Hybrid Federated and Centralized Learning

Ahmet M. Elbir[1], Sinem Coleri[1] and Kumar Vijay Mishra[2]
[1] Department of Electrical and Electronics Engineering, Koc University, Istanbul, Turkey
[2] United States CCDC Army Research Laboratory, Adelphi, MD 20783 USA
E-mail: ahmetmelbir@gmail.com, scoleri@ku.edu.tr, kumarvijay-mishra@uiowa.edu

***Abstract*—Many of the machine learning (ML) tasks are focused on centralized learning (CL), which requires the transmission of local datasets from the clients to a parameter server (PS) leading to a huge communication overhead. Federated learning (FL) overcomes this issue by allowing the clients to send only the model updates to the PS instead of the whole dataset. In this way, FL brings the learning to edge level, wherein powerful computational resources are required on the client side. This requirement may not always be satisfied because of diverse computational capabilities of edge devices. We address this through a novel hybrid federated and centralized learning (HFCL) framework to effectively train a learning model by exploiting the computational capability of the clients. In HFCL, only the clients who have sufficient resources employ FL; the remaining clients resort to CL by transmitting their local dataset to PS. This allows all the clients to collaborate on the learning process regardless of their computational resources. We also propose a sequential data transmission approach with HFCL (HFCL-SDT) to reduce the training duration. The proposed HFCL frameworks outperform previously proposed non-hybrid FL (CL) based schemes in terms of learning accuracy (communication overhead) since all the clients collaborate on the learning process with their datasets regardless of their computational resources.**

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

I. Introduction

Machine learning (ML) has emerged as a promising engineering for future technologies such as internet of things (IoT), autonomous driving and next-generation wireless communications [1, 2]. These applications require massive data processing and abstraction 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 model-free data mapping with huge number of learnable parameters [3, 4]. The implementation of these learning models demands powerful computational 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 of the success of ML algorithms [5].

Implementations of ML usually focus on centralized learning (CL) algorithms, where a powerful ANN is trained at a parameter server (PS) [1, 6, 7]. This inherently assumes the availability of data at the PS. In case of wireless edge devices (clients), transmitting the collected data to the PS in a reliable manner is expensive in terms of energy and bandwidth, thereby introducing delays and possible infringements of client privacy [8]. For example, in long-term evolution (LTE) networks, a single frame of 5 MHz bandwidth and 10 ms duration may carry only 6000 complex symbols [9], whereas the size of the whole dataset could be hundreds of thousands symbols [10]. As a result, CL-based techniques require huge bandwidths and communications overhead during training.

As 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 [7, 11]. In FL, rather than transmitting local datasets to the PS, the clients send the model updates (gradients) to collaboratively train the learning model. The collected model updates are aggregated at the PS and then sent back to the clients to further update the learning parameters iteratively. Compared to CL, FL provides less communications overhead with a slight prediction loss because of insufficient gradient components and data corruptions during wireless gradient transmission.

Recently, FL has been applied to wireless communications [12, 13], including architectures such as cellular networks [9, 14, 15], vehicular networks [16], unmanned aerial vehicles [17] and IoT networks [18]. Here, FL architectures relied on the fact that all of the clients were capable of gradient computation, often using powerful parallel processing units. However, in practice, considering the diversity of the devices with different computational capabilities - such as mobile phones, vehicular components and IoT devices - this requirement cannot be met. A possible solution lies in a hybrid learning technique that benefits from both CL and FL. Here, the devices incapable of sufficient computation power deploy CL while the rest use FL. To this end, the client selection algorithms proposed in [19, 20] do not involve a collaboration of all the devices; instead, only the trusted clients [19] or the ones with sufficient computational resources [20] are selected for FL-based training.

In this paper, we introduce a hybrid FL and CL (HFCL) framework to effectively train a learning model where the edge devices collaborate on the learning process by exploiting their computational capabilities (Fig. 1). In the beginning, we designate the clients as *passive* (CL) or *active* (FL) depending on their computational power. Then, the active clients transmit the gradient information to the PS based on their local dataset. On the other hand, the passive clients transmit their dataset to the PS, which computes the corresponding gradients data on behalf of them. At the PS, the computed gradients are utilized to aggregate the model parameters, which are then sent back to active devices. The challenge for HFCL is the wait

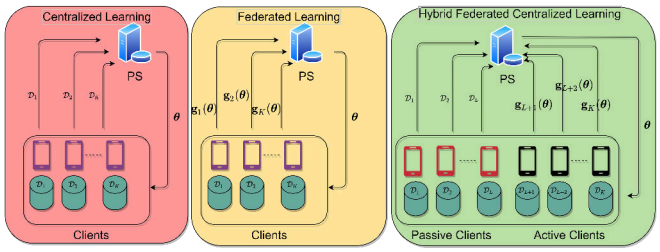


Fig. 1. In a CL framework (left), all clients transmit their datasets to the PS. On the other hand, the datasets are preserved at the clients in FL (center) and only model parameters are sent to the PS. In HFCL (right), the clients are grouped into active and passive depending on their computational capability to either perform CL and FL, respectively.

required by the active clients when the passive clients complete their data set transmission at the beginning of the training. To mitigate this problem, we propose a sequential dataset transmission (SDT) approach where the passive clients do not send the entire local dataset at once. Rather the local dataset is divided into smaller blocks so that both active/passive devices perform gradient/data transmission during the same communication exchange. We evaluate the performance of the proposed approaches on MNIST dataset [21] and show that both techniques provide higher learning performance than FL while introducing a slight increase in the communication overhead arising from dataset transmission. However, this overhead is still less than that of CL.

Notation: Throughout the paper, we denote the vectors and matrices by boldface lower and upper case symbols, respectively. In case of a vector $\mathbf{a}$, $[\mathbf{a}]_i$ represents its i-th element. For a matrix $\mathbf{A}$, $[\mathbf{A}]_{i,j}$ denotes the (i,j)-th entry. The $\mathbf{I}_N$ is the identity matrix of size $N \times N$; $\|\cdot\|_{\mathcal{F}}$ is the Frobenius norm; The notation expressing a convolutional layer with N filters/channels of size $D \times D$ is given by $N@D \times D$.

II. Desiderata on CL and FL

Consider a scenario, wherein K edge devices collaborate on solving an optimization problem through ML. Define $\mathcal{D}_k$ to be the local dataset of the k-th client so that the whole dataset is $\mathcal{D} = \bigcup_{k\in\mathcal{K}} \mathcal{D}_k$, where $\mathcal{K} = \{1, \ldots, K\}$. Denote the input and output tuple $\mathcal{D}_i$ of the i-th element of the dataset $\mathcal{D}$ by $\mathcal{X}_i$ and $\mathcal{Y}_i$, respectively, such that $\mathcal{D}_i = (\mathcal{X}_i, \mathcal{Y}_i)$ and $\mathsf{D} = |\mathcal{D}|$ is the number of input-output pairs. The CL-based training considers a learning model parameterized by $\boldsymbol{\theta} \in \mathbb{R}^P$ and optimized via

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \mathcal{F}(\boldsymbol{\theta}) = \frac{1}{\mathsf{D}} \sum_{i=1}^{\mathsf{D}} \mathcal{J}(f(\mathcal{X}_i|\boldsymbol{\theta}), \mathcal{Y}_i), \tag{1}$$

where the non-linear mapping between the input and output is constructed via $f(\mathcal{X}_i|\boldsymbol{\theta})$ by minimizing the empirical loss $\mathcal{F}(\boldsymbol{\theta})$ over the learnable parameters $\boldsymbol{\theta}$ and the loss function $\mathcal{J}(\cdot)$ is the mean-squared-error (MSE) between the label data $\mathcal{Y}_i$ and the prediction $f(\mathcal{X}_i|\boldsymbol{\theta})$ of the learning model over the whole dataset, i.e.,

$$\mathcal{J}(f(\mathcal{X}_i|\boldsymbol{\theta}), \mathcal{Y}_i) = \|f(\mathcal{X}_i|\boldsymbol{\theta}) - \mathcal{Y}_i\|_{\mathcal{F}}^2. \tag{2}$$

In CL, (1) is solved at a PS, which collects the local datasets $\mathcal{D}_k, k \in \mathcal{K}$ from the clients.

On the other hand, transmission of datasets is avoided in FL by considering the following optimization problem

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \frac{1}{K} \sum_{k=1}^{K} \mathcal{F}_k(\boldsymbol{\theta}),$$
$$\text{subject to: } \mathcal{F}_k(\boldsymbol{\theta}) = \frac{1}{\mathsf{D}_k} \sum_{i=1}^{\mathsf{D}_k} \mathcal{J}(f(\mathcal{X}_i^{(k)}|\boldsymbol{\theta}), \mathcal{Y}_i^{(k)}), \tag{3}$$

for which the learning model is trained over the local datasets $\mathcal{D}_k$ with input-output pair $(\mathcal{X}_i^{(k)}, \mathcal{Y}_i^{(k)})$ and $\mathsf{D}_k = |\mathcal{D}_k|$.

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 of the clients can contribute to the learning task with their datasets regardless of their ability to compute model parameters. In ML tasks, training a model requires huge computational power to compute the model parameters. This requirement cannot always be satisfied by the computational capability of the client devices. In order to train the ML model effectively taking into account the computational capability of the clients, a hybrid training framework is introduced in this work. We assume that only a portion of the clients with sufficient computational power performs FL, while the remaining clients, which suffer from computational capability, send their datasets to the PS for model computation, as illustrated in Fig. 1.

In order to train the learning model, the minimization of $\mathcal{F}(\boldsymbol{\theta})$ is carried out iteratively through gradient descent (GD). Denote the model parameters at the t-th communication round/iteration as $\boldsymbol{\theta}_t$, $t = 1, \ldots, T$, where T denotes the number of iterations to reach the convergence. Then, the $(t+1)$-th iteration of GD yields

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \mathbf{g}(\boldsymbol{\theta}_t), \tag{4}$$

where η_t is the learning rate and

$$\mathbf{g}(\boldsymbol{\theta}_t) = \nabla_{\boldsymbol{\theta}} \mathcal{F}(\boldsymbol{\theta}_t) = \frac{1}{\mathsf{D}} \sum_{i=1}^{\mathsf{D}} \nabla_{\boldsymbol{\theta}} \mathcal{J}(f(\mathcal{X}_i|\boldsymbol{\theta}_t), \mathcal{Y}_i), \tag{5}$$

denotes the *full* or *batch* gradient vector in $\mathbb{R}^P$, where P is the total number of learnable model parameters.

For large datasets, GD is computationally inefficient. A better alternative is stochastic GD (SGD), where $\mathcal{D}$ is partitioned into M_B mini-batches as $\mathcal{D} = \bigcup_{m\in\mathcal{M}_B} \mathcal{D}_m$, for $\mathcal{M}_B = \{1, \ldots, M_B\}$ [22]. Then, $\boldsymbol{\theta}_t$ is updated by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \mathbf{g}_{\mathcal{M}_B}(\boldsymbol{\theta}_t), \tag{6}$$

where $\mathbf{g}_{\mathcal{M}_B}(\boldsymbol{\theta}_t) = \frac{1}{M_B} \sum_{m=1}^{M_B} \mathbf{g}_m(\boldsymbol{\theta}_t)$ includes the contribution of gradients computed over $\{\mathcal{D}_m\}_{m\in\mathcal{M}_B}$ as

$$\mathbf{g}_m(\boldsymbol{\theta}_t) = \frac{1}{\mathsf{D}_m} \sum_{i=1}^{\mathsf{D}_m} \nabla_{\boldsymbol{\theta}} \mathcal{J}(f(\mathcal{X}_{m,i}|\boldsymbol{\theta}_t), \mathcal{Y}_{m,i}), \tag{7}$$

where $(\mathcal{X}_{m,i}, \mathcal{Y}_{m,i})$ denotes the i-th input-output pair for the m-th mini-batch and $\mathsf{D}_m = |\mathcal{D}_m|$ is the mini-batch size.

The gradient term $\mathbf{g}_{\mathcal{M}_B}(\boldsymbol{\theta}_t)$ satisfies $\mathbb{E}\{\mathbf{g}_{\mathcal{M}_B}(\boldsymbol{\theta}_t)\} = \nabla_{\boldsymbol{\theta}}\mathcal{F}(\boldsymbol{\theta}_t)$ and, therefore, SGD provides the minimization of the empirical loss by partitioning the dataset into M_B mini-batches and accelerates the learning process, which is known as mini-batch learning [22]. Employing SGD in parallel among several devices allows us to compute the gradients on devices and aggregate them in the PS, which is known as FL [8].

In our HFCL approach, denote the group of active and passive clients that employ gradient computation at the PS and device levels as the index sets $\mathcal{L} = \{1, \ldots, L\}$ and $\bar{\mathcal{L}} = \mathcal{K} \setminus \mathcal{L} = \{L+1, \ldots, K\}$, respectively. Then, (4) becomes

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \Bigg(\underbrace{\frac{1}{L}\sum_{k\in\mathcal{L}} \mathbf{g}_k(\boldsymbol{\theta}_t)}_{\text{On Server}} + \underbrace{\frac{1}{K-L}\sum_{k\in\bar{\mathcal{L}}} \bar{\mathbf{g}}_k(\boldsymbol{\theta}_t)}_{\text{On Device}} \Bigg), \quad (8)$$

the computation of $\mathbf{g}_{k\in\mathcal{L}}(\boldsymbol{\theta}_t)$ is done via mini-batch learning as $\mathbf{g}_{k\in\mathcal{L}}(\boldsymbol{\theta}_t) = \frac{1}{M_B}\sum_{m=1}^{M_B} \mathbf{g}_{m,k\in\mathcal{L}}(\boldsymbol{\theta}_t)$ where $\mathbf{g}_{m,k\in\mathcal{L}}(\boldsymbol{\theta}_t) = \nabla_{\boldsymbol{\theta}}\mathcal{J}(f(\mathcal{X}_{m,i}^{(k)}|\boldsymbol{\theta}_t), \mathcal{Y}_{m,i}^{(k)})$ considering that the PS has access to the dataset $\mathcal{D}_{k\in\mathcal{L}}$. The gradients corresponding to the active clients $\bar{\mathbf{g}}_{k\in\bar{\mathcal{L}}}$ are collected at the PS through a noisy wireless channel as

$$\bar{\mathbf{g}}_{k\in\bar{\mathcal{L}}}(\boldsymbol{\theta}_t) = Q_B(\mathbf{g}_{k\in\bar{\mathcal{L}}}(\boldsymbol{\theta}_t)) + \mathbf{w}_{k\in\bar{\mathcal{L}},t}, \quad (9)$$

where $Q_B(\cdot)$ represents the quantization operator with B bits and $\mathbf{w}_{k\in\bar{\mathcal{L}},t} \in \mathbb{R}^P$ denotes the noise term added onto $Q_B(\mathbf{g}_{k\in\bar{\mathcal{L}}}(\boldsymbol{\theta}_t))$ at the t-th iteration.

Without loss of generality, we assume that $\mathbf{w}_{k\in\bar{\mathcal{L}},t}$ obeys normal distribution, i.e., $\mathbf{w}_{k\in\bar{\mathcal{L}},t} \sim \mathcal{N}(0, \sigma_{\boldsymbol{\theta}}^2\mathbf{I}_P)$ with variance $\sigma_{\boldsymbol{\theta}}^2$ and the signal-to-noise-ratio (SNR) in gradient transmission is given by $\mathrm{SNR}_{\boldsymbol{\theta}} = 20\log_{10}\frac{||\mathbf{g}_{k\in\mathcal{L}}(\boldsymbol{\theta}_t)||_2^2}{\sigma_{\boldsymbol{\theta}}^2}$. The same amount of noise is also added onto $\mathcal{D}_{k\in\mathcal{L}}$ during dataset transmission for passive clients. Once the model aggregation is completed during model training, the PS returns the updated model parameters $\boldsymbol{\theta}_{t+1}$ to only the active clients.

A. Communication Delay During Model Training

The bandwidth resources need to be optimized to reduce the latency of the transmission of both $\mathbf{g}_k(\boldsymbol{\theta})$ ($k \in \bar{\mathcal{L}}$) 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 updates ($\mathbf{g}_k(\boldsymbol{\theta})$ for $k \in \bar{\mathcal{L}}$), and can defined as $\tau_k = \frac{d_k}{R_k}$, 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, for which B_k and SNR_k, respectively, denote the allocated bandwidth and the signal-to-noise ratio (SNR) for the kth client. The PS solves $\min_{B_k} \max_{k\in\mathcal{K}} \quad \tau_k$ 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 passive (i.e., $k \in \mathcal{L}$) and active (i.e., $k \in \bar{\mathcal{L}}$) clients [23]. Depending on the client type, d_k can be given by

$$d_k = \begin{cases} P, & k \in \bar{\mathcal{L}} \\ \mathsf{d}_k, & k \in \mathcal{L} \end{cases}, \quad (10)$$

which is fixed to the number of model parameters P for the active clients and to $\mathsf{d}_k = D_k(U_xV_x + U_yV_y)$ 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., $\mathsf{d}_{k\in\mathcal{L}} > P$ [7, 8, 12], the dataset transmission of the passive clients is expected to take longer than the model transmission of the active clients, i.e., $\tau_{k\in\mathcal{L}} > \tau_{k\in\bar{\mathcal{L}}}$ [23]. Previous FL-based works reported that $\tau_{k\in\mathcal{L}}$ can be approximately 10 times longer than $\tau_{k\in\bar{\mathcal{L}}}$ [12, 13]. This introduces a significant delay especially at the beginning of the training. Because, the HFCL problem in (8) can be performed only if $\mathcal{D}_{k\in\mathcal{L}}$ is collected at the PS for the first iteration. To circumvent this issue and keep the training continue, we propose the SDT approach in the following.

B. HFCL With Sequential Data Transmission

The HFCL approach described above performs model aggregation only after the dataset transmission for $k \in \mathcal{L}$ is completed when $t = 1$. This causes delays during model training. In order to circumvent this problem, we propose HFCL-SDT that partitions the local dataset of the passive clients into small blocks and transmits them to the PS sequentially without any delay. A conventional way is to partition the dataset $\mathsf{D}_{k\in\mathcal{L}}$ into $N = \frac{\mathsf{D}_k}{P}$ blocks, where the size of the transmitted symbols for each communication round is equal to P for both passive and active clients. For non-integer N, $N = \lceil \frac{\mathsf{D}_k}{P} \rceil$.

Then, compute $\mathbf{g}_{k\in\mathcal{L}}(\boldsymbol{\theta}_t) = \frac{1}{M_B}\sum_{m=1}^{M_B} \mathbf{g}_{m,k\in\mathcal{L}}(\boldsymbol{\theta}_t)$, where

$$\mathbf{g}_{m,k\in\mathcal{L}}(\boldsymbol{\theta}_t) = \begin{cases} \frac{1}{tP}\sum_{i=1}^{tP} \nabla_{\boldsymbol{\theta}}\mathcal{J}(f(\mathcal{X}_{m,i}^{(k)}|\boldsymbol{\theta}_t), \mathcal{Y}_{m,i}^{(k)}), t \le N \\ \frac{1}{\mathsf{D}_k}\sum_{i=1}^{\mathsf{D}_k} \nabla_{\boldsymbol{\theta}}\mathcal{J}(f(\mathcal{X}_{m,i}^{(k)}|\boldsymbol{\theta}_t), \mathcal{Y}_{m,i}^{(k)}), t > N \end{cases}. \quad (11)$$

Note that the size of the training dataset for $\mathbf{g}_{k\in\mathcal{L}}(\boldsymbol{\theta}_t)$ grows as $t \to N$. It is, however, constant for $t > N$ when the size of the transmitted dataset is fixed at P because the collected blocks of the dataset are stored at the PS for $t \le N$. Consequently, HFCL-SDT mitigates transmission delay but exhibits the same communication overhead as HFCL. Furthermore, HFCL-SDT performs better than HFCL because smaller datasets imply quick learning of the features in the data at the beginning of the training.

Communication overhead is measured by the number of transmitted symbols during model training [3, 7, 8, 12]. For CL ($\mathcal{T}_{\mathrm{CL}}$), this overhead is the number of symbols used to transmit datasets. The same for FL ($\mathcal{T}_{\mathrm{FL}}$) is proportional to the number of communication rounds T and model parameters P. Denote $\bar{\mathsf{D}} = \sum_k \bar{\mathsf{D}}_k$ as the number of symbols of the whole dataset. The communication overheads of CL, FL and HFCL are, respectively,

$$\mathcal{T}_{\mathrm{CL}} = \bar{\mathsf{D}}, \quad (12)$$

$$\mathcal{T}_{\mathrm{FL}} = 2TPK, \quad (13)$$

$$\mathcal{T}_{\mathrm{HFCL}} = L\bar{\mathsf{D}}_{k\in\mathcal{L}} + 2TP(K-L), \quad (14)$$

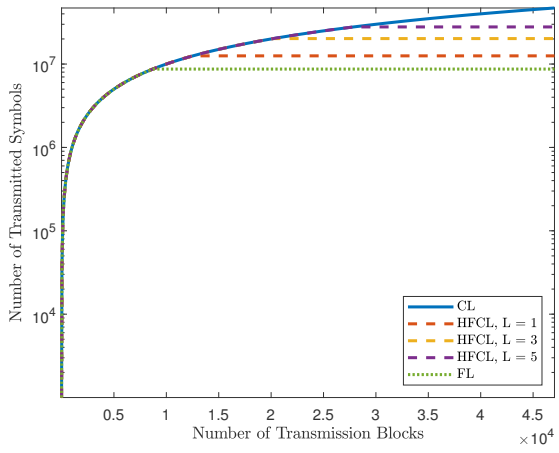


Fig. 2. Communication overhead comparison.

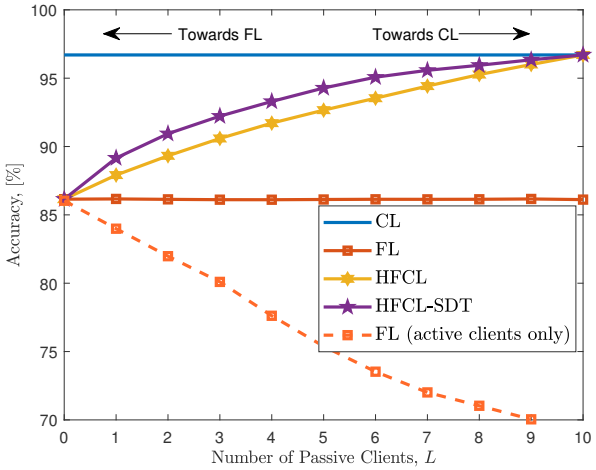


Fig. 3. Classification accuracy versus L when $\mathrm{SNR}_{\boldsymbol{\theta}} = 20$ dB and $B = 5$.

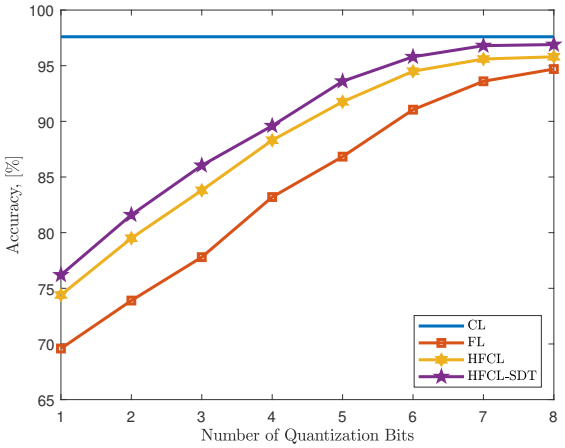


Fig. 4. Classification accuracy versus B when $\mathrm{SNR}_{\boldsymbol{\theta}} = 20$ dB.

where $\mathcal{T}_{\mathrm{HFCL}}$ includes the transmission of dataset of L passive clients and gradients of $K - L$ active clients. In general, FL has lower communication overhead than CL [3, 7, 8, 12, 13]. Hence, it follows that $\mathcal{T}_{\mathrm{FL}} \leq \mathcal{T}_{\mathrm{HFCL}} \leq \mathcal{T}_{\mathrm{CL}}$.

IV. Numerical Simulations

We evaluated the performance of our HFCL approach using the MNIST dataset [21] comprising 28×28 gray-scale images of handwritten digits with 10 classes. The number of symbols on the whole dataset is $\bar{\mathsf{D}} = 28^2 \cdot 60,000 \approx 47 \times 10^6$. During model training, the dataset is partitioned into $K = 10$ blocks, each of which is available at the clients that are independently and identically distributed. Further, we train a CNN with two convolutional layers with 5×5@128 and 3×3@128 spatial filters. Thus, we have $P = 128(5^2+3^2) = 4,352$. The validation data of MNIST dataset includes $10,000$ images and it is used for performance comparison for the competing algorithms. The learning rate is selected as 0.001 and the mini-batch size is 128 for CL. The loss function was the cross-entropy cost as $-\frac{1}{\bar{\mathsf{D}}} \sum_{i=1}^{\bar{\mathsf{D}}} \sum_{c=1}^{\bar{C}} \left[y_i^{(c)} \ln \hat{y}_i^{(c)} + (1-y_i^{(c)}) \ln(1-\hat{y}_i^{(c)}) \right]$, where $\{y_i^{(c)}, \hat{y}_i^{(c)}\}_{i=1,c=1}^{\bar{\mathsf{D}},\bar{C}}$ is the true and predicted response for the classification layer with $\bar{C} = 10$ and the classification accuracy is $\mathrm{Accuracy}(\%) = \frac{\mathsf{U}}{\bar{\mathsf{D}}} \times 100$, in which the model identified the image class correctly U times.

Fig. 2 depicts the communication overhead of CL, FL and HFCL for $L = \{0, 1, 3, 5, 7, 10\}$. During model training, 1000 data symbols are transmitted at each transmission block. It takes approximately 47×10^3 transmission blocks to complete CL-based training while FL demands approximately 8.5×10^3 data blocks (approximately 6 times lower than that of CL). The communication overhead of HFCL (as well as HFCL-SDT) lies between CL and FL because it depends on the number of passive clients L and approaches to $\mathcal{T}_{\mathrm{CL}}$ as $L \to K$.

Fig. 3 shows the classification accuracy with respect to number of passive clients, L when $B = 5$ quantization bits are used and $\mathrm{SNR}_{\boldsymbol{\theta}} = 20$ dB. The proposed HFCL and HFCL-SDT approaches perform better than FL for $0 < L < K$ because the collected gradients from the active clients are corrupted by wireless channel and quantization whose effects reduce as $L \to K$. When $L = 0$, HFCL and HFCL-SDT are identical to FL (all clients are active) whereas they perform identically as CL if $L = K = 10$ (all clients are passive). The HFCL-SDT provides higher accuracy than HFCL for $0 < L < K$ because the former performs gradient computation on smaller datasets at the beginning of training for $t < N$ thus reaching higher accuracy levels quicker than the latter.

In Fig. 3, we also present the performance of FL with active clients only, that is to say, the learning model is trained only on the dataset of active clients, whereas it is tested on the whole dataset. 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 our HFCL approach, in which all of the clients participate the learning stage. It is worthwhile to note that when $L = K = 10$,

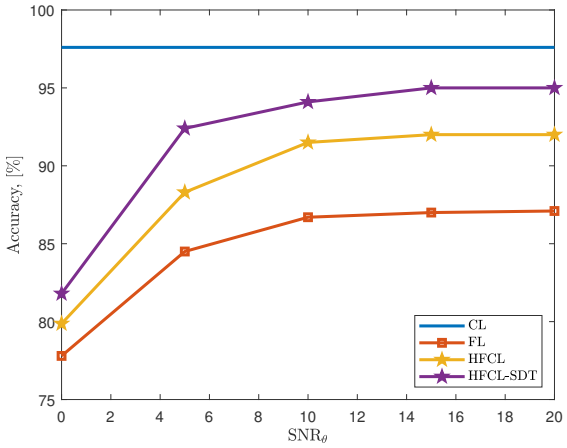


Fig. 5. Classification accuracy versus $\mathrm{SNR}_{\boldsymbol{\theta}}$ when $Q = 5$.

there will be no active clients as FL with only active clients cannot work. The performance loss as $L \to K$ is due to the absence of passive clients' datasets can be severe if the dataset is non-identically distributed because the active clients cannot learn the whole features in the dataset of other devices.

The classification accuracy (Fig. 4) in terms of the quantization levels for $B \in [1, 8]$ when $\mathrm{SNR}_{\boldsymbol{\theta}} = 20$ dB shows that both HFCL approaches perform better than FL and approach to CL as an increase in B improves the resolution of the quantization operation. The classification performance in Fig. 5 with respect to the noise level on the model parameters, i.e., $\boldsymbol{\theta}$ and $\mathbf{g}_k(\boldsymbol{\theta})$ compares the competing algorithms for $\mathrm{SNR}_{\boldsymbol{\theta}} \in [0, 20]$ dB. Here, at least 10 dB noise level is required for reliable model training for all approaches. While the same level of noise is also added onto the dataset of passive clients, its influence is more apparent on the gradients since it directly affects the learning performance.

V. Summary

We introduced a hybrid federated and centralized learning (HFCL) approach for distributed machine learning tasks. The proposed approach is helpful if a part of the edge devices lack computational capability for gradient computation during model training. In order to train the learning model collaboratively, only active devices with sufficient computational capability perform gradient computation on their local datasets whereas the remaining passive devices transmit their local datasets to the PS. The delays arising from the transmission of local datasets during training of the size of the passive clients are large are mitigated by HFCL-SDT.

Acknowledgements

S. Coleri acknowledges the support of the Scientific and Technological Research Council of Turkey (TUBITAK) EU CHIST-ERA grant 119E350. A. M. Elbir acknowledges the support of TUBITAK.

References

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

[2] J. A. Hodge, K. V. Mishra, and A. I. Zaghloul, "Deep inverse design of reconfigurable metasurfaces for future communications," *arXiv preprint arXiv:2101.09131*, 2021.

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

[4] 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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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

[12] 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.

[13] A. M. Elbir and S. Coleri, "Federated Learning for Channel Estimation in Conventional and IRS-Assisted Massive MIMO," *arXiv preprint arXiv:2008.10846*, 2020.

[14] 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.

[15] 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.

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

[17] 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.

[18] 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," *arXiv*, Sep 2020. [Online]. Available: https://arxiv.org/abs/2009.13012v1

[19] 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.

[20] 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.

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

[22] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Advances in Neural Information Processing Systems*, 2017, pp. 1709–1720.

[23] 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," *arXiv*, Feb 2020. [Online]. Available: https://arxiv.org/abs/2002.11343v2