L+1, \ldots, K\}$, respectively, where $\mathcal{K} = \mathcal{L} \bigcup \bar{\mathcal{L}}$ and $\mathcal{L} \cap \bar{\mathcal{L}} = \emptyset$. Furthermore, we refer to the clients in $\mathcal{L}$ and $\bar{\mathcal{L}}$ as *inactive* and *active* clients, respectively. The determination of client's condition (i.e., active/inactive) can be done by the devices sending information on their computational resources to the PS [26, 27]. By exploiting the availability of computational resources of the clients, the HFCL problem can be formulated as optimization of model parameters at the client and PS as follows

$$\text{Client}: \quad \boldsymbol{\theta}_k = \arg\min_{\boldsymbol{\theta}} \mathcal{F}_k(\boldsymbol{\theta}), \quad k \in \bar{\mathcal{L}}, \tag{7a}$$
$$\text{PS}: \quad \boldsymbol{\theta}_k = \arg\min_{\boldsymbol{\theta}} \mathcal{F}_k(\boldsymbol{\theta}), \quad k \in \mathcal{L}, \tag{7b}$$
$$\text{PS}: \quad \boldsymbol{\theta} = \frac{\sum_{k\in\mathcal{K}} D_k \boldsymbol{\theta}_k}{D}, \tag{7c}$$

where the PS actively participates in the model computation process and computes $\boldsymbol{\theta}_k$ for $k \in \mathcal{L}$ since the inactive clients do not have sufficient computational capabilities. Once the model parameters are computed, the PS aggregates them as in (7c).

A. Noisy Learning Model

In HFCL, both the model parameters and the datasets are corrupted by noise due to wireless links [20, 31]. Compared to the noise in the datasets (for image classification, see Section VII), the effect of noisy model parameters is significant since it directly corrupts the learning model [1]. During model transmission for $k \in \bar{\mathcal{L}}$, the noisy model parameters at the PS and the clients are respectively given by

$$\text{Client}: \quad \tilde{\boldsymbol{\theta}}_k = \tilde{\boldsymbol{\theta}} + \Delta\tilde{\boldsymbol{\theta}}_k, k \in \bar{\mathcal{L}}, \tag{8a}$$
$$\text{PS}: \quad \tilde{\boldsymbol{\theta}} = \frac{1}{D}\Big\{ \sum_{k\in\mathcal{L}} D_k \boldsymbol{\theta}_k + \sum_{k\in\bar{\mathcal{L}}} D_k(\boldsymbol{\theta}_k + \Delta\boldsymbol{\theta}_k) \Big\}. \tag{8b}$$

In (8b), $\boldsymbol{\theta}_{k\in\bar{\mathcal{L}}}$ denotes the true model parameter transmitted from the active clients, while $\Delta\boldsymbol{\theta}_{k\in\bar{\mathcal{L}}}$ represents the noise term added onto $\boldsymbol{\theta}_{k\in\bar{\mathcal{L}}}$, which is observed at the PS. After the model

aggregation in (8b), $\tilde{\boldsymbol{\theta}}$ is broadcast to the active clients and it is corrupted by the noise term $\Delta\tilde{\boldsymbol{\theta}}_k$ as in (8a) for $k \in \bar{\mathcal{L}}$. Notice that the first term on the right hand side of (8b) includes no noise corruption since the model parameters are computed at the PS. Let us rewrite (8b) as

$$\tilde{\boldsymbol{\theta}} = \frac{1}{D}\sum_{k\in\mathcal{K}} D_k\boldsymbol{\theta}_k + \frac{1}{D}\sum_{k\in\bar{\mathcal{L}}} D_k\Delta\boldsymbol{\theta}_k. \tag{9}$$

Therefore, using (4b), (9) becomes

$$\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} + \widetilde{\Delta\boldsymbol{\theta}}, \tag{10}$$

where $\widetilde{\Delta\boldsymbol{\theta}} = \frac{1}{D}\sum_{k\in\bar{\mathcal{L}}} D_k\Delta\boldsymbol{\theta}_k$ is the aggregated noise term due to the model transmission from the active clients to the PS. Without loss of generality, we assume that both noise terms due to model transmission in client-PS and PS-client links, i.e., $\widetilde{\Delta\boldsymbol{\theta}}$ and $\Delta\tilde{\boldsymbol{\theta}}_k$ ($k \in \bar{\mathcal{L}}$), are additive white Gaussian noise (AWGN) vectors [20, 31][1]. Hence, regarding the aggregation noise at the PS, we have $\mathbb{E}\{\widetilde{\Delta\boldsymbol{\theta}}\} = \mathbf{0}$ and $\mathbb{E}\{\widetilde{\Delta\boldsymbol{\theta}}\widetilde{\Delta\boldsymbol{\theta}}^{\mathsf{T}}\} = \tilde{\sigma}^2\mathbf{I}_P$. Likewise, at the kth active client ($k \in \bar{\mathcal{L}}$), we have $\mathbb{E}\{\Delta\tilde{\boldsymbol{\theta}}_k\} = \mathbf{0}$, $\mathbb{E}\{\Delta\tilde{\boldsymbol{\theta}}_k\Delta\tilde{\boldsymbol{\theta}}_k^{\mathsf{T}}\} = \sigma_k^2\mathbf{I}_P$ and $\mathbb{E}\{\Delta\boldsymbol{\theta}_{k_1}\Delta\boldsymbol{\theta}_{k_2}^{\mathsf{T}}\} = \mathbf{0}$ for $k_1, k_2 \in \bar{\mathcal{L}}$ and $k_1 \neq k_2$. Furthermore, by using (10) and rewriting (8a) as

$$\tilde{\boldsymbol{\theta}}_k = \boldsymbol{\theta} + \widetilde{\Delta\boldsymbol{\theta}} + \Delta\tilde{\boldsymbol{\theta}}_k = \boldsymbol{\theta} + \widetilde{\Delta\boldsymbol{\theta}}_k, \tag{11}$$

we can define the noise on $\boldsymbol{\theta}$ at the kth client with variance $\tilde{\sigma}^2 + \sigma_k^2$. Specifically, $\tilde{\sigma}^2$ corresponds to the noise term $\widetilde{\Delta\boldsymbol{\theta}}$ in (10) generated during the model transmission from the active clients to the PS, while σ_k^2 corresponds to $\widetilde{\Delta\boldsymbol{\theta}}_k$ in (11), which is due to the transmission of the model parameters from the PS to the kth active client.

Upon the above analysis on the noise corruption, it is clear that two different noise corruptions occur during training: noise corruption with $\tilde{\sigma}^2$ and $\tilde{\sigma}^2 + \tilde{\sigma}_k^2$ for inactive and active clients, respectively. In the sequel, we modify the loss functions used in the training process according to the noise corruption to provide an expectation-based convergence [15, 17].

Let us first consider the loss function for active clients to solve (7a). In order to solve the local problem in (7a) effectively, the loss function should be modified to take into account the effect of noise. Thus, we define a regularized loss function $\bar{\mathcal{F}}_k(\boldsymbol{\theta})$ as

$$\bar{\mathcal{F}}_k(\boldsymbol{\theta}) = \mathcal{F}_k(\boldsymbol{\theta}) + (\tilde{\sigma}^2 + \sigma_k^2)||\mathbf{g}(\boldsymbol{\theta}_k)||^2, \tag{12}$$

where $\tilde{\sigma}^2 + \sigma_k^2$ describes the total noise term added onto $\boldsymbol{\theta}$ at the kth client ($k \in \bar{\mathcal{L}}$) in (11). The loss function in (12) is widely used in stochastic optimization [34] and it can be obtained via a first-order Taylor expansion of the expectation-based loss $\mathbb{E}\{||\mathcal{F}_k(\boldsymbol{\theta} + \widetilde{\Delta\boldsymbol{\theta}}_k)||^2\}$, which can be approximately written as

$$\begin{aligned}\mathbb{E}\{||\mathcal{F}_k(\boldsymbol{\theta}+\widetilde{\Delta\boldsymbol{\theta}}_k)||^2\} &\approx \mathbb{E}\{||\mathcal{F}_k(\boldsymbol{\theta}) + \widetilde{\Delta\boldsymbol{\theta}}_k\nabla\mathcal{F}_k(\boldsymbol{\theta})||^2\},\\ &\approx \mathbb{E}\{||\mathcal{F}_k(\boldsymbol{\theta})||^2\} + \mathbb{E}\{||\widetilde{\Delta\boldsymbol{\theta}}_k||^2\}\mathbb{E}\{||\nabla\mathcal{F}_k(\boldsymbol{\theta})||^2\},\\ &\approx \mathbb{E}\{||\mathcal{F}_k(\boldsymbol{\theta})||^2\} + (\tilde{\sigma}^2 + \sigma_k^2)||\mathbf{g}(\boldsymbol{\theta}_k)||^2, \end{aligned}\tag{13}$$

where the first term in (13) corresponds to the minimization of the loss function with perfect estimation and the second term is the additional cost due to noise. The expectation-based loss in (13) provides a good approximation under the effect of uncertainties due to the noise by adding the regularizer term $(\tilde{\sigma}^2 + \sigma_k^2)||\mathbf{g}(\boldsymbol{\theta}_k)||^2$ [31, 34]. Similarly, the loss function in (7b) for inactive clients at the PS can be regularized as

$$\tilde{\mathcal{F}}_k(\boldsymbol{\theta}) = \mathcal{F}_k(\boldsymbol{\theta}) + \tilde{\sigma}^2||\mathbf{g}(\boldsymbol{\theta}_k)||^2. \tag{14}$$

The combination of (8b), (12) and (14) yields the regularized version of the HFCL problem as

$$\text{Client}: \quad \boldsymbol{\theta}_k = \arg\min_{\boldsymbol{\theta}} \bar{\mathcal{F}}_k(\boldsymbol{\theta}), \quad k \in \bar{\mathcal{L}}, \tag{15a}$$

$$\text{PS}: \quad \boldsymbol{\theta}_k = \arg\min_{\boldsymbol{\theta}} \tilde{\mathcal{F}}_k(\boldsymbol{\theta}), \quad k \in \mathcal{L}, \tag{15b}$$

$$\text{PS}: \quad \tilde{\boldsymbol{\theta}} = \frac{1}{D}\Big\{\sum_{k\in\mathcal{L}} D_k\boldsymbol{\theta}_k + \sum_{k\in\bar{\mathcal{L}}} D_k(\boldsymbol{\theta}_k + \Delta\boldsymbol{\theta}_k)\Big\}, \tag{15c}$$

where the training on the active and inactive clients are performed in (15a) and (15b), respectively. To effectively solve (15), the GD algorithm is utilized to find an optimal[2] model $\boldsymbol{\theta}$ for all clients. Then, the model parameter update is performed iteratively as follows

$$\text{Client}: \quad \boldsymbol{\theta}_k^{(t+1)} = \boldsymbol{\theta}_k^{(t)} - \eta\bar{\mathbf{g}}_k(\boldsymbol{\theta}_k^{(t)}), \quad k \in \bar{\mathcal{L}}, \tag{16a}$$

$$\text{PS}: \quad \boldsymbol{\theta}_k^{(t+1)} = \boldsymbol{\theta}_k^{(t)} - \eta\tilde{\mathbf{g}}_k(\boldsymbol{\theta}_k^{(t)}), \quad k \in \mathcal{L}, \tag{16b}$$

$$\text{PS}: \quad \tilde{\boldsymbol{\theta}}^{(t+1)} = \frac{1}{D}\sum_{k\in\mathcal{K}} D_k\boldsymbol{\theta}_k^{(t+1)}, \tag{16c}$$

where $\bar{\mathbf{g}}_k(\boldsymbol{\theta}_k^{(t)}) = \nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta}_k^{(t)})$ and $\tilde{\mathbf{g}}_k(\boldsymbol{\theta}_k^{(t)}) = \nabla\tilde{\mathcal{F}}_k(\boldsymbol{\theta}_k^{(t)})$ denote the gradient of the regularized loss function in (12) and (14), respectively. In (16), (16a) and (16b) compute the parameter updates for active and inactive clients, respectively. Afterwards, the model aggregation is performed at the PS as in (16c), and the aggregated model $\tilde{\boldsymbol{\theta}}^{(t+1)}$ is broadcast to the active clients.

B. Communication Delay During Model Training

The bandwidth resources need to be optimized to reduce the latency of the transmission of both $\boldsymbol{\theta}_k$ ($k \in \bar{\mathcal{L}}$) and $\mathcal{D}_k$ ($k \in \mathcal{L}$) to the PS during training. Let τ_k be the communication time for the kth client to transmit its either dataset ($\mathcal{D}_k$ for $k \in \mathcal{L}$) or model parameters ($\boldsymbol{\theta}_k$ for $k \in \bar{\mathcal{L}}$). We define

$$\tau_k = \frac{d_k}{R_k}, \tag{17}$$

[1]While the Gaussian assumption is widely used in the relevant literature [17, 20, 31, 32], it is worth noting that this assumption is not completely satisfied in practice due to the source/channel coding and quantization operations during the transmission of the model parameters. In addition, recent works, e.g., [33], show that the noise on the model updates can also be modeled as unimodal symmetric distribution, which is close to Gaussian distribution.

[2]The optimality of the proposed approach is subject to finding the minimizer of the loss function $\bar{\mathcal{F}}_k(\boldsymbol{\theta})$ at each iteration via the GD method as in the other FL works [17, 33, 35]. While the global optimality of GD can be guaranteed for shallow neural networks, e.g., with a single layer, it usually reaches to a local optimum for wide and deep learning models [2].

where d_k denotes the number of dataset symbols to transmit and $R_k = B_k \ln(1 + \mathsf{SNR}_k)$ is the achievable transmission rate. Herein, B_k and SNR_k denote the allocated bandwidth and the signal-to-noise ratio (SNR) for the kth client, respectively. The PS solves $\min_{B_k} \max_{k\in\mathcal{K}} \ \tau_k$ to optimize the bandwidth allocation by minimizing the maximum communication delay. This is because the model aggregation in the PS can be performed only after the completion of the slowest transmission for $k \in \mathcal{K}$. Although R_k can vary for $k \in \mathcal{K}$, d_k differentiates more significantly than R_k between the inactive (i.e., $k \in \mathcal{L}$) and active (i.e., $k \in \bar{\mathcal{L}}$) clients [36, 37]. Especially, depending on the client type, d_k can be given by

$$d_k = \begin{cases} P, & k \in \bar{\mathcal{L}} \\ \mathsf{d}_k, & k \in \mathcal{L} \end{cases}, \tag{18}$$

which is fixed to the number of model parameters P for the active clients, and to $\mathsf{d}_k = D_k(U_x V_x + U_y V_y)$ for D_k input ($\in \mathbb{R}^{U_x \times V_x}$) and output ($\in \mathbb{R}^{U_y \times V_y}$) dataset samples.

Since the dataset size is usually larger than the number of model parameters in ML applications, i.e., $\mathsf{d}_{k\in\mathcal{L}} > P$ [11, 12, 19], the dataset transmission of the inactive clients is expected to take longer than the model transmission of the active clients, i.e., $\tau_{k\in\mathcal{L}} > \tau_{k\in\bar{\mathcal{L}}}$ [36]. Previous FL-based works reported that $\tau_{k\in\mathcal{L}}$ can be approximately 10 times longer than $\tau_{k\in\bar{\mathcal{L}}}$ [19, 20]. This introduces a significant delay especially at the beginning of the training since the HFCL problem in (15b) or (16b) can be performed only if $\mathcal{D}_{k\in\mathcal{L}}$ is collected at the PS for the first iteration. To tackle this issue and keep the training continuing, in the following, we propose two approaches, namely, ICpC and SDT in Section IV and V, respectively. Both of these techniques are applied only at the beginning of the training to handle effectively the dataset transmission of the inactive clients.

Algorithm 1 HFCL-ICpC

Input: η, $\mathcal{D}_{k\in\mathcal{K}}$.
Output: $\boldsymbol{\theta}$.
1: Initialize with $\boldsymbol{\theta}_k^{(t)}$ for $t = 0$.
2: **repeat**
3: **if** $k \in \bar{\mathcal{L}}$ [Active clients],
4: **if** $t = 0$
5: $t' := t$.
6: **repeat**
7: $\boldsymbol{\theta}_k^{(t'+1)} = \boldsymbol{\theta}_k^{(t')} - \eta \bar{\mathbf{g}}_k(\boldsymbol{\theta}_k^{(t')})$.
8: $t' \leftarrow t' + 1$.
9: **until** $t' = N$
10: $\boldsymbol{\theta}_k^{(t+1)} = \boldsymbol{\theta}_k^{(N)}$.
11: **else**
12: $\boldsymbol{\theta}_k^{(t+1)} = \boldsymbol{\theta}_k^{(t)} - \eta \bar{\mathbf{g}}_k(\boldsymbol{\theta}_k^{(t)})$.
13: **end**
14: Send $\boldsymbol{\theta}_k^{(t+1)}$ to the PS.
15: **if** $k \in \mathcal{L}$ [Inactive clients],
16: **if** $t = 0$
17: Send $\mathcal{D}_k$ to the PS.
18: **else**
19: $\boldsymbol{\theta}_k^{(t+1)} = \boldsymbol{\theta}_k^{(t)} - \eta \tilde{\mathbf{g}}_k(\boldsymbol{\theta}_k^{(t)})$.
20: **end**
21: **end**
22: Aggregate models as $\boldsymbol{\theta}^{(t+1)} = \frac{1}{D}\sum_{k\in\mathcal{K}} D_k \boldsymbol{\theta}_k^{(t+1)}$.
23: Broadcast $\boldsymbol{\theta}^{(t+1)}$ to the clients for $k \in \bar{\mathcal{L}}$.
24: $t \leftarrow t + 1$.
25: **until** convergence

IV. HFCL With Increased Computation-per-Client

Prior to the model training, i.e., at $t = 0$, the active clients need to wait until the inactive clients complete the dataset transmission, during which the active clients perform only one model computation. In order to keep the active clients continue model computing during the data transmission of inactive clients, we propose the ICpC approach, wherein the active clients move forward computing local model updates, but do not send them to the PS until the dataset transmission of the inactive clients is completed [15]. This approach improves the convergence rate and the learning performance due to the continuation of the model updates at the inactive clients as similar observations are also reported in [15], in which FL-only training is presented.

The algorithmic steps of the HFCL-ICpC approach are presented in Algorithm 1, for which the inputs are the datasets $\mathcal{D}_{k\in\mathcal{K}}$ and the learning rate η. Different from HFCL, HFCL-ICpC involves with the model updates at the active clients during first communication round, i.e., $t = 0$, as in the lines $6-9$ of Algorithm 1. Using (17) and (18), and defining Q as the block size, the active clients can perform $N = \frac{\max_{k\in\mathcal{L}} \mathsf{d}_k}{Q}$ iterations when $t = 0$ until the dataset transmission of the inactive clients is completed (Line 10 of Algorithm 1). Although this method keeps the active clients busy with model computation instead of staying idle, it does not reduce the communication latency at the first iteration. As a result, the active clients perform N local updates, which improve the convergence rate from $O(\frac{1}{t})$ to $O(\frac{N^2}{t})$ as compared to HFCL [32].

V. HFCL With Sequential Data Transmission

Compared to HFCL and HFCL-ICpC, wherein the first communication round awaits for the completion of the transmission of inactive clients' datasets $\mathcal{D}_{k\in\mathcal{L}}$, HFCL-SDT involves with the transmission of smaller blocks of datasets. Hence, the data transmission time per communication round is smaller while the total amount of time to transmit $\mathcal{D}_{k\in\mathcal{L}}$ to the PS is the same as in HFCL and HFCL-ICpC.

The algorithmic steps of the HFCL-SDT approach are given in Algorithm 2. Let Q be the block size of each portion. The number of blocks can be calculated as $N = \frac{\max_{k\in\mathcal{L}} \mathsf{d}_k}{Q}$, which aims at minimizing the latency when $t = 0$ by taking into account the largest dataset in terms of $\max_{k\in\mathcal{L}} \mathsf{d}_k$ since we need to wait until the transmission of the largest dataset. If $\mathsf{d}_{k_1} = \mathsf{d}_{k_2}$ for $k_1, k_2 \in \mathcal{L}$, the size of the transmitted data for all clients becomes the same and equals to Q[3]. If N is

[3]As a special case, $Q = P$ can be selected to allow the transmitted data from both inactive and active clients is equal. However, this is only possible if $N = \frac{\max_{k\in\mathcal{L}} \mathsf{d}_k}{P} < \gamma T$ so that dataset transmission is completed before γT iterations. If $\gamma = 1$, then both dataset transmission and the training are completed at the same time, which is not efficient. Empirically, $\gamma = 0.1$ is a good choice.

Algorithm 2 HFCL-SDT

Input: η, $\mathcal{D}_{k\in\mathcal{K}}$, $\mathcal{F}_k$ as in (2).
Output: $\boldsymbol{\theta}$

1: Initialize with $\boldsymbol{\theta}_k^{(t)}$ for $t=0$.
2: **repeat**
3: **if** $k\in\bar{\mathcal{L}}$ [Active clients],
4: $\boldsymbol{\theta}_k^{(t+1)}=\boldsymbol{\theta}_k^{(t)}-\eta\bar{\mathbf{g}}_k(\boldsymbol{\theta}_k^{(t)})$.
5: Send $\boldsymbol{\theta}_k^{(t+1)}$ to the PS.
6: **if** $k\in\mathcal{L}$ [Inactive clients],
7: **if** $t\leq N$
8: Send $\mathcal{D}_k^{(i)}$ to the PS for $i=(t-1)Q+1,\ldots,tQ$.
9: Use the collected dataset and compute $\mathcal{F}_k(\boldsymbol{\theta}_k^{(t)})=\frac{\sum_{i=1}^{tQ}\|f(\mathcal{X}_k^{(i)}|\boldsymbol{\theta}_k^{(t)})-\mathcal{Y}_k^{(i)}\|^2}{tQ}$.
10: **if** $t>N$
11: Use the whole dataset and compute $\mathcal{F}_k(\boldsymbol{\theta}_k^{(t)})=\frac{\sum_{i=1}^{D_k}\|f(\mathcal{X}_k^{(i)}|\boldsymbol{\theta}_k^{(t)})-\mathcal{Y}_k^{(i)}\|^2}{D_k}$.
12: **end**
13: $\boldsymbol{\theta}_k^{(t+1)}=\boldsymbol{\theta}_k^{(t)}-\eta\tilde{\mathbf{g}}_k(\boldsymbol{\theta}_k^{(t)})$.
14: **end**
15: Aggregate models as $\boldsymbol{\theta}^{(t+1)}=\frac{1}{D}\sum_{k\in\mathcal{K}}D_k\boldsymbol{\theta}_k^{(t+1)}$.
16: Broadcast $\boldsymbol{\theta}^{(t+1)}$ to the clients for $k\in\bar{\mathcal{L}}$.
17: $t\leftarrow t+1$.
18: **until** convergence

not an integer, then $N=\lceil\frac{\max_{k\in\mathcal{L}}\mathsf{d}_k}{Q}\rceil$ can be selected. Let us assume that $\mathcal{F}_k(\boldsymbol{\theta})$ is the regression loss as in (2)[4], then, the loss function for the inactive clients in the case of the HFCL-SDT algorithm are computed as

$$\mathcal{F}_k(\boldsymbol{\theta}_k^{(t)})=\begin{cases}\frac{\sum_{i=1}^{tQ}\|f(\mathcal{X}_k^{(i)}|\boldsymbol{\theta}_k^{(t)})-\mathcal{Y}_k^{(i)}\|^2}{tQ}, & t\leq N\\ \frac{\sum_{i=1}^{D_k}\|f(\mathcal{X}_k^{(i)}|\boldsymbol{\theta}_k^{(t)})-\mathcal{Y}_k^{(i)}\|^2}{D_k}, & t>N\end{cases}, \tag{19}$$

as given in lines $6-14$ of Algorithm 2. The size of the training dataset, collected at the PS (i.e., tQ) for $\mathcal{F}_k(\boldsymbol{\theta}_k^{(t)})$, becomes larger as $t\to N$, and becomes equal to D_k when $t=N$. When $t>N$, $\mathcal{F}_k(\boldsymbol{\theta}_k^{(t)})$ is computed for the whole dataset of inactive clients for $k\in\mathcal{L}$, as in the 11th step of Algorithm 2. While the size of the transmitted data from the inactive clients changes, the size of the transmitted information is fixed as P for the active clients. This approach not only reduces the latency of the inactive clients, but also improves the learning accuracy as compared to HFCL since the features in the data are quickly learned at the PS due to the use of smaller datasets at the beginning. As a result, the model parameters corresponding to the inactive clients are computed in a mini-batch learning way, in which the dataset is partitioned into mini-batches of size Q. Thus, the convergence rate of the inactive clients in HFCL-SDT is $O(\frac{1}{\sqrt{Qt}}+\frac{1}{t})$ while the rate of active clients is $O(\frac{1}{t})$ [38].

VI. Complexity Analysis and Communication Overhead

In this section, we investigate the efficiency of the proposed HFCL framework in terms of complexity and communication overhead.

[4]The same expression can also be written for the loss function in (3).

A. Complexity Analysis

The complexity of the learning schemes, i.e., CL, FL and HFCL, can be analyzed in terms of convergence speed since the computation of the model parameters is done by solving the optimization problems (1), (4) and (16) via GD during training [17, 31, 39]. Specifically, solving the optimization problems (1), (4) and (16) for in CL, FL and HFCL, respectively, yields the same convergence rate of $O(1/t)$ although the accuracy of these algorithms differ depending on the noise corruption ($\tilde{\sigma}^2$ and σ_k^2) on the model parameters [17]. In particular, CL has higher learning accuracy than FL and HFCL since the model parameters are noise-free in CL while the accuracy of HFCL is between CL and FL since only a part of the model parameters (of inactive clients) are noise-free (see, e.g., Fig. 5). Furthermore, the convergence rates of the proposed HFCL techniques, HFCL-ICpC and HFCL-SDT, are different from those of CL, FL and HFCL since they involve the participation of both active and inactive clients during training. In HFCL-ICpC, the convergence rate of solving the optimization problem for inactive clients is $O(\frac{1}{t})$ while it is $O(\frac{N^2}{t})$ for the active clients due to performing N local updates per iteration [15]. As a result, the convergence is faster in active clients as compared to the inactive ones. Also in HFCL-SDT, the convergence rates of the active and inactive clients are $O(\frac{1}{t})$ and $O(\frac{1}{\sqrt{Qt}}+\frac{1}{t})$, respectively, due to the dataset is partitioned into Q mini-batches in inactive clients [38]. Hence, one can conclude that the complexity of HFCL-ICpC is less than HFCL-SDT since the former reaches to convergence faster (see, e.g, Fig. 4). We summarize the convergence rates of the HFCL methods for both active and inactive clients in Table III.

Next, we investigate the convergence of the proposed HFCL framework. In the literature, the convergence of the ML models has been studied for centralized [40] and federated [17, 31] schemes separately. In this paper, we analyze the convergence of the hybrid scenario, in which the convergence of the loss functions for active ($\bar{\mathcal{F}}_k(\boldsymbol{\theta})$) and inactive ($\tilde{\mathcal{F}}_k(\boldsymbol{\theta})$) clients in the presence of corrupted model parameters. First, similar to the previous studies [15, 17, 31], we make the following assumptions needed to ensure the convergence, which are typical for the l_2-norm regularized linear regression, logistic regression, and softmax classifiers.

Assumption 1: The loss function $\mathcal{F}_k(\boldsymbol{\theta})$ is convex, i.e., $\mathcal{F}_k((1-\lambda)\boldsymbol{\theta}+\lambda\boldsymbol{\theta}')\leq(1-\lambda)\boldsymbol{\theta}\mathcal{F}_k(\boldsymbol{\theta})+\lambda\mathcal{F}_k(\boldsymbol{\theta}')$ for $\lambda\in[0,1]$ and arbitrary $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$.

Assumption 2: $\mathcal{F}_k(\boldsymbol{\theta})$ is *L-Lipschitz*, i.e., $||\mathcal{F}_k(\boldsymbol{\theta})-\mathcal{F}_k(\boldsymbol{\theta}')||\leq L||\boldsymbol{\theta}-\boldsymbol{\theta}'||$ for arbitrary $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$.

Assumption 3: $\mathcal{F}_k(\boldsymbol{\theta})$ is β-*Smooth*, i.e., $||\nabla\mathcal{F}_k(\boldsymbol{\theta})-\nabla\mathcal{F}_k(\boldsymbol{\theta}')||\leq\beta||\boldsymbol{\theta}-\boldsymbol{\theta}'||$ for arbitrary $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$.

In order to prove the convergence of the loss functions for inactive and active clients, i.e., $\tilde{\mathcal{F}}_k(\boldsymbol{\theta})$ and $\bar{\mathcal{F}}_k(\boldsymbol{\theta})$, we first investigate the β-*Smoothness* of $\bar{\mathcal{F}}_k(\boldsymbol{\theta})$ in the following lemma.

Lemma 1. *$\bar{\mathcal{F}}_k(\boldsymbol{\theta})$ is a $\bar{\beta}$-Smooth function with $||\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})-\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta}')||\leq\bar{\beta}||\boldsymbol{\theta}-\boldsymbol{\theta}'||$, where $\bar{\beta}=(1+\tilde{\sigma}^2+\sigma_k^2)\beta$.*

TABLE III
CONVERGENCE RATES FOR HFCL, HFCL-ICPC, AND HFCL-SDT

HFCL		HFCL-ICpC		HFCL-SDT	
Inactive	Active	Inactive	Active	Inactive	Active
$O(\frac{1}{t})$	$O(\frac{1}{t})$	$O(\frac{1}{t})$	$O(\frac{N^2}{t})$	$O(\frac{1}{\sqrt{Qt}}+\frac{1}{t})$	$O(\frac{1}{t})$

Proof: See Appendix A. ∎

Using Lemma 1 and (14), it is straightforward to show that $\tilde{\mathcal{F}}_k(\boldsymbol{\theta})$ is a $\tilde{\beta}$-*smooth* function with $\tilde{\beta} = \tilde{\sigma}^2\beta$.

Theorem 1. *At the kth active client ($k \in \bar{\mathcal{L}}$), the loss function $\bar{\mathcal{F}}_k(\boldsymbol{\theta})$ satisfies*

$$\bar{\mathcal{F}}_k(\boldsymbol{\theta}^{(t)}) - \bar{\mathcal{F}}_k(\boldsymbol{\theta}^\star) \leq ||\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^\star||^2 \frac{1}{2\eta}\frac{1}{t}, \tag{20}$$

where the learning rate is subject to $\eta \leq \frac{1}{(1+\tilde{\sigma}^2+\sigma_k^2)\beta}$, and $\boldsymbol{\theta}^\star$ is the minimizer of $\bar{\mathcal{F}}_k(\boldsymbol{\theta})$.

Proof: See Appendix B. ∎

Based on Theorem 1, the loss function of the inactive clients $\tilde{\mathcal{F}}_k(\boldsymbol{\theta})$ is said to be convergent with

$$\tilde{\mathcal{F}}_k(\boldsymbol{\theta}^{(t)}) - \tilde{\mathcal{F}}_k(\boldsymbol{\theta}^\star) \leq ||\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^\star||^2 \frac{1}{2\eta}\frac{1}{t}, \tag{21}$$

where the learning rate obeys to $\eta \leq \frac{1}{(1+\tilde{\sigma}^2)\beta}$ since the problem for inactive clients is corrupted by the noise $\widetilde{\Delta\boldsymbol{\theta}} = \frac{1}{D}\sum_{k\in\bar{\mathcal{L}}} D_k \Delta\boldsymbol{\theta}_k$ with variance $\tilde{\sigma}^2$ as in (10).

Consequently, if all clients perform CL, i.e., $L = K$, we have $\tilde{\sigma}^2 = \sigma_k^2 = 0$. Conversely, if all clients perform FL, i.e., $L = 0$, the noise-free model parameters in (9) will vanish and the noise variance becomes $\tilde{\sigma}^2_{\text{FL}} = \frac{1}{D}\sum_{k\in\mathcal{K}} D_k\sigma_k^2$. Thus, FL converges with lower learning performance than that of CL [15]. On the other hand, assuming $D_1 = \cdots = D_K$ and $\sigma_1^2 = \cdots = \sigma_K^2$, the noise variance in HFCL $\tilde{\sigma}^2_{\text{HFCL}} = \frac{1}{D}\sum_{k\in\bar{\mathcal{L}}} D_k\sigma_k^2$ becomes less than $\tilde{\sigma}^2_{\text{FL}}$ since $\sum_{k\in\bar{\mathcal{L}}}\sigma_k^2 < \sum_{k\in\mathcal{K}}\sigma_k^2$, where $K - L < K$ for $L \geq 1$. Therefore, the performance of HFCL is upper (lower) bounded by CL (FL) as $0 \leq \tilde{\sigma}^2_{\text{HFCL}} \leq \tilde{\sigma}^2_{\text{FL}}$.

Remark 1: If $D_{k_1} \neq D_{k_2}$, where $k_1, k_2 \in \mathcal{K}$ and $k_1 \neq k_2$, the performance of HFCL may be lower than FL. For instance, assuming that $\sigma_1^2 = \cdots = \sigma_K^2$, if $\frac{1}{D}\sum_{k\in\bar{\mathcal{L}}} D_k > \frac{1}{D}\sum_{k\in\mathcal{K}} D_k$, HFCL converges with higher noise than that of FL since $\tilde{\sigma}^2_{\text{HFCL}} > \tilde{\sigma}^2_{\text{FL}}$. While the equality of $\sigma^2_{k\in\mathcal{K}}$ may be practical, the performance of HFCL and FL is subject to the size of the datasets [15, 17].

B. Communication Overhead

The communication overhead is due to the transmission of the dataset or the model parameters that are exchanged between the clients and the PS, wherein several signal processing techniques, such as channel acquisition, quantization, and resource allocation, are employed. Hence, the communication overhead can be measured by the number of transmitted symbols during model training [7, 12, 19, 20]. Regarding the communication overhead of CL ($\mathcal{T}_{\text{CL}}$), it can be given by the number of symbols used to transmit datasets while the overhead during FL ($\mathcal{T}_{\text{FL}}$) is proportional to the number of communication rounds T and model parameters P. Let $\mathsf{D} = \sum_{k\in\mathcal{K}} \mathsf{d}_k$ be the number of symbols of the whole dataset, then the communication overhead of CL, FL and HFCL are given respectively as

$$\mathcal{T}_{\text{CL}} = \mathsf{D}, \tag{22}$$
$$\mathcal{T}_{\text{FL}} = 2TPK, \tag{23}$$
$$\mathcal{T}_{\text{HFCL}} = L\mathsf{d}_{k\in\mathcal{L}} + 2TP(K-L), \tag{24}$$

where $\mathcal{T}_{\text{HFCL}}$ includes the transmission of the dataset from L inactive clients and the model parameters from $K - L$ active clients. It is reasonable to assume that the size of the datasets is larger than the size of the model parameters, i.e., $\mathcal{T}_{\text{FL}} \leq \mathcal{T}_{\text{CL}}$ [7, 11, 12, 19, 20], which gives $\mathcal{T}_{\text{FL}} \leq \mathcal{T}_{\text{HFCL}} \leq \mathcal{T}_{\text{CL}}$. Notice that the communication overhead is equal for all HFCL techniques proposed in this work, since they all involve the same amount of dataset and model transmission.

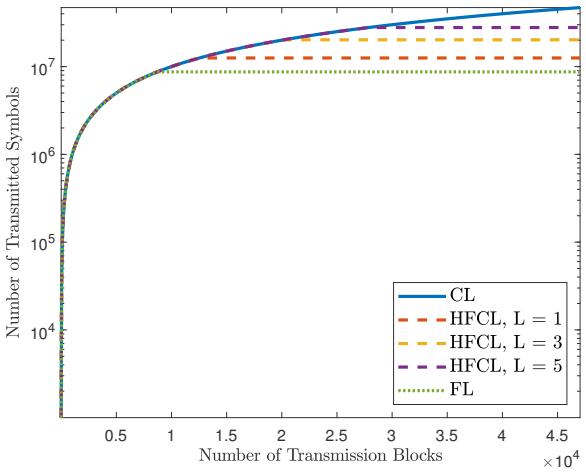


Fig. 2. Communication overhead comparison.

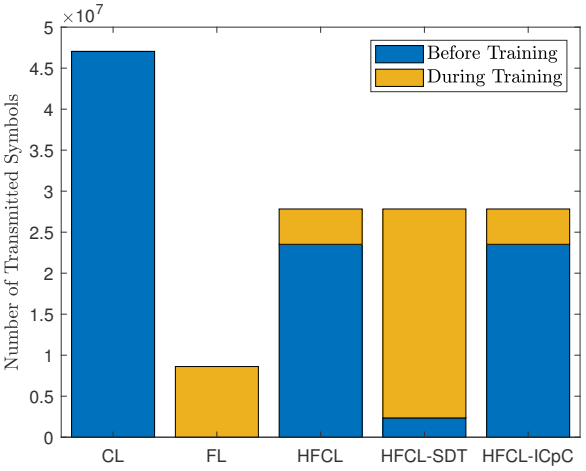


Fig. 3. Number of transmitted symbols before and during training ($L = 5$).

VII. Numerical Simulations

In this section, we present the performance of the proposed HFCL frameworks in comparison to traditional CL- and FL-based training. We evaluate the performance on two datasets: i) image classification on the MNIST dataset [41] and ii) 3D object detection on the Lyft Level 5 AV dataset [42].

A. Image Classification

We evaluate the performance on the MNIST dataset [41] including 28×28 gray-scale images of handwritten digits with 10 classes. The number of symbols on the whole dataset is $\mathsf{D} = 28^2 \cdot 60,000 \approx 47 \times 10^6$. During model training, the dataset is partitioned into $K = 10$ blocks, each of which is available at the clients with identically independent distribution. Further, we train a CNN with 6 layers. The first layer is 28×28 input layer. The second and the fourth layers are convolutional layers with 5×5@128 and 3×3@128 spatial filters, respectively. After each convolutional layer, there is a ReLU layer, which operates $\max(0, x)$ for its input x. The output layer is a classification layer, which computes the probability distribution of the input data for 10 classes. Thus, we have $P = 128 \cdot (5^2 + 3^2) = 4,352$ learnable parameters. The validation data of the MNIST dataset include $10,000$ images. The learning rate is selected as 0.001, which is reduced by half after each 30 iterations, and the mini-batch size is selected as 128 for CL. The loss function was the cross-entropy cost as $-\frac{1}{D}\sum_{i=1}^{D}\sum_{c=1}^{\bar{C}}\left[\mathcal{Y}_i^{(c)} \ln \hat{\mathcal{Y}}_i^{(c)} + (1 - \mathcal{Y}_i^{(c)}) \ln(1-\hat{\mathcal{Y}}_i^{(c)})\right]$, where $\{\mathcal{Y}_i^{(c)}, \hat{\mathcal{Y}}_i^{(c)}\}_{i=1,c=1}^{D,\bar{C}}$ are the true and predicted response for the classification layer with $\bar{C} = 10$. The classification accuracy is $\text{Accuracy}(\%) = \frac{\mathsf{U}}{D} \times 100$, in which the model identified the image class correctly U times. Further, we define the SNR during model transmission as $\text{SNR}_{\boldsymbol{\theta}} = 20\log_{10}\frac{||\boldsymbol{\theta}||_2^2}{\sigma_{\boldsymbol{\theta}}^2}$, where $\sigma_{\boldsymbol{\theta}}^2 = \tilde{\sigma}^2 + \sigma_k^2$ denotes the total amount of noise variance during model transmission, for which we assume $\sigma_{L+1}^2 = \cdots = \sigma_K^2$ for simplicity.

Fig. 2 shows the communication overhead of CL, FL, and HFCL for $L = \{0, 1, 3, 5, 7, 10\}$. During model training, we assume that 1000 data symbols are transmitted at each transmission block. Thus, it takes approximately 47×10^3 transmission blocks to complete CL-based training, while FL demands approximately $8,5 \times 10^3$ data blocks, which are approximately 6 times lower than that of CL. The communication overhead of HFCL (as well as HFCL-SDT and HFCL-ICpC) is between CL and FL since it depends on the number of inactive clients L and approaches to $\mathcal{T}_{\text{CL}}$ as $L \to K$.

To explicitly present the overhead for the hybrid frameworks, the number of transmitted symbols before ($t = 0$) and during ($t > 0$) training is presented in Fig. 3. Specifically, the whole communication overhead of CL is at $t = 0$ before training due the dataset transmission. In contrast, the overhead of FL is observed during training since no dataset transmission is involved, yet model transmission is taken place when $t > 0$. For the HFCL algorithms, they involve communication overheads before and during training since they involve both dataset and model transmission. Also, it is clear that the overhead of all hybrid frameworks are the same because they all involve the same amount of dataset and model transmission. However, they are distinguished in terms of the number of symbols transmitted before and during training. While HFCL and HFCL-ICpC have the same amount of overhead during training, HFCL-SDT has a lower overhead before training due to transmitting the datasets sequentially. Furthermore, we see that the overhead of FL is twice the amount during training for HFCL and HFCL-ICpC. This is because $L = 5$ for HFCL and HFCL-ICpC, and $L = K = 10$ for FL. While the overhead of FL may seem lower compared to our HFCL frameworks, the former does not work if the clients have no capability for model computation, which is taken into account in the latter frameworks at the cost of a moderate increase in the overhead.

In Fig. 4, we present the training performance of the competing methods for $L = 5$, $\text{SNR}_{\boldsymbol{\theta}} = 20$ dB, and $B = 5$ quantization bits with respect to number of communication

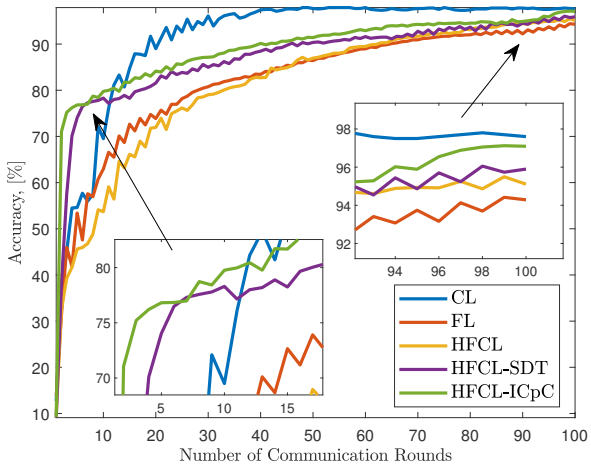


Fig. 4. Classification accuracy versus the number of communication rounds ($L = 5$ $\text{SNR}_{\boldsymbol{\theta}} = 20$ dB, and $B = 5$).

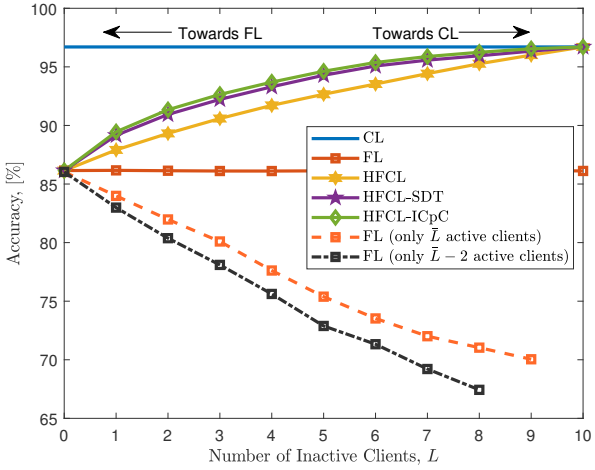
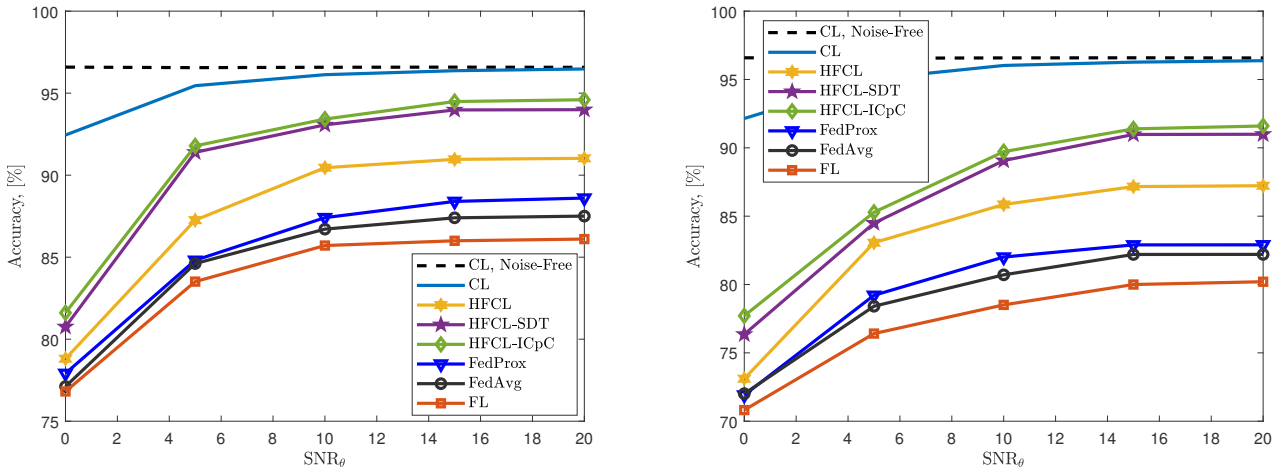


Fig. 5. Classification accuracy versus L ($\text{SNR}_{\boldsymbol{\theta}} = 20$ dB, and $B = 8$).



(a) (b)

Fig. 6. Classification accuracy versus SNR$_{\boldsymbol{\theta}}$ for (a) i.i.d. and (b) non-i.i.d. datasets ($L = 5$ and $B = 5$).

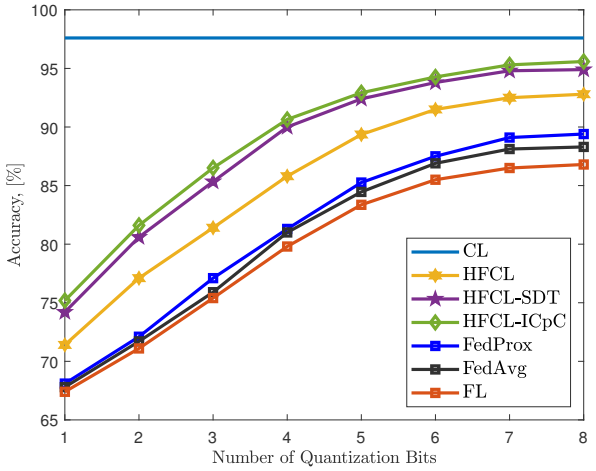
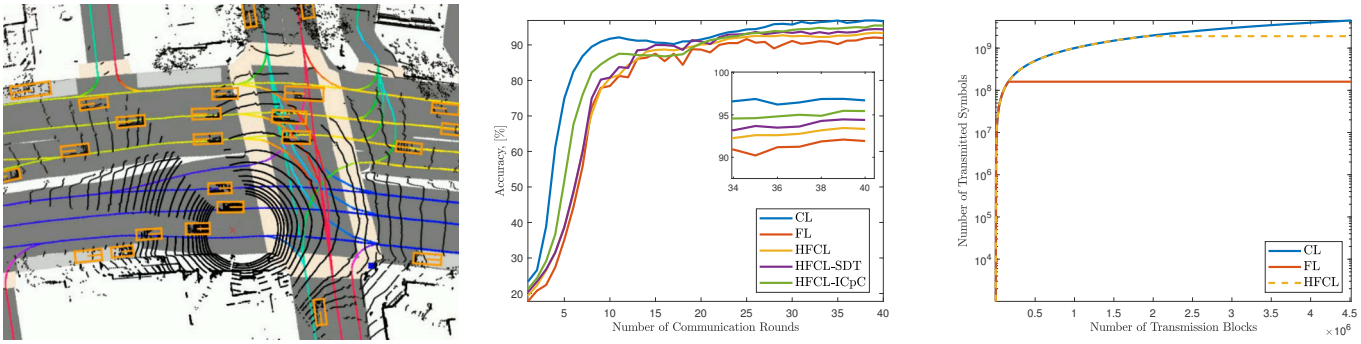


Fig. 7. Classification accuracy versus B (SNR$_{\boldsymbol{\theta}} = 20$ dB).

rounds. Note that we assume the dataset transmission is completed in CL prior to the training, and the completion of each communication round in FL corresponds to each iteration in CL. HFCL-ICpC has a higher accuracy at the beginning of the training due to multiple updates at the active clients, and it keeps outperforming HFCL and HFCL-SDT for the remaining iterations. Thanks to STD, HFCL-SDT also has higher performance than HFCL, which performs better than conventional FL. Notice that all the proposed HFCL approaches have moderate performance between FL and CL, as theoretically proved in Section VI. We note that the performance of FL is computed by assuming that all K clients participate in training, which is not possible if there exist inactive clients, for which FL is not the applicable training framework.

Fig. 5 shows the classification accuracy with respect to the number of inactive clients L when SNR$_{\boldsymbol{\theta}} = 20$ dB. The proposed HFCL approaches perform better than FL for $0 < L < K$ since, in FL, the collected models from the active clients are corrupted by the wireless channel whose effects reduce as $L \to K$. When $L = 0$, HFCL, HFCL-SDT, and HFCL-ICpC are identical to FL since all the clients are active whereas they perform the same as CL if $L = K = 10$ since all the clients are inactive, i.e., they transmit their datasets to the PS, and the noise-free model parameters can be used for model training. Comparing the HFCL algorithms, it yields that both HFCL-ICpC and HFCL-SDT provide higher accuracy than HFCL for $0 < L < K$. HFCL-ICpC starts the training process when $t = 0$ with locally-updated models at the active clients, thus, it provides a higher accuracy than both HFCL and HFCL-SDT. Also, HFCL-SDT starts the training with the same conditions as in HFCL while HFCL-SDT incorporates smaller datasets at the beginning for $t < N$. Thus, higher accuracy levels are reached quicker than HFCL, which computes the model parameters on the whole local dataset of the inactive clients for $t < N$, leading to a slower convergence rate.

In Fig. 5, we also present the performance of FL with only $\bar{L}$ active clients where $\bar{L} = K - L$ is the number of active clients. That is to say, the learning model is trained on only the dataset of active clients, whereas it is tested on the whole dataset. We also consider the scenario that the training is conducted by only $\bar{L} - 2$ active clients, two of which do not participate into the training due to the privacy concerns or limited conditions, such as having very low channel gain [33, 43]. We observe that FL becomes unable to learn the data as L increases since the training is conducted only on the datasets of active clients. This shows the effectiveness of the proposed HFCL approach, in which all of the clients participate in the learning stage. It is worthwhile to note that when $L = K = 10$, there will be no active clients, hence FL will not be performed. The performance loss can be severe as $L \to K$ due to the absence of inactive clients' datasets if the dataset is non-identically distributed because the active clients cannot learn the whole features in the dataset of other devices.



(a) (b) (c)

Fig. 8. 3D object detection. (a) Visualization of input and output data, (b) object detection performance, and (c) communication overhead.

We further present the classification performance with respect to the noise corruption on the model parameters for both identically independent distribution (IID) and non-IID cases in Fig. 6a and Fig. 6b, respectively. In IID scenario, the dataset is uniform randomly shuffled over the clients while for non-IID case, the dataset is sorted by labels and each client is then randomly assigned with 1 or 2 labels [15, 35]. We compare the classification accuracy of the proposed HFCL methods with existing advanced FL techniques, i.e., FedAvg [15] and FedProx [44]. Specifically, in FedAvg, all the clients apply N local model updates in each communication round whereas $N = 1$ in conventional FL. Also, the clients employ different number of local model updates in FedProx (i.e., variable N) [44]. We set $\text{SNR}_{\boldsymbol{\theta}}$ as $\text{SNR}_{\boldsymbol{\theta}} = [0, 20]$ dB and $\text{SNR}_{\boldsymbol{\theta}} = \text{SNR}_{\mathcal{D}}$, which is defined as the SNR of the noise added onto the transmitted datasets. We can see from Fig. 6 that the FL-based methods (e.g., FedAvg, FedProx and FL) are exposed to approximately 4.5% loss in the learning accuracy while the HFCL methods perform 3.5% poorer due to non-IID datasets. The slight performance improvement of HFCL methods can be attributed to the computation of model parameters at the PS. Furthermore, we observe that FedAvg has superior performance than FL due to multiple local model updates while FedProx performs better due to performing different number of local updates. Nevertheless, the proposed HFCL techniques have higher accuracy due to the computation of the model updates (of inactive clients) in the PS. As expected, compared to the FL-based methods and the proposed hybrid algorithms, CL is more robust against noise because its model parameters are not corrupted by wireless transmission since they are computed at the PS. Comparing the effect of noise in datasets and model parameters, it can be concluded that the learning accuracy depends more on the corruptions in the model than the datasets. This is because the noise in the model makes it unable to learn the data while the noise in the dataset is, to some extend, helpful to make the model robust against the imperfections in the data. Therefore, artificial noise is usually added onto the dataset in many applications, such as image classification [2, 3], and physical layer design in wireless communications [20, 25].

In Fig. 7, we present the classification accuracy with respect to the quantization level of the model parameters for $B \in [1, 8]$ when $\text{SNR}_{\boldsymbol{\theta}} = 20$ dB. Note that the quantization is only applied to wireless-transmitted models, i.e., FL, FedAvg, FedProx and HFCL. Hence, the performance of CL does not change. The model parameters are quantized layer by layer between the maximum and minimum weights of each layer [14]. As expected, the classification accuracy improves as B increases due to the enhancement of the model precision. We can conclude that at least a 5-bit quantization is required for a reliable classification accuracy.

B. 3D Object Detection

We evaluate the performance of the HFCL framework on the 3D object detection problem in vehicular networks, based on the Lyft Level 5 AV dataset [42], collected from lidar and cameras mounted on vehicles [6]. The input data are selected as a top view image of the ego vehicle, which includes the received lidar signal strengths for different elevations, as shown in Fig. 8a. The output data are the classified representation of the vehicles/objects as boxes, which are obtained by the preprocessing of the images from the cameras, as illustrated in Fig. 8a [5]. The training dataset is collected from 10 vehicles in different areas after preprocessing of the camera and lidar data. We assume that $L = 3$ of the vehicles are inactive while the remaining ones are active clients. Each dataset includes 10^3 input-output pairs, whose sizes are $336 \times 336 \times 3$ and $336 \times 336 \times 1$, respectively. Hence, the total number of data symbols are $(336 \times 336 \times 3 + 336 \times 336) \times 10^4 \approx 4.5 \times 10^9$. The dataset has 9 classes, i.e., car, motorcycle, bus, bicycle, truck, pedestrian, other vehicle, animal, and emergency vehicle, which are represented by the boxes as shown in Fig. 8a. We have used the U-net [45] with 8 convolutional layers to learn the features in the input data and achieve 3D object detection and segmentation. The total number of parameters in U-net is approximately 2×10^6, and the training is conducted for $T = 40$ communication rounds.

The training performance of the competing methods is presented in Fig. 8b, from which we obtain similar observations as in the image classification scenario in the previous part.

[5] While we assumed that the dataset is labeled, the annotation of the objects in the data samples is an important task, for which the reader can refer to [6, 7].

HFCL methods provide a moderate performance between CL and FL. Nevertheless, all of the vehicles can participate in training while the conventional FL methods cannot support it and the communication overhead of CL is prohibitive. Thus, HFCL provides a solution for the the trade-off between the computational capabilities of the clients and the communication overhead.

In Fig.8c, the numbers of transmitted data symbols for CL, FL and HFCL are presented. The overhead of CL is due to the transmission of whole data symbols, i.e., approximately 4.5×10^9. In contrast, the complexity of FL is due to the two way (edge $\leftrightarrows$ server) transmission of the model updates during training until convergence, i.e., $2 \times 40 \times (2 \times 10^6) = 160 \times 10^6$ for 40 iterations. As a result, FL and HFCL have approximately 28 and 3 times lower communication overhead compared to CL, respectively. The effectiveness of HFCL in terms of overhead is reduced compared to the image classification application because of the larger input size of the object detection problem (i.e., $336\times336\times3$ versus $28\times28\times1$).

VIII. Summary

In this paper, we introduced a hybrid federated and centralized learning (HFCL) approach for distributed ML tasks. The proposed approach is helpful if a portion of the edge devices do not have the computational capability for model computation during training, which is quite common in practice. In order to train the learning model collaboratively, only the active devices, which have sufficient computational capability, perform FL by computing the model updates on their local datasets whereas the remaining inactive devices, which do not have enough computational power, resort to CL and send their local datasets to the PS, which performs the model computation on behalf of them. As a result, HFCL provides a novel solution achieving a trade-off between FL and CL in terms of computation capability of edge devices and communication overhead. Moreover, the transmission of local datasets may generate delays during training if the dataset size of the inactive clients is large. We suggested a mitigation of this problem by proposing HFCL-ICpC and HFCL-SDT frameworks. While the former improves the learning accuracy with a model computation-per-client, the latter reduces the size of the transmitted datasets. Compared to CL, the proposed approach is more advantageous due to the reduction in the communication latency. In the meantime, compared to FL, a drawback of the proposed HFCL approach is that HFCL requires the dataset transmission of the inactive clients, which may raise privacy concerns, however, these are already apparent in CL. Nevertheless, HFCL provides access to the dataset of all devices regardless of their computational capability whereas, in FL, only the dataset of the inactive clients could be used. We showed that our HFCL approach has a significant performance improvement as compared to FL-based training with only active clients since FL cannot access the dataset of all clients. As a future work, we plan to study the application of HFCL to physical layer applications, such as channel estimation, resource allocation, and beamforming. Furthermore, the noisy learning model may not represent the practical case, hence requires further research on realistic modeling.

Appendix A
Proof of Lemma 1

Using (12), we get

$$\begin{aligned}
&||\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta}) - \nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta}')|| \\
&= ||\nabla(\mathcal{F}_k(\boldsymbol{\theta}) + (\tilde{\sigma}^2+\sigma_k^2)||\nabla\mathcal{F}_k(\boldsymbol{\theta})||^2) \\
&\qquad - \nabla(\mathcal{F}_k(\boldsymbol{\theta}') + (\tilde{\sigma}^2+\sigma_k^2)||\nabla\mathcal{F}_k(\boldsymbol{\theta}')||^2)|| \\
&= ||\big(\nabla\mathcal{F}_k(\boldsymbol{\theta}) + (\tilde{\sigma}^2+\sigma_k^2)\nabla||\nabla\mathcal{F}_k(\boldsymbol{\theta})||^2\big) \\
&\qquad - \big(\nabla\mathcal{F}_k(\boldsymbol{\theta}') + (\tilde{\sigma}^2+\sigma_k^2)\nabla||\nabla\mathcal{F}_k(\boldsymbol{\theta}')||^2\big)|| \\
&= ||\nabla\mathcal{F}_k(\boldsymbol{\theta}) - \nabla\mathcal{F}_k(\boldsymbol{\theta}') + (\tilde{\sigma}^2+\sigma_k^2) \\
&\times \big(\nabla\mathrm{tr}\{\nabla\mathcal{F}_k(\boldsymbol{\theta})^\mathsf{T}\nabla\mathcal{F}_k(\boldsymbol{\theta})\} - \nabla\mathrm{tr}\{\nabla\mathcal{F}_k(\boldsymbol{\theta}')^\mathsf{T}\nabla\mathcal{F}_k(\boldsymbol{\theta}')\}\big)|| \\
&= ||\nabla\mathcal{F}_k(\boldsymbol{\theta}) - \nabla\mathcal{F}_k(\boldsymbol{\theta}') \\
&\qquad + (\tilde{\sigma}^2+\sigma_k^2)\big(\nabla\mathcal{F}_k(\boldsymbol{\theta}) - \nabla\mathcal{F}_k(\boldsymbol{\theta}')\big)|| \\
&= ||(1+\tilde{\sigma}^2+\sigma_k^2)\big(\nabla\mathcal{F}_k(\boldsymbol{\theta}) - \nabla\mathcal{F}_k(\boldsymbol{\theta}')\big)|| \\
&= (1+\tilde{\sigma}^2+\sigma_k^2)||\nabla\mathcal{F}_k(\boldsymbol{\theta}) - \nabla\mathcal{F}_k(\boldsymbol{\theta}')||. \qquad (25)
\end{aligned}$$

By incorporating (25), Assumption 2 and the fact that $1 + (\bar{\sigma}^2+\sigma_k^2) \geq 0$, we get

$$||\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta}) - \nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta}')|| \leq \tilde{\beta}||\boldsymbol{\theta}-\boldsymbol{\theta}'||^2, \qquad (26)$$

where $\bar{\beta} = (1+(\tilde{\sigma}^2+\sigma_k^2))\beta$.

Appendix B
Proof of Theorem 1

Using (26), Assumption 2 and Assumption 3 imply that $\bar{\mathcal{F}}_k(\boldsymbol{\theta})$ is second order differentiable as $\nabla^2\bar{\mathcal{F}}_k(\boldsymbol{\theta}) \preceq \bar{\beta}\mathbf{I}_P$. Using this fact, performing a quadratic expression around $\bar{\mathcal{F}}_k(\boldsymbol{\theta})$ yields

$$\begin{aligned}
\bar{\mathcal{F}}_k(\boldsymbol{\theta}') &\leq \bar{\mathcal{F}}_k(\boldsymbol{\theta}) + \nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})^\mathsf{T}(\boldsymbol{\theta}'-\boldsymbol{\theta}) + \frac{1}{2}\nabla^2\bar{\mathcal{F}}_k(\boldsymbol{\theta})||\boldsymbol{\theta}'-\boldsymbol{\theta}||^2 \\
&\leq \mathcal{F}_k(\boldsymbol{\theta}) + \nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})^\mathsf{T}(\boldsymbol{\theta}'-\boldsymbol{\theta}) + \frac{1}{2}\bar{\beta}||\boldsymbol{\theta}'-\boldsymbol{\theta}||^2. \qquad (27)
\end{aligned}$$

Substituting the GD update $\boldsymbol{\theta}' = \boldsymbol{\theta} - \eta\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})$ in (27), we get

$$\begin{aligned}
\bar{\mathcal{F}}_k(\boldsymbol{\theta}') &\leq \bar{\mathcal{F}}_k(\boldsymbol{\theta}) + \nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})^\mathsf{T}(\boldsymbol{\theta}'-\boldsymbol{\theta}) + \frac{1}{2}\bar{\beta}||\boldsymbol{\theta}'-\boldsymbol{\theta}||^2 \\
&= \bar{\mathcal{F}}_k(\boldsymbol{\theta}) + \nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})^\mathsf{T}(\boldsymbol{\theta} - \eta\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta}) - \boldsymbol{\theta}) \\
&\quad + \frac{1}{2}\nabla^2\bar{\mathcal{F}}_k(\boldsymbol{\theta})||\boldsymbol{\theta} - \eta\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta}) - \boldsymbol{\theta}||^2 \\
&= \bar{\mathcal{F}}_k(\boldsymbol{\theta}) - \eta\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})^\mathsf{T}\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta}) + \frac{1}{2}\bar{\beta}||\eta\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})||^2 \\
&= \bar{\mathcal{F}}_k(\boldsymbol{\theta}) - \eta||\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})||^2 + \frac{1}{2}\bar{\beta}\eta^2||\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})||^2 \\
&= \bar{\mathcal{F}}_k(\boldsymbol{\theta}) - (1-\frac{\bar{\beta}\eta}{2})\eta||\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})||^2, \qquad (28)
\end{aligned}$$

which bounds the GD update $\bar{\mathcal{F}}_k(\boldsymbol{\theta}')$ with $\bar{\mathcal{F}}_k(\boldsymbol{\theta})$. Now, let us bound $\bar{\mathcal{F}}_k(\boldsymbol{\theta}')$ with the optimal objective value $\bar{\mathcal{F}}_k(\boldsymbol{\theta}^\star)$. Using Assumption 1, we have

$$\begin{aligned}
\bar{\mathcal{F}}_k(\boldsymbol{\theta}^\star) &\geq \bar{\mathcal{F}}_k(\boldsymbol{\theta}) + \nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})^\mathsf{T}(\boldsymbol{\theta}^\star - \boldsymbol{\theta}), \\
\bar{\mathcal{F}}_k(\boldsymbol{\theta}) &\leq \bar{\mathcal{F}}_k(\boldsymbol{\theta}^\star) + \nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})^\mathsf{T}(\boldsymbol{\theta} - \boldsymbol{\theta}^\star). \qquad (29)
\end{aligned}$$

Furthermore, using $\eta \leq \frac{1}{\bar{\beta}}$, we have $-(1-\frac{\bar{\beta}\eta}{2}) = \frac{1}{2}\bar{\beta}\eta - 1 \leq \frac{1}{2}\bar{\beta}(1/\bar{\beta}) - 1 = \frac{1}{2} - 1 = -\frac{1}{2}$. Thus, (28) becomes

$$\bar{\mathcal{F}}_k(\boldsymbol{\theta}') \leq \bar{\mathcal{F}}_k(\boldsymbol{\theta}) - \frac{\eta}{2}||\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})||^2 \tag{30}$$

By plugging (29) into (30), we get

$$\bar{\mathcal{F}}_k(\boldsymbol{\theta}') \leq \bar{\mathcal{F}}_k(\boldsymbol{\theta}^\star) + \nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})^\mathsf{T}(\boldsymbol{\theta} - \boldsymbol{\theta}^\star) - \frac{\eta}{2}||\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})||^2, \tag{31}$$

which can be rewritten as

$$\bar{\mathcal{F}}_k(\boldsymbol{\theta}') - \bar{\mathcal{F}}_k(\boldsymbol{\theta}^\star) \leq \frac{1}{2\eta}\left(2\eta\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})^\mathsf{T}(\boldsymbol{\theta} - \boldsymbol{\theta}^\star) - \eta^2||\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})||^2\right). \tag{32}$$

By adding $\frac{1}{2\eta}(||\boldsymbol{\theta} - \boldsymbol{\theta}^\star||^2 - ||\boldsymbol{\theta} - \boldsymbol{\theta}^\star||^2)$ into the right hand side of (32), we get

$$\bar{\mathcal{F}}_k(\boldsymbol{\theta}') - \bar{\mathcal{F}}_k(\boldsymbol{\theta}^\star) \leq \frac{1}{2\eta}\left(||\boldsymbol{\theta} - \boldsymbol{\theta}^\star||^2 - ||\boldsymbol{\theta} - \boldsymbol{\theta}^\star - \eta\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})||^2\right), \tag{33}$$

which is obtained after incorporating the expansion of $||\boldsymbol{\theta} - \boldsymbol{\theta}^\star - \eta\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})||^2$. Substituting the GD update $\boldsymbol{\theta}' = \boldsymbol{\theta} - \eta\nabla\bar{\mathcal{F}}_k(\boldsymbol{\theta})$ into (33), we have

$$\bar{\mathcal{F}}_k(\boldsymbol{\theta}') - \bar{\mathcal{F}}_k(\boldsymbol{\theta}^\star) \leq \frac{1}{2\eta}\left(||\boldsymbol{\theta} - \boldsymbol{\theta}^\star||^2 - ||\boldsymbol{\theta}' - \boldsymbol{\theta}^\star||^2\right). \tag{34}$$

Now, let us replace $\boldsymbol{\theta}'$ with $\boldsymbol{\theta}^{(i)}$, then summing over $i = 1, \ldots, t$ yields

$$\begin{aligned}
&\sum_{i=1}^{t}(\bar{\mathcal{F}}_k(\boldsymbol{\theta}^{(i)}) - \bar{\mathcal{F}}_k(\boldsymbol{\theta}^\star)) \\
&\leq \sum_{i=1}^{t}\frac{1}{2\eta}\left(||\boldsymbol{\theta}^{(i-1)} - \boldsymbol{\theta}^\star||^2 - ||\boldsymbol{\theta}^{(i)} - \boldsymbol{\theta}^\star||^2\right) \\
&= \frac{1}{2\eta}\left(||\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^\star||^2 - ||\boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}^\star||^2\right) \\
&\leq \frac{1}{2\eta}||\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^\star||^2,
\end{aligned} \tag{35}$$

where the summation on the right hand side disappears since the consecutive terms cancel each other. Since $\bar{\mathcal{F}}_k(\boldsymbol{\theta}^{(t)})$ is a decreasing function, we have

$$\bar{\mathcal{F}}_k(\boldsymbol{\theta}^{(t)}) - \bar{\mathcal{F}}_k(\boldsymbol{\theta}^\star) \leq \frac{1}{t}\sum_{i=1}^{t}(\bar{\mathcal{F}}_k(\boldsymbol{\theta}^{(i)}) - \bar{\mathcal{F}}_k(\boldsymbol{\theta}^\star)). \tag{36}$$

Inserting (35) into (36), we finally have

$$\bar{\mathcal{F}}_k(\boldsymbol{\theta}^{(t)}) - \bar{\mathcal{F}}_k(\boldsymbol{\theta}^\star) \leq \frac{1}{2\eta t}||\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^\star||^2. \tag{37}$$

REFERENCES

[1] A. M. Elbir, S. Coleri, and K. V. Mishra, "Hybrid Federated and Centralized Learning," in *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE, Aug 2021, pp. 1541–1545.

[2] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] R. Mayer and H.-A. Jacobsen, "Scalable Deep Learning on Distributed Infrastructures: Challenges, Techniques, and Tools," *ACM Comput. Surv.*, vol. 53, no. 1, pp. 1–37, Feb 2020.

[4] H. Ye, L. Liang, G. Ye Li, J. Kim, L. Lu, and M. Wu, "Machine Learning for Vehicular Networks: Recent Advances and Application Examples," *IEEE Veh. Technol. Mag.*, vol. 13, no. 2, pp. 94–101, 2018.

[5] "Machine learning for internet of things data analysis: a survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161 – 175, 2018.

[6] A. M. Elbir, B. Soner, and S. Coleri, "Federated Learning in Vehicular Networks," *arXiv*, Jun 2020. [Online]. Available: https://arxiv.org/abs/2006.01412v2

[7] A. M. Elbir and K. V. Mishra, "Cognitive Learning-Aided Multi-Antenna Communications," *arXiv preprint arXiv:2010.03131*, 2020.

[8] A. M. Elbir and K. V. Mishra, "A Survey of Deep Learning Architectures for Intelligent Reflecting Surfaces," *arXiv*, Sep 2020. [Online]. Available: https://arxiv.org/abs/2009.02540v2

[9] O. Simeone, "A Very Brief Introduction to Machine Learning With Applications to Communication Systems," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 648–664, Dec 2018.

[10] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless Network Intelligence at the Edge," *Proc. IEEE*, vol. 107, no. 11, pp. 2204–2239, Nov 2019.

[11] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, 2020.

[12] M. Mohammadi Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *IEEE Trans. Signal Process.*, vol. 68, pp. 2155–2169, 2020.

[13] M. M. Amiri and D. Gündüz, "Federated Learning Over Wireless Fading Channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3546–3557, 2020.

[14] A. M. Elbir and K. V. Mishra, "Joint antenna selection and hybrid beamformer design using unquantized and quantized deep learning networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1677–1688, March 2020.

[15] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," *arXiv*, Feb 2016. [Online]. Available: https://arxiv.org/abs/1602.05629v3

[16] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated Machine Learning: Concept and Applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, Jan 2019.

[17] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the Convergence of FedAvg on Non-IID Data," in *International Conference on Learning Representations*, 2020.

[18] D. Guliani, F. Beaufays, and G. Motta, "Training Speech Recognition Models with Federated Learning: A Quality/Cost Framework," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Jun. 2021, pp. 3080–3084.

[19] A. M. Elbir and S. Coleri, "Federated Learning for Hybrid Beamforming in mm-Wave Massive MIMO," *IEEE Commun. Lett.*, vol. 24, no. 12, pp. 2795–2799, Aug 2020.

[20] A. M. Elbir and S. Coleri, "Federated Learning for Channel Estimation in Conventional and RIS-Assisted Massive MIMO," *IEEE Trans. Wireless Commun.*, p. 1, Nov 2021.

[21] D. Ma, L. Li, H. Ren, D. Wang, X. Li, and Z. Han, "Distributed Rate Optimization for Intelligent Reflecting Surface with Federated Learning," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2020, pp. 1–6.

[22] M. M. Wadu, S. Samarakoon, and M. Bennis, "Federated learning under channel uncertainty: Joint client scheduling and resource allocation," *arXiv preprint arXiv:2002.00802*, 2020.

[23] T. Zeng, O. Semiari, M. Mozaffari, M. Chen, W. Saad, and M. Bennis, "Federated Learning in the Sky: Joint Power Allocation and Scheduling with UAV Swarms," *arXiv preprint arXiv:2002.08196*, 2020.

[24] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated Learning for Internet of Things: Recent Advances, Taxonomy, and Open Challenges," *IEEE Commun. Surv. Tutorials*, vol. 23, no. 3, pp. 1759–1799, Jun. 2021.

[25] A. M. Elbir, A. K. Papazafeiropoulos, and S. Chatzinotas, "Federated Learning for Physical Layer Design," *IEEE Commun. Mag.*, vol. 59, no. 11, pp. 81–87, Dec 2021.

[26] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23 920–23 935, 2020.

[27] T. Nishio and R. Yonetani, "Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[28] G. Shi, L. Li, J. Wang, W. Chen, K. Ye, and C. Xu, "HySync: Hybrid Federated Learning with Effective Synchronization," in *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, Dec 2020, pp. 628–633.

[29] X. Zhang, W. Yin, M. Hong, and T. Chen, "Hybrid Federated Learning: Algorithms and Implementation," *arXiv*, Dec 2020. [Online]. Available: https://arxiv.org/abs/2012.12420v3

[30] A. Huang, Y. Liu, T. Chen, Y. Zhou, Q. Sun, H. Chai, and Q. Yang, "StarFL: Hybrid Federated Learning Architecture for Smart Urban Computing," *ACM Trans. Intell. Syst. Technol.*, vol. 12, no. 4, pp. 1–23, Aug 2021.

[31] 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, Mar 2020.

[32] F. Haddadpour and M. Mahdavi, "On the Convergence of Local Descent Methods in Federated Learning," *arXiv*, Oct 2019. [Online]. Available: https://arxiv.org/abs/1910.14425v2

[33] G. Zhu, Y. Du, D. Gündüz, and K. Huang, "One-Bit Over-the-Air Aggregation for Communication-Efficient Federated Edge Learning: Design and Convergence Analysis," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 2120–2135, Nov 2020.

[34] C. M. Bishop, "Training with Noise is Equivalent to Tikhonov Regularization," *Neural Comput.*, vol. 7, no. 1, pp. 108–116, Jan 1995.

[35] X. Wei and C. Shen, "Federated Learning over Noisy Channels," in *ICC 2021 - IEEE International Conference on Communications*. IEEE, Jun. 2021, pp. 1–6.

[36] S. Luo, X. Chen, Q. Wu, Z. Zhou, and S. Yu, "HFEL: Joint Edge Association and Resource Allocation for Cost-Efficient Hierarchical Federated Edge Learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6535–6548, Jun. 2020.

[37] W. Y. B. Lim, J. S. Ng, Z. Xiong, J. Jin, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Decentralized Edge Intelligence: A Dynamic Resource Allocation Framework for Hierarchical Federated Learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 536–550, Jul. 2021.

[38] M. Li, T. Zhang, Y. Chen, and A. J. Smola, *Efficient mini-batch training for stochastic optimization*. New York, NY, USA: Association for Computing Machinery, Aug 2014.

[39] S. Zhou and G. Y. Li, "Communication-Efficient ADMM-based Federated Learning," *arXiv*, Oct 2021. [Online]. Available: https://arxiv.org/abs/2110.15318v3

[40] S. Sra, S. Nowozin, and S. J. Wright, *Optimization for Machine Learning*. Cambridge, MA, USA: The MIT Press, Sep 2011.

[41] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[42] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet, "Lyft Level 5 AV Dataset 2019," 2019; https://level5.lyft.com/dataset/, accessed 1 Jun. 2020.

[43] W. Shi, S. Zhou, Z. Niu, M. Jiang, and L. Geng, "Joint Device Scheduling and Resource Allocation for Latency Constrained Wireless Federated Learning," *IEEE Trans. Wireless Commun.*, p. 1, Sep 2020.

[44] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 429–450, 2020.

[45] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.