Hierarchical Federated Learning in Wireless Networks: Pruning Tackles Bandwidth Scarcity and System Heterogeneity

Md Ferdous Pervej, *Member, IEEE*, Richeng Jin, *Member, IEEE*, and Huaiyu Dai, *Fellow, IEEE*

***Abstract*—While a practical wireless network has many tiers where end users do not directly communicate with the central server, the users' devices have limited computation and battery powers, and the serving base station (BS) has a fixed bandwidth. Owing to these practical constraints and system models, this paper leverages model pruning and proposes a pruning-enabled hierarchical federated learning (PHFL) in heterogeneous networks (HetNets). We first derive an upper bound of the convergence rate that clearly demonstrates the impact of the model pruning and wireless communications between the clients and the associated BS. Then we jointly optimize the model pruning ratio, central processing unit (CPU) frequency and transmission power of the clients in order to minimize the controllable terms of the convergence bound under strict delay and energy constraints. However, since the original problem is not convex, we perform successive convex approximation (SCA) and jointly optimize the parameters for the relaxed convex problem. Through extensive simulation, we validate the effectiveness of our proposed PHFL algorithm in terms of test accuracy, wall clock time, energy consumption and bandwidth requirement.**

***Index Terms*—Heterogeneous network, hierarchical federated learning, model pruning, resource management.**

I. Introduction

FEDERATED learning (FL) has garnered significant attention as a privacy-preserving distributed edge learning solution in wireless edge networks [1]–[3]. Since the original FL follows the parameter server paradigm, many state-of-the-art works consider a single server with distributed clients as the general system model in order to study the analytical and empirical performance [2]–[6]. Given that there are $\mathscr{U} := \{u\}_{u=1}^{\mathrm{U}}$ clients, each with a local dataset of $\mathscr{D}_u := \{\mathbf{x}_a, y_a\}_{a=1}^{A}$, where $\mathbf{x}_a$ and y_a are the a^{th} feature vector and the corresponding label, the central server wants to train a global machine learning (ML) model $\mathbf{w}$ by minimizing a weighted combination of the clients' local objective functions $f_u(\mathbf{w})$'s, as follows

$$f(\mathbf{w}) := \sum_{u=1}^{\mathrm{U}} \alpha_u f_u(\mathbf{w}), \tag{1}$$

$$f_u(\mathbf{w}) := (1/|\mathscr{D}_u|) \sum_{(\mathbf{x}_a, y_a) \in \mathscr{D}_u} \mathrm{l}(\mathbf{w}, \mathbf{x}_a, y_a), \tag{2}$$

where α_u is the corresponding weight, and $\mathrm{l}(\mathbf{w}, \mathbf{x}_a, y_a)$ denotes the loss function associated with the a^{th} data sample. However, general networks usually follow a hierarchical structure [7], where the clients are connected to edge servers, the edge servers are connected to fog nodes/servers, and these fog nodes/servers are connected to the cloud server [8]. Naturally, some recent works [9]–[14] have extended FL to accommodate this hierarchical network topology.

A client[1] does not directly communicate with the central server in the hierarchical network topology. Instead, the clients usually perform multiple local rounds of model training before sending the updated models to the edge server. The edge server aggregates the received models and updates its edge model, and then broadcasts the updated model to the associated clients for local training. The edge servers repeat this for multiple edge rounds and finally send the updated edge models to the upper-tier servers, which then undergo the same process before finally sending the updated models to the cloud/central server. This is usually known as hierarchical federated learning (HFL) [11]. On the one hand, HFL acknowledges the practical wireless heterogeneous network (HetNet) architecture. On the other hand, it avoids costly direct communication between the far-away cloud server and the capacity-limited clients [14]. Moreover, since local averaging improves learning accuracy [7], the central server ends up with a better-trained model.

While HFL can alleviate communication bottlenecks for the cloud server, data and system heterogeneity amongst the clients still need to be addressed. Since the clients are usually scattered in different locations and have various onboard sensors, the data collected/sensed by these clients are diverse, causing statistical data heterogeneity that the server cannot govern. As such, we need to embrace it in our theoretical and empirical study. Besides, the well-known system heterogeneity arises from the clients' diverse computation powers [15]. Recently, some works have been proposed to deal with system heterogeneity. For example, FedProx [16], anarchic federated

Manuscript received 3 August 2023; revised 5 January 2024; accepted 10 March 2024. This research was supported in part by the National Natural Science Foundation of China under Grants 62301487, in part by the Zhejiang Provincial Natural Science Foundation of China under Grant No. LQ23F010021 and LD21F010001, in part by the Ng Teng Fong Charitable Foundation in the form of ZJU-SUTD IDEA Grant under Grant No. 188170-11102, and in part by the US National Science Foundation under grants CNS-1824518 and ECCS-2203214. (*Corresponding author: Richeng Jin.*)

M. F. Pervej was with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695, USA, and now is with the Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA 90089, USA (e-mail: pervej@usc.edu).

H. Dai is with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695, USA (e-mail: hdai@ncsu.edu).

R. Jin is with the College of Information Science and Electronic Engineering, Zhejiang University, the Zhejiang–Singapore Innovation and AI Joint Research Lab, and the Zhejiang Provincial Key Lab of Information Processing, Communication and Networking (IPCAN), Hangzhou, China, 310000 (e-mail: richengjin@zju.edu.cn).

[1]The terms client and UE are interchangeably used when there is no ambiguity.

averaging (AFA) [17] and federated normalized averaging algorithm (FedNova) [18], to name a few, considered different local rounds for different clients to address the system heterogeneity. More specifically, FedProx adds a proximal term to the client's local objective function to handle heterogeneity. AFA and FedNova present different ways to aggregate clients trained models' weights at the server to tackle this system heterogeneity. However, these algorithms still assume that the client has and trains the original ML model, i.e., neither the computation time for the client's local training nor the communication overhead for offloading the trained model to the server is considered in system design.

Model pruning has attracted research interest recently [19], [20]. It makes the over-parameterized model sparser, which allows the less computationally capable clients to perform local training more efficiently without sacrificing much of the test accuracy. Besides, since the trained model contains fewer non-zero entries, the communication overhead over the unreliable wireless link between the client and the associated base station (BS) also dramatically reduces. However, pruning generally introduces errors that only partially vanish, causing the pruned model to converge only to a neighborhood of the optimal solution [19]. Besides, unlike the traditional FL, where the model averaging happens only at the central server, HFL has multiple hierarchical levels that may adopt their own aggregation strategy. Therefore, model pruning at the local client level leads to additional errors in the available models at different levels, eventually contributing to the global model. As such, more in-depth study is in need to understand the full benefit of model pruning in hierarchical networks.

A. Related Work

Some recent works studied HFL [9]–[14] and model pruning-based traditional single server based FL [20]–[24] separately. In [9], Xu *et al.* proposed an adaptive HFL scheme, where they optimized edge aggregation intervals and bandwidth allocation to minimize a weighted combination of the model training delay and training loss. Liu *et al.* proposed network-assisted HFL in [10], where they optimized wireless resource allocation and user associations to minimize 1) learning latency for independent identically distributed (IID) data distribution and 2) weighted sum of the total data distance and learning latency for the non-IID data distribution. Similar to [9], [10], Luo *et al.* jointly optimized the wireless network parameters in order to minimize the weighted combination of the total energy consumption and delay during the training process in [11]. Besides, [12] also proposed an HFL algorithm based on federated averaging (FedAvg) [1]. In [13], Feng *et al.* proposed a mobility-aware clustered FL algorithm owing to user mobility. More specifically, assuming that all users had an equal probability of staying at a cluster, the authors derived an upper bound of the convergence rate to capture the impact of user mobility, data heterogeneity and network heterogeneity. Abad *et al.* also optimized wireless resources to reduce the communication latency and facilitate HFL in [14].

On the model pruning side, Jiang *et al.* considered two-stage distributed model pruning in [20] with traditional single server based FL setting without any wireless network aspects. In a similar setting, Zhu *et al.* proposed a layer-wise pruning mechanism in [21]. Liu *et al.* optimized the pruning ratio and time allocation in [22] in order to maximize the convergence rate in a time division multiple access operated small BS (sBS). The idea was extended to joint client selection, pruning ratio optimization and time allocation in [23]. Using a similar network model, Ren *et al.* optimized pruning ratios and bandwidth allocations jointly to minimize a weighted combination of the FL training time and pruning error in [24]. These works [22]–[24] decomposed the original problem into different sub-problems that they solved iteratively in an attempt to solve the original problem sub-optimally. Moreover, [22]–[24] considered a simple network system model with a single BS serving the distributed clients with the wireless links.

B. Our Contributions

While the studies mentioned above shed some light on HFL and model pruning in the traditional single server based FL, the impact of pruning on HFL in resource-constrained wireless HetNet is yet to be explored. On the one hand, the clients need to train the original model for a few local episodes to determine the neurons they shall prune, which adds additional time and energy costs. On the other hand, pruning adds errors to the learning performance. Therefore, it is necessary to theoretically and empirically study these errors from different levels in HFL. Moreover, it is also crucial to justify how and when one should adopt model pruning in practical wireless HetNets. Motivated by these, in this work, we present our pruning-enabled HFL (PHFL) framework with the following major contributions:

- Considering a practical wireless HetNet, we propose a PHFL solution in which the clients perform local training on the initial models to determine the neurons to prune, perform extensive training on the pruned models, and offload the trained models under strict delay and energy constraints.
- We theoretically analyze how pruning introduces errors in different levels under resource constraints in wireless HetNets by deriving a convergence bound that captures the impact of the wireless links between the clients and server and the pruning ratios. More specifically, the proposed solution converges to the neighborhood of a stationary point of traditional HFL with a convergence rate of $\mathcal{O}\left(1/\sqrt{\mathrm{U}T}\right)+\mathcal{O}(\beta^2 D^2 \delta^{\mathrm{th}})$, where U is the total number of clients, T is the total local iterations, β quantifies smoothness of the loss function, D^2 is an upper bound of the L_2 norm of the model weights, and $0<\delta^{\mathrm{th}}<1$ is the maximum allowable pruning ratio.
- Then, we formulate an optimization problem to maximize the convergence rate by jointly configuring wireless resources and system parameters. To tackle the non-convexity of the original problem, we use a successive convex approximation (SCA) algorithm to solve the relaxed convex problem efficiently.
- Finally, using extensive simulation on two popular datasets and three popular ML models, we show the

TABLE I: Important Notations

Notation	Description
u; b; l	u^{th} user; b^{th} sBS; l^{th} mBS
$\mathscr{B}_l$; k	sBS set under the l^{th} mBS; k^{th} sBS in $\mathscr{B}_l$
$\mathscr{V}_{k,l}$; j	VC set of sBS k under the l^{th} mBS; j^{th} VC of sBS k
$\mathscr{U}_{j,k,l}$; i	Client set of the j^{th} VC of l^{th} sBS under the l^{th} mBS; i^{th} client in $\mathscr{U}_{j,k,l}$
z; $\mathscr{Z}$	z^{th} pRB; pRB set
P_i^t	Client i's transmission power during t
$\mathbf{w}$; $\mathbf{m}$; $\tilde{\mathbf{w}}$	Original model; binary mask; pruned model
$\mathbf{w}_i$; $\mathbf{w}_j$; $\mathbf{w}_k$; $\mathbf{w}_l$	Local model of the client, VC, sBS, and mBS
$f_i(\cdot)$; $f_j(\cdot)$; $f_k(\cdot)$; $f_l(\cdot)$; $f(\cdot)$	Loss function of the client, VC, sBS, mBS, and central server, respectively
$\nabla f_i(\cdot)$; $g(\cdot)$; η	True gradient; stochastic gradient; learning rate
d; d_p	Total & pruned parameters of the ML model
δ_i^t; δ^{th}	Pruning ratio of client i during t; max pruning ratio
α_i; α_j; α_k; α_l	Weight of i^{th} client, j^{th} VC, k^{th} sBS, and l^{th} mBS
ρ	Number of SGD rounds on $\mathbf{w}$ to get winning ticket
κ_0; κ_1; κ_2; κ_3	Number of local, VC, sBS, and mBS rounds
$\mathbf{1}_i^t$; p_i^t	Binary indicator function to define if sBS receives i^{th} client's trained model; probability that $\mathbf{1}_i^t = 1$
β	Smoothness of the loss functions
σ^2	Bounded variance of the gradients
ε^2	Bounded divergence of the loss functions of two inter-connected tiers
G^2; D^2	Upper bound of the L_2-norm of stochastic gradients and model weights, respectively
f_i^t; $\mathrm{f}_i^{\max}$	CPU clock cycle of i during t; max CPU cycle of i
b, n	Batch size; number of mini-batch
c_i; D_i	Required number of CPU cycle of i to process 1-bit data; each data sample size in bits
FPP	Floating point precision
$\mathrm{t}_i^{\text{cpd}}$; $\mathrm{e}_i^{\text{cpd}}$	Time and energy overheads to get the lottery ticket
$\mathrm{t}_i^{\text{cps}}$; $\mathrm{e}_i^{\text{cps}}$	Time and energy overheads to compute κ_0 local SGD rounds with the pruned model
t_i^{up}; e_i^{up}	Time and energy overheads for offloading client i's trained model
$\mathrm{t}_i^{\text{tot}}$; $\mathrm{e}_i^{\text{tot}}$	Client i's total time and energy overheads to finish one VC round
t^{th}; e_i^{th}	Time and energy budgets to finish one VC round

effectiveness of our proposed solution in terms of test accuracy, training time, energy consumption and bandwidth requirement.

The rest of the paper is organized as follows: Section II introduces our system model. Detailed theoretical analysis is performed in Section III, followed by our joint problem formulation and solution in Section IV. Based on our extensive simulation, we discuss the results in Section V. Finally, Section VI concludes the paper. Moreover, Table I summarizes the important notations used in the paper.

II. System Model

A. Wireless Network Model

We consider a generic heterogeneous network (HetNet) consisting of some UEs, sBSs and macro BSs (mBSs), as shown in Fig. 1. Denote the UE, sBS and mBS sets by $\mathscr{U} := \{u\}_{u=1}^{\mathrm{U}}$, $\mathscr{B} := \{b\}_{b=1}^{\mathrm{B}}$ and $\mathscr{L} := \{l\}_{l=1}^{L}$, respectively. Each UE and sBS are connected to one sBS and mBS, respectively. The mBSs are connected to the central server. While the UEs communicate over wireless links with the sBS, the connections between the sBS and mBS and between the mBS and central server are wired. Moreover, due to UEs' system heterogeneity, we consider that the sBS groups UEs with similar computation and battery powers into a virtual cluster (VC).

We can benefit from the VC since it enables one additional aggregation tier. Besides, thanks to the recent progress of the proximity-services in practical networks [25], one can also select a cluster head and leverage device-to-device communication to receive and distribute the models to the UEs in the same VC. However, in this work, we assume that the sBS creates the VC and also manages it. We use the notation $\mathscr{B}_l := \{k\}_{k=1}^{B_l}$, $\mathscr{V}_{k,l} := \{j\}_{j=1}^{V_{k,l}}$ and $\mathscr{U}_{j,k,l} := \{i\}_{i=1}^{U_{j,k,l}}$ to represent the sBS set associated to mBS l, VC set of sBS $k \in \mathscr{B}_l$ and UE set of VC $j \in \mathscr{V}_{k,l}$, respectively. Moreover, denote the UEs associated with sBS k, mBS l by $\mathscr{U}_{k,l} = \bigcup_{j=1}^{V_{k,l}} \mathscr{U}_{j,k,l}$ and $\mathscr{U}_l = \bigcup_{k=1}^{B_l} \mathscr{U}_{k,l}$, respectively. Finally, $\mathscr{U} = \bigcup_{l=1}^{L} \mathscr{U}_l$. Note that we consider that these associations are known and provided by the network administrator. The network has a fixed bandwidth divided into orthogonal physical resource blocks (pRBs) for performing the FL task. Denote the pRB set by $\mathscr{Z} := \{z\}_{z=1}^{Z}$. Each mBS reuses the same pRB set, i.e., the frequency reuse factor is 1. Besides, each mBS allocates dedicated pRBs to its associated sBSs. Each sBS further uses dedicated pRBs to communicate with the associated UEs. As such, there is no intra-tier interference in the system.

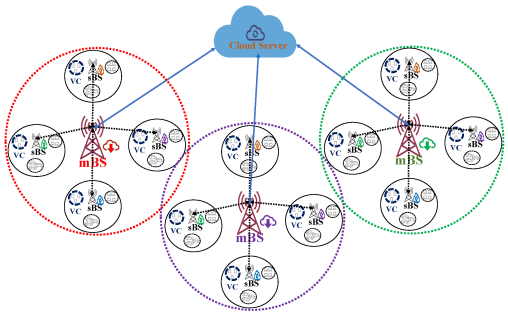


Fig. 1: Pruning-enabled hierarchical FL system model

To this end, denote the distance between UE i and the sBS k by $d_{i,k}$. The wireless fading channel between the UE and sBS follows Rayleigh distribution[2], and is denoted by $h_{i,k}^t$. The transmission power of UE is denoted by P_i^t. As such, the uplink signal-to-noise-plus-interference ratio (SINR) is expressed as

$$\gamma_{i,k}^t = \mathrm{P}_i^t h_{i,k}^t d_{i,k}^{-\alpha} / (\omega \zeta^2 + I_{i,k}^t), \tag{3}$$

where α is the path loss exponent, ζ^2 is the variance of the circularly symmetric zero-mean Gaussian distributed random noise and ω is the pRB size. Moreover, $I_{i,k}^t = \sum_{l=1}^{L} \sum_{k'=1, k' \neq k}^{B_l} \sum_{j'=1}^{V_{k,l}} \sum_{i' \in \mathscr{U}_{j',k',l}} \mathrm{P}_{i'}^t h_{i',k'}^t d_{i',k'}^{-\alpha}$ is the inter-cell interference. Thus, the data rate

$$r_i^t = \omega \log_2 \left[1 + \gamma_{i,k}^t\right]. \tag{4}$$

B. Pruning-Enabled Hierarchical Federated Learning (PHFL)

In this work, we consider that each client uses mini-batch stochastic gradient descent (SGD) to minimize (2) over n mini-batches since the gradient descent over the entire dataset is time-consuming. Denote the stochastic gradient $g(\mathbf{w})$ such that $\mathbb{E}_{\xi_n \sim \mathscr{D}_i}[g(\mathbf{w})] := \nabla f_i(\mathbf{w})$, where ξ_n is a randomly sampled batch from dataset $\mathscr{D}_i$. However, computation with the original model $\mathbf{w} \in \mathbb{R}^d$ is still costly and may significantly extend the

[2]Rayleigh distribution is widely used for its simplicity. Other distributions are not precluded.

training time of a computationally limited client. To alleviate this, the UE trains a pruned model by removing some of the weights of the original model $\mathbf{w}$ [19].

Denote a binary mask by $\mathbf{m}_i \in \mathbb{R}^d$ and the pruned model by $\tilde{\mathbf{w}}_i := \mathbf{w}_i \odot \mathbf{m}_i$, where $\odot$ means element-wise multiplication. Note that training the pruned model $\tilde{\mathbf{w}}_i$ is computationally less expensive as it has fewer parameters than the original model. It is worth pointing out that the UE utilizes the state-of-the-art lottery ticket hypothesis [26] to find the winning ticket $\tilde{\mathbf{w}}_i$ and the corresponding mask with the following key steps. Denote the number of parameters required to be pruned by d_p. The UE performs ρ local iterations on the original model $\mathbf{w}_i$ as

$$\mathbf{w}_i^{\rho} = \mathbf{w}_i - \eta \sum_{\bar{\rho}=1}^{\rho} g(\mathbf{w}_i^{\bar{\rho}}), \tag{5}$$

where η is the step size. The UE then prunes d_p entries of $\mathbf{w}_i^{\rho} \in \mathbb{R}^d$ with the smallest magnitudes[3] and generates a binary mask $\mathbf{m}_i \in \{0,1\}^d$. To that end, the client obtains the winning ticket $\tilde{\mathbf{w}}_i$ by retaining the original weights of the corresponding nonzero entries of the mask $\mathbf{m}_i$ from the original initial model $\mathbf{w}_i$ [26]. Note that other pruning techniques can also be adopted. Moreover, we denote the pruning ratio by [23]

$$\delta_i := d_p/d. \tag{6}$$

Given the pruned model $\tilde{\mathbf{w}}_i^{t,0} := \mathbf{w}_i^t \odot \mathbf{m}_i^t$, the loss function of the UE is rewritten as

$$f_i(\tilde{\mathbf{w}}_i^{t,0}) := [1/|\mathscr{D}_i|] \sum_{(\mathbf{x}_a, y_a) \in \mathscr{D}_i} f(\tilde{\mathbf{w}}_i^{t,0}, \mathbf{x}_a, y_a), \tag{7}$$

Each UE updates its pruned model as

$$\tilde{\mathbf{w}}_i^{t+1} := \tilde{\mathbf{w}}_i^{t,0} - \eta g(\tilde{\mathbf{w}}_i^{t,0}) \odot \mathbf{m}_i^t. \tag{8}$$

As such, we denote the loss functions of the j^{th} VC, k^{th} sBS, l^{th} mBS and central server as $f_j(\mathbf{w}_j) := \sum_{i=1}^{U_{j,k,l}} \alpha_i f_i(\tilde{\mathbf{w}}_i)$, $f_k(\mathbf{w}_k) := \sum_{j=1}^{V_{k,l}} \alpha_j f_j(\mathbf{w}_j)$, $f_l(\mathbf{w}_l) := \sum_{k=1}^{B_l} \alpha_k f_k(\mathbf{w}_k)$, and $f(\mathbf{w}) := \sum_{l=1}^{L} \alpha_l f_l(\mathbf{w}_l)$, respectively. Note that for simplicity, we consider identical weights, i.e., $\alpha_i = 1/U_{j,k,l}$, $\alpha_j = 1/V_{k,l}$, $\alpha_k = 1/B_l$ and $\alpha_l = 1/L$, which can be easily adjusted for other weighting strategies. Besides, since aggregation happens at different times and different levels, we need to capture the time indices explicitly. Let each UE perform κ_0 local iterations before sending the updated model to the associated VC. It is worth noting that the winning ticket and the corresponding binary mask are only obtained before these κ_0 local rounds begin. Besides, since training the original model is costly, it is reasonable to consider $\rho \ll \kappa_0$[4]. Moreover, although the sBS-mBS and mBS-central server links are wired, communication and computation at these nodes incur additional burdens. As such, we assume that each VC, sBS and mBS perform κ_1, κ_2 and κ_3 rounds, respectively, before sending the trained model to the respective upper layers. Denote the indices of the current global round, mBS round, sBS round, VC round and UE's local round by m, t_3, t_2, t_1 and t_0, respectively. Besides, similar to [9], [13], let $t := [\{(m\kappa_3 + t_3)\kappa_2 + t_2\}\kappa_1 + t_1]\kappa_0 + t_0$ denote the index of local update iterations.

If $t \ mod \ \kappa_0 = 0$, the UE receives the latest available model of its associated VC, i.e.,

$$\mathbf{w}_i^{\bar{t}_0} \leftarrow \mathbf{w}_j^{\bar{t}_0}, \tag{9}$$

where $\bar{t}_0 = [\{(m\kappa_3 + t_3)\kappa_2 + t_2\}\kappa_1 + t_1]\kappa_0$. The UE then computes the pruned model $\tilde{\mathbf{w}}_i^{\bar{t}_0,0}$ and the binary mask $\mathbf{m}_i^{\bar{t}_0}$. It then performs κ_0 local SGD rounds as

$$\tilde{\mathbf{w}}_i^{\bar{t}_0+\kappa_0} = \tilde{\mathbf{w}}_i^{\bar{t}_0,0} - \eta \sum_{t_0=1}^{\kappa_0} g\big(\tilde{\mathbf{w}}_i^{\bar{t}_0+t_0,0}\big) \odot \mathbf{m}_i^{\bar{t}_0}. \tag{10}$$

Each VC j performs $t_1 = 0, \ldots, \kappa_1 - 1$ local rounds. When $(t_1+1) \ mod \ \kappa_1 = 0$, the VC's model gets updated by the latest available sBS model, i.e.,

$$\mathbf{w}_j^{\bar{t}_1} \leftarrow \mathbf{w}_k^{\bar{t}_1}, \tag{11}$$

where $\bar{t}_1 = \{(m\kappa_3 + t_3)\kappa_2 + t_2\}\kappa_1\kappa_0$. Besides, between two VC rounds, the local model of the VC is updated as

$$\mathbf{w}_j^{\bar{t}_1+(t_1+1)\kappa_0} = \sum_{i=1}^{U_{j,k,l}} \alpha_i \tilde{\mathbf{w}}_i^{\bar{t}_1+(t_1+1)\kappa_0} = \tilde{\mathbf{w}}_j^{\bar{t}_1+t_1\kappa_0,0} - \eta \sum_{i=1}^{U_{j,k,l}} \big[\mathbf{1}_i^{\bar{t}_1+t_1\kappa_0}/p_i^{\bar{t}_1+t_1\kappa_0}\big] \alpha_i \sum_{t_0=1}^{\kappa_0} g\big(\tilde{\mathbf{w}}_i^{\bar{t}_1+t_1\kappa_0+t_0,0}\big) \odot \mathbf{m}_i^{\bar{t}_1+t_1\kappa_0}, \tag{12}$$

where $\tilde{\mathbf{w}}_j^{\bar{t}_1+t_1\kappa_0,0} := \sum_{i=1}^{U_{j,k,l}} \alpha_i \tilde{\mathbf{w}}_i^{\bar{t}_1+t_1\kappa_0,0}$ and $\mathbf{1}_i^{\bar{t}_1+t_1\kappa_0}$ is a binary indicator function that indicates whether the sBS receives the trained model of i during the VC aggregation round $\bar{t}_1 + (t_1+1)\kappa_0$ or not, and is defined as follows:

$$\mathbf{1}_i^{\bar{t}_1+t_1\kappa_0} := \begin{cases} 1, & \text{with probability } p_i^{\bar{t}_1+t_1\kappa_0}, \\ 0, & \text{otherwise}, \end{cases}, \tag{13}$$

where $p_i^{\bar{t}_1+t_1\kappa_0}$ is the probability of receiving the trained model over the wireless link and is calculated in the subsequent section (*c.f.* (42)). Note that since the sBS has to receive the gradient over the wireless link, we use the binary indicator function $\mathbf{1}_i^{\bar{t}_1+t_1\kappa_0}$ in (12) as a common practice [13], [27].

The sBS performs $t_2 = 0, \ldots, \kappa_2 - 1$ local rounds before updating its model. When $(t_2+1) \ mod \ \kappa_2 = 0$, the sBS updates its local model with the latest available model at its associated mBS, i.e.,

$$\mathbf{w}_k^{\bar{t}_2} \leftarrow \mathbf{w}_l^{\bar{t}_2}, \tag{14}$$

where $\bar{t}_2 = (m\kappa_3 + t_3)\kappa_2\kappa_1\kappa_0$. In each sBS round t_2, the sBS updates its model as

$$\mathbf{w}_k^{\bar{t}_2+(t_2+1)\kappa_1\kappa_0} = \sum_{j=1}^{V_{k,l}} \alpha_j \mathbf{w}_j^{\bar{t}_2+(t_2+1)\kappa_1\kappa_0} = \tilde{\mathbf{w}}_k^{\bar{t}_2+t_2\kappa_1\kappa_0,0} - \eta \sum_{j=1}^{V_{k,l}} \alpha_j \sum_{t_1=0}^{\kappa_1-1} \sum_{i=1}^{U_{j,k,l}} \alpha_i \frac{\mathbf{1}_i^{\bar{t}_2+\tilde{t}_2}}{p_i^{\bar{t}_2+\tilde{t}_2}} \sum_{t_0=1}^{\kappa_0} g\big(\tilde{\mathbf{w}}_i^{\bar{t}_2+\tilde{t}_2+t_0,0}\big) \odot \mathbf{m}_i^{\bar{t}_2+\tilde{t}_2}, \tag{15}$$

where $\tilde{\mathbf{w}}_k^{\bar{t}_2+t_2\kappa_1\kappa_0,0} := \sum_{j=1}^{V_{k,l}} \alpha_j \tilde{\mathbf{w}}_{j,k,l}^{\bar{t}_2+t_2\kappa_1\kappa_0,0}$ and $\tilde{t}_2 = (t_2\kappa_1 + t_1)\kappa_0$. Similarly, the mBS performs $t_3 = 0, \ldots, \kappa_3 - 1$ local rounds before updating its local model with the latest available global model when $(t_3+1) \ mod \ \kappa_3 = 0$, i.e.,

$$\mathbf{w}_l^{m\kappa_3\kappa_2\kappa_1\kappa_0} \leftarrow \mathbf{w}^{m\kappa_3\kappa_2\kappa_1\kappa_0}. \tag{16}$$

Moreover, between two mBS rounds, we can write

$$\mathbf{w}_l^{(m\kappa_3+(t_3+1))\kappa_2\kappa_1\kappa_0} = \tilde{\mathbf{w}}_l^{(m\kappa_3+t_3)\kappa_2\kappa_1\kappa_0,0} -$$

[3]The time complexity to sort the d parameters and then prune the d_p smallest ones depends on the sorting technique. Many sorting algorithms have logarithmic time complexity, which can be computed quickly in modern graphical processing units. Following the common practice in literature [22]–[24], the overhead for pruning is ignored in this work. Moreover, the proposed method can be readily extended to incorporate the time overhead for pruning.

[4]In our simulation, we considered $\rho < \kappa_0$ and observed that even $\rho = 1$ with $\kappa_0 \geq 5$ performed well.

Algorithm 1: Pruning-Enabled Hierarchical FL

Input: Total global round M, initial model $\mathbf{w}^0$
Synchronize all edge devices with the initial model $\mathbf{w}^0$
for *All global rounds* $m = 0$ *to* $M-1$ **do**
  for *All mBS rounds* $t_3 = 0, 1, \ldots, \kappa_3 - 1$ **do**
    for *All sBS rounds* $t_2 = 0, 1, \ldots, \kappa_2 - 1$ **do**
      for *All VC rounds* $t_1 = 0, 1, \ldots, \kappa_1 - 1$ **do**
        for $i \in \mathscr{U}_{j,k,l}$ *in parallel* **do**
          UE receives the latest available model from the associated VC
          Compute binary mask and get the winning ticket using lottery ticket hypothesis
          for *All local rounds* $t_0 = 1, 2, \ldots, \kappa_0$ **do**
            $t \leftarrow [\{(m\kappa_3 + t_3)\kappa_2 + t_2\}\kappa_1 + t_1]\kappa_0 + t_0$
            UE updates local model using (8)
          end
        end
        sBS updates VC model using (11) and (12)
      end
      sBS update local cell model using (14) and (15)
    end
    mBS updates local cell model using (16) and (17)
  end
  Central server updates global model using (18)
end
Output: Global ML model $\mathbf{w}^{M-1}$

$$\eta \sum_{k=1}^{B_l} \alpha_k \sum_{t_2=0}^{\kappa_2-1} \sum_{j=1}^{V_{k,l}} \alpha_j \sum_{t_1=0}^{\kappa_1-1} \sum_{i=1}^{U_{j,k,l}} \alpha_i \frac{\mathbf{1}_i^{\bar{t}_3}}{p_i^{\bar{t}_3}} \sum_{t_0=1}^{\kappa_0} g\big(\tilde{\mathbf{w}}_i^{\bar{t}_3+t_0,0}\big) \odot \mathbf{m}_i^{\bar{t}_3}, \quad (17)$$

where $\tilde{\mathbf{w}}_l^{(m\kappa_3+t_3)\kappa_2\kappa_1\kappa_0,0} := \sum_{k=1}^{B_l} \alpha_k \tilde{\mathbf{w}}_k^{(m\kappa_3+t_3)\kappa_2\kappa_1\kappa_0,0}$ and $\bar{t}_3 = [((m\kappa_3+t_3)\kappa_2+t_2)\kappa_1+t_1]\kappa_0$. Finally, the central server performs global aggregation by collecting the updated models from all mBSs as follows:

$$\mathbf{w}^{(m+1)\prod_{z=0}^{3}\kappa_z} = \tilde{\mathbf{w}}^{m\prod_{z=0}^{3}\kappa_z,0} - \eta \sum_{l=1}^{L} \alpha_l \sum_{t_3=0}^{\kappa_3-1} \sum_{k=1}^{B_l} \alpha_k \sum_{t_2=0}^{\kappa_2-1} \sum_{j=1}^{V_{k,l}} \alpha_j \sum_{t_1=0}^{\kappa_1-1} \sum_{i=1}^{U_{j,k,l}} \alpha_i \frac{\mathbf{1}_i^{\bar{t}_3}}{p_i^{\bar{t}_3}} \sum_{t_0=1}^{\kappa_0} g\big(\tilde{\mathbf{w}}_i^{\bar{t}_3+t_0,0}\big) \odot \mathbf{m}_i^{\bar{t}_3}, \quad (18)$$

where $\tilde{\mathbf{w}}^{m\prod_{z=0}^{3}\kappa_z,0} := \sum_{l=1}^{L} \alpha_l \tilde{\mathbf{w}}_l^{m\prod_{z=0}^{3}\kappa_z,0}$.

The proposed PHFL process is summarized in Algorithm 1.

III. PHFL: Convergence Analysis

A. Assumptions

We make the following standard assumptions [7], [13], [20], [23], [28]:

1) The loss functions are lower-bounded, i.e., $f(\mathbf{w}) \geq f_{\text{inf}}$.
2) The loss functions are β-smooth, i.e., $\|\nabla f_i(\mathbf{w}) - \nabla f_i(\mathbf{w}')\| \leq \beta \|\mathbf{w} - \mathbf{w}'\|$.
3) Mini-batch gradients are unbiased $\mathbb{E}_{\xi \sim \mathscr{D}_i}[g(\tilde{\mathbf{w}}_i)] = \nabla f_i(\tilde{\mathbf{w}}_i)$. Besides, the variance of the gradients is bounded, i.e., $\|g(\tilde{\mathbf{w}}_i) - \nabla f_i(\tilde{\mathbf{w}}_i)\|^2 \leq \sigma^2$.
4) The divergence of the local, VC, sBS, mBS and global loss functions are bounded for all i, j, k, l and $\mathbf{w}$, i.e.,
$$\sum_{i=1}^{U_{j,k,l}} \alpha_i \|\nabla f_i(\mathbf{w}) - \nabla f_j(\mathbf{w})\|^2 \leq \varepsilon_{\text{vc}}^2,$$
$$\sum_{j=1}^{V_{k,l}} \alpha_j \|\nabla f_j(\mathbf{w}) - \nabla f_k(\mathbf{w})\|^2 \leq \varepsilon_{\text{sbs}}^2,$$
$$\sum_{k=1}^{B_l} \alpha_k \|\nabla f_k(\mathbf{w}) - \nabla f_l(\mathbf{w})\|^2 \leq \varepsilon_{\text{mbs}}^2,$$
$$\sum_{l=1}^{L} \alpha_l \|\nabla f_l(\mathbf{w}) - \nabla f(\mathbf{w})\|^2 \leq \varepsilon^2.$$
5) The stochastic gradients are independent of each other in different iterations.
6) The stochastic gradients are bounded, i.e., $\mathbb{E}\|g(\mathbf{w}_i)\|^2 \leq G^2$.
7) The model weights are bounded, i.e., $\mathbb{E}\|\mathbf{w}_i\|^2 \leq D^2$.
8) The pruning ratio $\delta_i^t \in [0, \delta^{\text{th}}]$, in which $0 < \delta^{\text{th}} < 1$ and δ^{th} is the maximum allowable pruning ratio, follows
$$\delta_i^t \geq \big\|\mathbf{w}_i^t - \tilde{\mathbf{w}}_i^{t,0}\big\|^2 / \|\mathbf{w}_i^t\|^2. \quad (19)$$

Since the updated global, mBS, sBS and VC models are not available in each local iteration t, similar to standard practice [7], [9], [13], we assume the virtual copies of these models, denoted by $\bar{\mathbf{w}}^t$, $\bar{\mathbf{w}}_l^t$, $\bar{\mathbf{w}}_k^t$ and $\bar{\mathbf{w}}_j^t$, respectively, are available. Besides, we assume that the bounded divergence assumptions amongst the above loss functions also hold for these virtual models. Moreover, analogous to our previous notations, we express $\tilde{\bar{\mathbf{w}}}^{t,0} := \sum_{l=1}^{L} \alpha_l \sum_{k=1}^{B_l} \alpha_k \sum_{j=1}^{V_{k,l}} \alpha_j \sum_{i=1}^{U_{j,k,l}} \alpha_i \tilde{\mathbf{w}}_i^{t,0} = \sum_{u=1}^{\text{U}} \alpha_u \tilde{\mathbf{w}}_u^{t,0}$ and $\tilde{\bar{\mathbf{w}}}^0 := \tilde{\bar{\mathbf{w}}}^{0,0}$.

B. Convergence Analysis

Similar to existing literature [7], [9], [13], [20], we consider the average global gradient norm as the indicator of the proposed PHFL algorithm's convergence. As such, in the following, we seek an θ_{PHFL}-suboptimal solution such that $\frac{1}{T}\sum_{t=0}^{T-1} \|\sum_{u=1}^{\text{U}} \alpha_u \nabla f_u(\tilde{\bar{\mathbf{w}}}^{t,0}) \odot \mathbf{m}_u^t\|^2 \leq \theta_{\text{PHFL}}$ and $\theta_{\text{PHFL}} \geq 0$. Particularly, we start with Theorem 1 that requires bounding the differences amongst the models in different hierarchical levels. These differences are first calculated in Lemma 1 to Lemma 4 and then plugged into Theorem 1 to get the θ_{PHFL}-suboptimal bound in Corollary 1.

Theorem 1. *When the assumptions in Section III-A hold and* $\eta \leq 1/\beta$, *we have*

$$\theta_{\text{PHFL}} \leq \mathscr{O}\left(\frac{f(\tilde{\bar{\mathbf{w}}}^0) - f_{\text{inf}}}{\eta T}\right) + \mathscr{O}\left(\frac{\beta \eta \sigma^2}{\text{U}}\right) + \underbrace{\mathscr{O}\big(\delta^{\text{th}} \beta^2 D^2\big)}_{\text{pruning error}} + \underbrace{\mathscr{O}\big(\beta \eta G^2 \cdot \varphi_{\text{w},0}(\boldsymbol{\delta}, \mathbf{f}, \mathbf{P})\big)}_{\text{wireless factor}} + \mathscr{O}\big(\beta^2 \big[\text{L}_1 + \text{L}_2 + \text{L}_3 + \text{L}_4\big]\big), \quad (20)$$

where $\boldsymbol{\delta} = \{\delta_1^t, \ldots, \delta_U^t\}_{t=0}^{T-1}$, $\mathbf{f} = \{\text{f}_1^t, \ldots, \text{f}_U^t\}_{t=0}^{T-1}$, $\mathbf{P} = \{P_1^t, \ldots, P_U^t\}_{t=0}^{T-1}$ *and* f_i^t *is the* i^{th} *client's central processing unit (CPU) frequency in the wireless factor. Besides, the terms* L_1, L_2, L_3 *and* L_4 *are*

$$\varphi_{\text{w},0}(\boldsymbol{\delta}, \mathbf{f}, \mathbf{P}) = \frac{1}{T} \sum_{t=0}^{T-1} \sum_{l=1}^{L} \alpha_l^2 \sum_{k=1}^{B_l} \alpha_k^2 \sum_{j=1}^{V_{k,l}} \alpha_j^2 \sum_{i=1}^{U_{j,k,l}} \alpha_i^2 \left[\frac{1}{p_i^t} - 1\right], \quad (21)$$

$$\text{L}_1 = \frac{1}{T} \sum_{t=0}^{T-1} \sum_{l=1}^{L} \alpha_l \sum_{k=1}^{B_l} \alpha_k \sum_{j=1}^{V_{k,l}} \alpha_j \sum_{i=1}^{U_{j,k,l}} \alpha_i \mathbb{E}\|\bar{\mathbf{w}}_j^t - \tilde{\mathbf{w}}_i^t\|^2, \quad (22)$$

$$\text{L}_2 = [1/T] \sum_{t=0}^{T-1} \sum_{l=1}^{L} \alpha_l \sum_{k=1}^{B_l} \alpha_k \sum_{j=1}^{V_{k,l}} \alpha_j \mathbb{E}\|\bar{\mathbf{w}}_k^t - \bar{\mathbf{w}}_j^t\|^2, \quad (23)$$

$$\text{L}_3 = [1/T] \sum_{t=0}^{T-1} \sum_{l=1}^{L} \alpha_l \sum_{k=1}^{B_l} \alpha_k \mathbb{E}\|\bar{\mathbf{w}}_l^t - \bar{\mathbf{w}}_k^t\|^2, \quad (24)$$

$$\text{L}_4 = [1/T] \sum_{t=0}^{T-1} \sum_{l=1}^{L} \alpha_l \mathbb{E}\|\bar{\mathbf{w}}^t - \bar{\mathbf{w}}_l^t\|^2. \quad (25)$$

The proof of Theorem 1 and the subsequent Lemmas are left in the supplementary materials.

Remark 1. *The first term in (20) is what we get for centralized learning, while the second term arises from the randomness*

of the mini-batch gradients [29]. The third term appears from model pruning. Besides, the fourth term arises from the wireless links among the sBS and UEs. It is worth noting that $\varphi_{\mathrm{w},0}(\boldsymbol{\delta},\mathbf{f},\mathbf{P})=0$ *when all* p_i^t*'s are* 1*'s. Finally, the last term is due to the difference among the VC-local, sBS-VC, sBs-mBS and mBS-global model parameters, respectively, which are derived in the following.*

Remark 2. *When the system has no pruning, i.e., all UEs use the original models, all* $\delta_i^t=0$*. Besides, under the perfect communication among the sBS and UEs, we have* $p_i^t=1$*. In such cases, the* θ_{PHFL}*-suboptimal bound boils down to*

$$\theta_{\mathrm{PHFL}} \leq \mathscr{O}\big((f(\bar{\mathbf{w}}^0)-f_{\mathrm{inf}})/[\eta T]\big)+\mathscr{O}\big(\beta\eta\sigma^2/\mathrm{U}\big)+ \mathscr{O}\big(\beta^2[\mathrm{L}_1+\mathrm{L}_2+\mathrm{L}_3+\mathrm{L}_4]\big). \quad (26)$$

Besides, the last term in (26) appears from the four hierarchical levels. When $\mathrm{U}=1$ *and there are no levels, i.e.,* $\mathrm{L}_1=\mathrm{L}_2=\mathrm{L}_3=\mathrm{L}_4=0$*, the convergence bound is exactly the same as the original SGD with non-convex loss function [7].*

To that end, we calculate the divergence among the local, VC, sBS, mBS and global model parameters, and derive the corresponding pruning errors in each level in what follows.

Lemma 1. *When* $\eta\leq 1/[2\sqrt{10}\kappa_0\beta]$*, the average difference between the VC and local model parameters, i.e., the* L_1 *term of (20), is upper bounded as*

$$\frac{\beta^2}{T}\sum_{t=0}^{T}\sum_{l=1}^{L}\alpha_l\sum_{k=1}^{B_l}\alpha_k\sum_{j=1}^{V_{k,l}}\alpha_j\sum_{i=1}^{U_{j,k,l}}\alpha_i\mathbb{E}\left\|\bar{\mathbf{w}}_j^t-\tilde{\mathbf{w}}_i^t\right\|^2\leq \mathscr{O}\big(\kappa_0\eta^2\beta^2\sigma^2\big)+ \mathscr{O}\big(\kappa_0^2\eta^2\beta^2\varepsilon_{\mathrm{vc}}^2\big)+\mathscr{O}\big(\delta^{\mathrm{th}}\beta^2D^2\big)+\mathscr{O}\big(\kappa_0\eta^2\beta^2G^2\cdot\varphi_{\mathrm{w},\mathrm{L}_1}\big), \quad (27)$$

where $\varphi_{\mathrm{w},\mathrm{L}_1}=\frac{1}{T}\sum_{l=1}^{L}\alpha_l\sum_{k=1}^{B_l}\alpha_k\sum_{j=1}^{V_{k,l}}\alpha_j\sum_{i=1}^{U_{j,k,l}}\alpha_i\sum_{t=0}^{T-1}(1/p_i^t-1)$*.*

Remark 3. *In (27), the first term comes from the statistical data heterogeneity, while the second term arises from the divergence between the local and VC loss functions. The third term emanates from model pruning. Finally, the fourth term stems from the wireless links among the UEs and sBS.*

Lemma 2. *When* $\eta\leq 1/[2\sqrt{10}\kappa_0\kappa_1\beta]$*, the difference between the sBS model parameters and VC model parameters, i.e., the* L_2 *term of (20), is upper bounded as*

$$\frac{\beta^2}{T}\sum_{t=0}^{T-1}\sum_{l=1}^{L}\alpha_l\sum_{k=1}^{B_l}\alpha_k\sum_{j=1}^{V_{k,l}}\alpha_j\mathbb{E}\left\|\bar{\mathbf{w}}_k^t-\bar{\mathbf{w}}_j^t\right\|^2\leq \mathscr{O}\big(\beta^4\kappa_0^4\kappa_1^2\eta^4\varepsilon_{\mathrm{vc}}^2\big)+ \mathscr{O}\big(\kappa_0^2\kappa_1^2\eta^2\beta^2\varepsilon_{\mathrm{sbs}}^2\big)+\mathscr{O}\big(\kappa_0\kappa_1\eta^2\sigma^2\beta^2\big)+\mathscr{O}\big(\delta^{th}\beta^2D^2\big)+ \mathscr{O}\big(\kappa_0^3\kappa_1^2\beta^4\eta^4G^2\varphi_{\mathrm{w},\mathrm{L}_1}\big)+\mathscr{O}\big(\kappa_0\kappa_1\beta^2\eta^2\varphi_{\mathrm{w},\mathrm{L}_2}\big), \quad (28)$$

where $\varphi_{\mathrm{w},\mathrm{L}_2}=\frac{1}{T}\sum_{t=0}^{T-1}\sum_{l=1}^{L}\alpha_l\sum_{k=1}^{B_l}\alpha_k\sum_{j=1}^{V_{k,l}}\alpha_j\sum_{i=1}^{U_{j,k,l}}\alpha_i^2(1/p_i^t-1)$*.*

Remark 4. *The first term in (28) appears from the divergence of the loss functions of the clients and VC, while the second term stems from the divergence between the loss function of the VC and sBS. The rest of the terms are due to the statistical data heterogeneity, model pruning and wireless links, respectively.*

Lemma 3. *When* $\eta\leq 1/[2\sqrt{14}\kappa_0\kappa_1\kappa_2\beta]$*, the average difference between the sBS and mBS model parameters, i.e., the* L_3 *term of (20), is upper bounded as*

$$[\beta^2/T]\sum_{t=0}^{T-1}\sum_{l=1}^{L}\alpha_l\sum_{k=1}^{B_l}\alpha_k\mathbb{E}\left\|\bar{\mathbf{w}}_l^t-\bar{\mathbf{w}}_k^t\right\|^2 \leq \mathscr{O}\big(\kappa_0^3\kappa_1^2\kappa_2^2\eta^4\beta^4\varepsilon_{\mathrm{vc}}^2\big)+\mathscr{O}\big(\kappa_0^4\kappa_1^4\kappa_2^2\eta^4\beta^4\varepsilon_{\mathrm{sbs}}^2\big)+ \mathscr{O}\big(\kappa_0^2\kappa_1^2\kappa_2^2\eta^2\beta^4\varepsilon_{\mathrm{mbs}}^2\big)+\mathscr{O}\big(\kappa_0\kappa_1\kappa_2\eta^2\beta^2\sigma^2\big)+ \mathscr{O}\big(\delta^{\mathrm{th}}\beta^2D^2\big)+\mathscr{O}\big(\kappa_0^3\kappa_1^2\kappa_2^2\eta^4\beta^4G^2\cdot\varphi_{\mathrm{w},\mathrm{L}_1}\big)+ \mathscr{O}\big(\kappa_0^2\kappa_1^2\kappa_2^2\beta^4\eta^4\cdot\varphi_{\mathrm{w},\mathrm{L}_2}\big)+\mathscr{O}\big(\kappa_0\kappa_1\kappa_2\eta^2\beta^2G^2\cdot\varphi_{\mathrm{w},\mathrm{L}_3}\big). \quad (29)$$

where $\varphi_{\mathrm{w},\mathrm{L}_3}=\frac{1}{T}\sum_{l=1}^{L}\alpha_l\sum_{k=1}^{B_l}\alpha_k\sum_{j=1}^{V_{k,l}}\alpha_j^2\sum_{i=1}^{U_{j,k,l}}\alpha_i^2\sum_{t=0}^{T-1}(1/p_i^t-1)$*.*

Lemma 4. *When* $\eta\leq 1/[6\sqrt{2}\kappa_0\kappa_1\kappa_2\kappa_3\beta]$*, the average difference between the global and the mBS models, i.e., the* L_4 *term, is bounded as follows:*

$$[\beta^2/T]\sum_{t=0}^{T-1}\sum_{l=1}^{L}\alpha_l\mathbb{E}\left\|\bar{\mathbf{w}}^t-\bar{\mathbf{w}}_l^t\right\|^2\leq \mathscr{O}\big(\kappa_0^4\kappa_1^2\kappa_2^2\kappa_3^2\eta^4\beta^4\varepsilon_{\mathrm{vc}}^2\big)+ \mathscr{O}\big(\kappa_0^4\kappa_1^4\kappa_2^2\kappa_3^2\eta^4\beta^4\varepsilon_{\mathrm{sbs}}^2\big)+\mathscr{O}\big(\kappa_0^4\kappa_1^4\kappa_2^4\kappa_3^2\eta^4\beta^6\varepsilon_{\mathrm{mbs}}^2\big)+ \mathscr{O}\big(\kappa_0^2\kappa_1^2\kappa_2^2\kappa_3^2\beta^4\eta^2\varepsilon^2\big)+\mathscr{O}\big(\kappa_0\kappa_1\kappa_2\kappa_3\beta^2\eta^2\sigma^2\big)+\mathscr{O}\big(\delta^{\mathrm{th}}\beta^2D^2\big)+ \mathscr{O}\big(\kappa_0^3\kappa_1^2\kappa_2^2\kappa_3^2\eta^4\beta^4G^2\varphi_{\mathrm{w},\mathrm{L}_1}\big)+\mathscr{O}\big(\kappa_0^3\kappa_1^3\kappa_2^2\kappa_3^2\beta^4\eta^4\varphi_{\mathrm{w},\mathrm{L}_2}\big)+ \mathscr{O}\big(\kappa_0^3\kappa_1^3\kappa_2^3\kappa_3^2\eta^4\beta^4G^2\varphi_{\mathrm{w},\mathrm{L}_3}\big)+\mathscr{O}\big(\kappa_0\kappa_1\kappa_2\kappa_3\beta^2\eta^2G^2\varphi_{\mathrm{w},\mathrm{L}_4}\big), \quad (30)$$

where $\varphi_{\mathrm{w},\mathrm{L}_4}=\frac{1}{T}\sum_{l=1}^{L}\alpha_l\sum_{k=1}^{B_l}\alpha_k^2\sum_{j=1}^{V_{k,l}}\alpha_j^2\sum_{i=1}^{U_{j,k,l}}\alpha_i^2\sum_{t=0}^{T-1}(1/p_i^t-1)$*.*

Note that we have similar observations for (29) and (30) as in Remark 4. Now, using the above Lemmas, we find the final convergence rate in Corollary 1.

Corollary 1. *When* $\eta\leq 1/[6\sqrt{2}\kappa_0\kappa_1\kappa_2\kappa_3\beta]$*, the* θ_{PHFL} *bound of Theorem 1 boils down to*

$$\theta_{\mathrm{PHFL}}\leq \mathscr{O}\big([f(\tilde{\mathbf{w}}^0)-f_{\mathrm{inf}}]/[\eta T]\big)+\mathscr{O}(\beta\eta\sigma^2/\mathrm{U})+ \mathscr{O}\big(\kappa_0^2\eta^2\beta^2\varepsilon_{\mathrm{vc}}^2\big)+\mathscr{O}\big(\kappa_0^2\kappa_1^2\eta^2\beta^2\varepsilon_{\mathrm{sbs}}^2\big)+ \mathscr{O}\big(\kappa_0^2\kappa_1^2\kappa_2^2\eta^2\beta^4\varepsilon_{\mathrm{mbs}}^2\big)+\mathscr{O}\big(\kappa_0^2\kappa_1^2\kappa_2^2\kappa_3^2\beta^4\eta^2\varepsilon^2\big)+ \underbrace{\mathscr{O}\big(\delta^{\mathrm{th}}\beta^2D^2\big)}_{\text{pruning error}}+\underbrace{\mathscr{O}\big(\beta\eta G^2\cdot\varphi_{\mathrm{w},0}(\boldsymbol{\delta},\mathbf{f},\mathbf{P})\big)}_{\text{wireless factor}}. \quad (31)$$

Remark 5. *In (31), the third, fourth, fifth and sixth terms appear from the divergence between client-VC, VC-sBS, sBS-mBS and mBS-global loss functions, respectively.*

Remark 6. *In typical HFL with no model pruning, i.e.,* $\delta_u^t=0$*,* $\forall u\in\mathscr{U}$*,* $\mathscr{O}\big(\delta^{\mathrm{th}}\beta^2D^2\big)=0$*. Besides, when the wireless links are ignored, the last term in (31) becomes zero. In such a special case, Corollary 1 boils down to*

$$\theta_{\mathrm{PHFL}}\leq \mathscr{O}\big([f(\tilde{\mathbf{w}}^0)-f_{\mathrm{inf}}]/[\eta T]\big)+\mathscr{O}(\beta\eta\sigma^2/\mathrm{U})+ \mathscr{O}\big(\kappa_0^2\eta^2\beta^2\varepsilon_{\mathrm{vc}}^2\big)+\mathscr{O}\big(\kappa_0^2\kappa_1^2\eta^2\beta^2\varepsilon_{\mathrm{sbs}}^2\big)+ \mathscr{O}\big(\kappa_0^2\kappa_1^2\kappa_2^2\eta^2\beta^4\varepsilon_{\mathrm{mbs}}^2\big)+\mathscr{O}\big(\kappa_0^2\kappa_1^2\kappa_2^2\kappa_3^2\beta^4\eta^2\varepsilon^2\big). \quad (32)$$

Remark 7. *When* $\eta=\sqrt{\mathrm{U}/T}$*, we have* $T\geq 1/[72\mathrm{U}\kappa_0^2\kappa_1^2\kappa_2^2\kappa_3^2\beta^2]$*. With a sufficiently large* T*, when the trained model reception success probability is* 1 *for all users in all time steps, we have* $\theta_{\mathrm{PHFL}}\approx\mathscr{O}\big(1/\sqrt{\mathrm{U}T}\big)+\mathscr{O}\big(\delta^{\mathrm{th}}\beta^2D^2\big)$*, where the second term comes from the pruning error. Therefore, the proposed PHFL solution converges to the neighborhood of a stationary point of traditional HFL.*

IV. Joint Problem Formulation and Solution

Similar to existing literature [3], [4], [8], [27], we ignore the downlink delay in this paper since the sBS can utilize the higher spectrum and transmission power to broadcast the

updated model. Moreover, since the sBS-mBS and mBS-cloud server links are wired, we ignore the transmission delays for these links[5]. Furthermore, since the sBS, mBS and the cloud server usually have high computation power, we also ignore the model aggregation and processing delays[6]. Therefore, at the beginning of each VC round, i.e., $t \ni (m\prod_{z=0}^{\kappa_z}+t_0) \ mod \ \kappa_0 = 0$, we first calculate the required computation time for finding the lottery ticket as

$$\mathrm{t}_i^{\mathrm{cp_d}} = \rho \times \left(bnc_i \mathrm{D}_i / \mathrm{f}_i^t\right), \tag{33}$$

where b is the batch size, n is the number of batches, c_i is the CPU cycles to process 1-bit data, D_i is UE u_i's each data sample's size in bits and f_i^t is the CPU frequency. Upon finding the pruned model, each client performs κ_0 local iterations, which require the following computation time [23]

$$\mathrm{t}_i^{\mathrm{cp_s}} = \kappa_0 \times \left(bn(1-\delta_i^t)c_i \mathrm{D}_i / \mathrm{f}_i^t\right). \tag{34}$$

To that end, the UE only offloads the non-zero weights along with the binary mask to the sBS. As such, we calculate the uplink payload size of UE i as follows[7]:

$$s_i \leq d\left[1-\delta_i^t\right](\mathrm{FPP}+1)+d, \tag{35}$$

where FPP is the floating point precision. Note that, in (35), we need 1 bit to represent the sign of the entry. Therefore, we calculate the uplink payload offloading delay as follows:

$$\mathrm{t}_i^{\mathrm{up}} \leq d\left[\left(1-\delta_i^t\right)(\mathrm{FPP}+1)+1\right]/r_i^t. \tag{36}$$

As such, UE i's total duration for local computing and trained model offloading is

$$\mathrm{t}_i^{\mathrm{tot}} \leq \mathrm{t}_i^{\mathrm{cp_d}} + \mathrm{t}_i^{\mathrm{cp_s}} + \mathrm{t}_i^{\mathrm{up}}. \tag{37}$$

We now calculate the energy consumption for performing the model training, followed by the required energy for offloading the trained models. First, let us calculate the energy consumption to get the lottery ticket as

$$\mathrm{e}_i^{\mathrm{cp_d}} = \rho \times 0.5\xi bnc_i \mathrm{D}_i (\mathrm{f}_i^t)^2, \tag{38}$$

where 0.5ξ is the effective capacitance of UE's CPU chip. Similarly, we calculate the energy consumption to train κ_0 local iterations using the pruned model as

$$\mathrm{e}_i^{\mathrm{cp_s}} = \kappa_0 \times 0.5\xi bn(1-\delta_i^t)c_i \mathrm{D}_i (\mathrm{f}_i^t)^2. \tag{39}$$

Moreover, we calculate the uplink payload offloading energy consumption as follows:

$$\mathrm{e}_i^{\mathrm{up}} \leq d\left[\left(1-\delta_i^t\right)(\mathrm{FPP}+1)+1\right]\mathrm{P}_i^t / r_i. \tag{40}$$

Therefore, the total energy consumption is calculated as

$$\mathrm{e}_i^{\mathrm{tot}} \leq \mathrm{e}_i^{\mathrm{cp_d}} + \mathrm{e}_i^{\mathrm{cp_s}} + \mathrm{e}_i^{\mathrm{up}}. \tag{41}$$

[5]The transmissions over the wired sBS-mBS and mBS-cloud server links happen in the backhaul, and the corresponding delays are quite small. In order to calculate these delays, one should also consider the overall network loads, which are beyond the scope of this paper.

[6]The addition of d parameters and then taking the average have a time complexity of $\mathcal{O}(d+1)$. With highly capable CPUs at the sBS, mBS, and central server, the corresponding time delays for parameter aggregation are usually small and therefore ignored in the literature [9], [10], [23].

[7]Note that one may send the non-pruned weights and the corresponding indices, which are unknown until the original initial model is trained for ρ iterations. We consider an upper bound for the uplink payload, which will be used during the joint parameters optimization phase.

A. Problem Formulation

Denote the duration between VC aggregation t^{th}. Then, we calculate the probability of successful reception of UE's trained model as follows:

$$\begin{aligned} p_i^t &= \Pr\left\{\mathrm{t}_i^{\mathrm{tot}} \leq \mathrm{t}^{\mathrm{th}}\right\} = \Pr\left\{s_i \leq r_i^t\left[\mathrm{t}^{\mathrm{th}} - \mathrm{t}_i^{\mathrm{cp_d}} - \mathrm{t}_i^{\mathrm{cp_s}}\right]\right\} \\ &= \Pr\left\{h_{i,k}^t \geq \left[(2^{\chi_i^t}-1)(\omega\zeta^2 + I_{i,k}^t)/(\mathrm{P}_i^t d_{i,k}^{-\alpha})\right]\right\} \\ &\overset{(a)}{=} \exp\left[-(2^{\chi_i^t}-1)(\omega\zeta^2 + I_{i,k}^t)/(\mathrm{P}_i^t d_{i,k}^{-\alpha})\right], \end{aligned} \tag{42}$$

where $\chi_i^t = \frac{d\mathrm{f}_i^t\left[\left(1-\delta_i^t\right)(\mathrm{FPP}+1)+1\right]}{\omega\left[\mathrm{f}_i^t \mathrm{t}^{\mathrm{th}} - bnc_i \mathrm{D}_i\left(\rho+\kappa_0(1-\delta_i^t)\right)\right]}$ and (a) follows from the Rayleigh fading channels between the UE and the sBS.

Notice that the pruning ratio δ_i^t, CPU frequency f_i^t, transmission power P_i^t and the probability of successful model reception p_i^t are intertwined. More specifically, p_i^t depends on δ_i^t, f_i^t and P_i^t, given that the other parameters remain fixed. As such, we aim to optimize these parameters jointly by considering the controllable terms in our convergence bound in Corollary 1. Therefore, we focus on each VC round, i.e., the local iteration round t at which $t \ mod \ \kappa_0 = 0$. Specifically, we focus on minimizing the error terms due to pruning and wireless links, which are given by

$$\mathcal{O}\left(\delta^{\mathrm{th}}\beta^2 D^2\right) + \mathcal{O}\left(\beta\eta G^2 \cdot \varphi_{\mathrm{w},0}(\boldsymbol{\delta}, \mathbf{f}, \mathbf{P})\right). \tag{43}$$

Remark 8. *In the above expression, the first term appears from the pruning error* $\frac{12\beta^2}{T}\sum_{t=0}^{T-1}\sum_{l=1}^{L} \cdot \alpha_l \sum_{k=1}^{B_l}\alpha_k \sum_{j=1}^{V_{k,l}}\alpha_j \sum_{i=1}^{U_{j,k,l}}\alpha_i\big(1 + 2\big\{\alpha_i\big[1 + \alpha_j\big(1 + \alpha_k\big\{1 + \alpha_l\big\}\big)\big]\big\}\big)\delta_i^t\|\mathbf{w}_i^t\|^2 \leq \mathcal{O}\left(\delta^{\mathrm{th}}\beta^2 D^2\right)$, *while the second term comes from the wireless factor* $\frac{2\beta\eta}{T}\sum_{l=1}^{L}\alpha_l^2 \cdot \sum_{k=1}^{B_l}\alpha_k^2\sum_{j=1}^{V_{k,l}}\alpha_j^2\sum_{i=1}^{U_{j,k,l}}\alpha_i^2\sum_{t=0}^{T-1}(1/p_i^t - 1)\,\mathbb{E}\left\|\tilde{g}\left(\tilde{\mathbf{w}}_i^{t,0}\right)\right\|^2 \leq \mathcal{O}\left(\frac{\beta\eta G^2}{T}\sum_{l=1}^{L}\alpha_l^2\sum_{k=1}^{B_l}\alpha_k^2\sum_{j=1}^{V_{k,l}}\alpha_j^2\sum_{i=1}^{U_{j,k,l}}\alpha_i^2\sum_{t=0}^{T-1}(1/p_i^t - 1)\right)$.

Based on the above observations, we consider a weighted combination of these two terms as our objective function to minimize the bound in (43). Using (42) in the wireless factor, we, therefore, consider the following objective function.

$$\begin{aligned} \varphi^t(\boldsymbol{\delta}^t, \mathbf{f}^t, \mathbf{P}^t) = \phi_1 \sum_{l=1}^{L}\alpha_l \sum_{k=1}^{B_l}\alpha_k \sum_{j=1}^{V_{k,l}}\alpha_j \sum_{i=1}^{U_{j,k,l}}\alpha_i \delta_i^t + \\ \phi_2 \sum_{l=1}^{L}\alpha_l \sum_{k=1}^{B_l}\alpha_k \sum_{j=1}^{V_{k,l}}\alpha_j \sum_{i=1}^{U_{j,k,l}}\alpha_i \left[\exp\left(\frac{(2^{\chi_i^t}-1)(\omega\zeta^2 + I_{i,k}^t)}{\mathrm{P}_i^t d_{i,k}^{-\alpha}}\right) - 1\right], \end{aligned} \tag{44}$$

where ϕ_1 and ϕ_2 are two weights to strike the balance between the terms. Note that the wireless factor is multiplied by the learning rate and gradient in (43). Typically, the learning rate is small. Besides, the gradient becomes smaller as the training progresses. As such, the wireless factor term is relatively small when $p_i^t > 0$ for all UEs and VC aggregation rounds. The model weights are non-negative. Furthermore, a larger pruning ratio δ_i^t can dramatically reduce the computation and offloading time, making the wireless factor 0. However, as a higher pruning ratio means more model parameters are pruned, we wish to avoid making the δ_i^t's large to reduce the pruning-induced errors. The above facts suggest we put more weight on the pruning error term to penalize more for the δ_i^t's. As such, we consider $\phi_1 \gg \phi_2$. However, in our resource-constrained setting, a small δ_i^t can prolong the training and offloading time, leading p_i^t to be 0, i.e., the sBS will not receive the

local trained model. Therefore, although ϕ_2 is small, we keep the wireless factor to ensure p_i^t is never 0.

Therefore, we pose the following optimization problem to configure the parameters jointly.

$$\underset{\boldsymbol{\delta}^t,\mathbf{f}^t,\mathbf{P}^t}{\text{minimize}} \quad \varphi^t(\boldsymbol{\delta}^t,\mathbf{f}^t,\mathbf{P}^t), \tag{45}$$

$$\text{s.t.} \quad (C1) \quad \mathrm{t}_i^{\mathrm{tot}} \le \mathrm{t}^{\mathrm{th}}, \quad \forall i, \tag{45a}$$

$$(C2) \quad \mathrm{e}_i^{\mathrm{tot}} \le \mathrm{e}_i^{\mathrm{th}}, \quad \forall i, \tag{45b}$$

$$(C3) \quad 0 \le \mathrm{f}_i^t \le \mathrm{f}_i^{\max}, \quad \forall i, \tag{45c}$$

$$(C4) \quad 0 \le \mathrm{P}_i^t \le P_i^{\max}, \quad \forall i, \tag{45d}$$

$$(C5) \quad 0 \le \delta_i^t \le \delta^{\mathrm{th}}, \quad \forall i, \tag{45e}$$

where constraint $(C1)$ ensures that the completion of one VC round is within the required deadline. Constraint $(C2)$ controls the energy expense to be within the allowable budget. Besides, $(C3)$ and $(C4)$ restrict us from choosing the CPU frequency and transmission power within the UE's minimum and maximum CPU cycles and transmission power, respectively. Finally, constraint $(C5)$ ensures the pruning ratio to be within a tolerable limit δ^{th}.

Remark 9. *We assume that clients' system configurations remain unchanged over time, while the channel state information (CSI) is dynamic and known at the sBS. The clients share their system configurations with their associated sBS. The sBSs share their respective users' system configurations and CSI with the central server. As such, problem (45) is solved centrally, and the optimized parameters are broadcasted to the clients. Besides, problem (45) is non-convex with the multiplications and divisions of the optimization variables in the second term. Moreover, constraints (C1) and (C2) are not convex. Therefore, it is not desirable to minimize this original problem directly. In the following, we transform the problem into an approximate convex problem that can be solved efficiently.*

B. Problem Transformation

Let us define $A(\delta_i^t,\mathrm{f}_i,P_i) := \exp\big[(2^{\chi_i^t}-1)(\omega\zeta^2 + I_{i,k}^t)/(\mathrm{P}_i^t d_{i,k}^{-\alpha})\big]$. Given an initial feasible point set $(\delta_i^{t,q}, \mathrm{f}_i^{t,q}, \mathrm{P}_i^{t,q})$, we perform a linear approximation of this non-convex expression as follows:

$$\begin{aligned} A(\delta_i^t,\mathrm{f}_i,P_i) \approx & A(\delta_i^{t,q},\mathrm{f}_i^{t,q},\mathrm{P}_i^{t,q}) + \nabla_{\delta_i^t}\big[A(\delta_i^{t,q},\mathrm{f}_i^{t,q},\mathrm{P}_i^{t,q})\big](\delta_i^t-\delta_i^{t,q}) \\ & + \nabla_{\mathrm{f}_i^t}\big[A(\delta_i^{t,q},\mathrm{f}_i^{t,q},\mathrm{P}_i^{t,q})\big](\mathrm{f}_i^t-\mathrm{f}_i^{t,q}) + \\ & \nabla_{\mathrm{P}_i^t}\big[A(\delta_i^{t,q},\mathrm{f}_i^{t,q},\mathrm{P}_i^{t,q})\big](\mathrm{P}_i^t-\mathrm{P}_i^{t,q}) := \tilde{A}(\delta_i^t,\mathrm{f}_i^t,\mathrm{P}_i^t), \end{aligned} \tag{46}$$

where $A(\delta_i^{t,q},\mathrm{f}_i^{t,q},\mathrm{P}_i^{t,q}) = \exp\big[(2^{\chi_i^{t,q}}-1)(\omega\zeta^2 + \tilde{I}_{i,k}^t)/(\mathrm{P}_i^{t,q} d_{i,k}^{-\alpha})\big]$, $\chi_i^{t,q} = \frac{d\mathrm{f}_i^{t,q}\left[(1-\delta_i^t)(\mathrm{FPP}+1)+1\right]}{\omega\left[\mathrm{f}_i^{t,q}\mathrm{t}^{\mathrm{th}} - bnc_i\mathrm{D}_i\left(\rho+\kappa_0(1-\delta_i^{t,q})\right)\right]}$ and $\tilde{I}_{i,k}^t = \sum_{l=1}^{L}\sum_{k'=1,k\neq k'}^{K}\sum_{j'=1}^{J_{k',l}}\sum_{u_{i'}\in\mathscr{U}_{j',k',l}} \mathrm{P}_{i'}^{t,q} h_{i',k'} d_{i',k'}^{-\alpha}$. Moreover, $\nabla_{\delta_i^t}\big[A(\delta_i^{t,q},\mathrm{f}_i^{t,q},\mathrm{P}_i^{t,q})\big]$, $\nabla_{\mathrm{f}_i^t}\big[A(\delta_i^{t,q},\mathrm{f}_i^{t,q},\mathrm{P}_i^{t,q})\big]$ and $\nabla_{\mathrm{P}_i^t}\big[A(\delta_i^{t,q},\mathrm{f}_i^{t,q},\mathrm{P}_i^{t,q})\big]$ are calculated in (47), (48) and (49), respectively.

As such, we approximate (44) as follows:

$$\tilde{\varphi}^t(\boldsymbol{\delta}^t,\mathbf{f}^t,\mathbf{P}^t) = \sum_{l=1}^{L}\alpha_l\sum_{k=1}^{B_l}\alpha_k\sum_{j=1}^{V_{k,l}}\alpha_j\sum_{i=1}^{U_{j,k,l}}\alpha_i\big(\phi_1\delta_i^t + \phi_2\big[\tilde{A}(\delta_i^t,\mathrm{f}_i^t,\mathrm{P}_i^t)-1\big]\big), \tag{52}$$

where $\tilde{A}(\delta_i^t,\mathrm{f}_i^t,\mathrm{P}_i^t)$ is calculated in (46).

We now focus on the non-convex constraints. First, let us approximate the local pruned model computation time as

$$\mathrm{t}_i^{\mathrm{cp_s}} \approx [\kappa_0 bnc_i\mathrm{D}_i/\mathrm{f}_i^{t,q}](1-\delta_i^t-(1-\delta_i^{t,q})(\mathrm{f}_i^t-\mathrm{f}_i^{t,q})/\mathrm{f}_i^{t,q}) = \tilde{\mathrm{t}}_i^{\mathrm{cp_s}}. \tag{53}$$

Then, we approximate the non-convex uplink model offloading delay as shown in (50). Using a similar treatment, we write

$$\begin{aligned} \mathrm{e}_i^{\mathrm{cp_s}} \approx \kappa_0\xi bnc_i\mathrm{D}_i\mathrm{f}_i^{t,q}\big[(\delta_i^{t,q}-0.5)\mathrm{f}_i^{t,q} - 0.5\mathrm{f}_i^{t,q}\delta_i^t + \\ (1-\delta_i^{t,q})\mathrm{f}_i^t\big] := \tilde{\mathrm{e}}_i^{\mathrm{cp_s}}. \end{aligned} \tag{54}$$

Similarly, we approximate the energy consumption for model offloading as shown in (51).

Therefore, we pose the following transformed problem

$$\underset{\boldsymbol{\delta}^t,\mathbf{f}^t,\mathbf{P}^t}{\text{minimize}} \quad \tilde{\varphi}^t(\boldsymbol{\delta}^t,\mathbf{f}^t,\mathbf{P}^t), \tag{55}$$

$$\text{s.t.} \quad (\tilde{C}1) \quad \tilde{\mathrm{t}}_i^{\mathrm{cp_d}} + \tilde{\mathrm{t}}_i^{\mathrm{cp_s}} + \tilde{\mathrm{t}}_i^{\mathrm{up}} \le \mathrm{t}^{\mathrm{th}}, \tag{55a}$$

$$(\tilde{C}2) \quad \mathrm{e}_i^{\mathrm{cp_d}} + \tilde{\mathrm{e}}_i^{\mathrm{cp_s}} + \tilde{\mathrm{e}}_i^{\mathrm{up}} \le \mathrm{e}_i^{\mathrm{th}}, \tag{55b}$$

$$(45c), (45d), (45e), \tag{55c}$$

where the constraints are taken for the same reasons as in the original problem. Besides, $\tilde{\mathrm{t}}_i^{\mathrm{cp_d}} = 2\rho bnc_i\mathrm{D}_i/\mathrm{f}_i^{t,q} - \rho bnc_i\mathrm{D}_i\mathrm{f}_i^t/(\mathrm{f}_i^{t,q})^2$.

Note that problem (55) is now convex and can be solved iteratively using existing solvers such as CVX [30]. The key steps of our iterative solution are summarized in Algorithm 2. Moreover, as (55) has 3U decision variables and 5U constraints, the time complexity of running Algorithm 2 for Q iterations is $\mathscr{O}\big(Q\times[(3\mathrm{U})^3\times 5\mathrm{U}]\big)$ [31]. While Algorithm 2 yields a suboptimal solution and converges to a local stationary solution set of the original problem (45), SCA-based solutions are well-known for fast convergence [32]. Moreover, our extensive empirical study in the sequel suggests that the proposed PHFL solution with Algorithm 2 delivers nearly identical performance to the upper bounded performance.

Algorithm 2: Iterative Joint Pruning Ratio, CPU Frequency and Transmission Power Selection Process

Input: Initial feasible set $(\boldsymbol{\delta}^{t,0},\mathbf{f}^{t,0},\mathbf{P}^{t,0})$, maximum iteration Q, precision level $\varepsilon^{\mathrm{prec}}$; set $q=0$
Repeat:
  Solve (55) using $(\boldsymbol{\delta}^{t,q},\mathbf{f}^{t,q},\mathbf{P}^{t,q})$ to get the optimized $(\boldsymbol{\delta}^t,\mathbf{f}^t,\mathbf{P}^t)$
  $q \leftarrow q+1$; $\boldsymbol{\delta}^{t,q} \leftarrow \boldsymbol{\delta}^t$; $\mathbf{f}^{t,q} \leftarrow \mathbf{f}^t$; $\mathbf{P}^{t,q} \leftarrow \mathbf{P}^t$
Until converge with $\varepsilon^{\mathrm{prec}}$ precision or $q=Q$
Output: Optimal $(\boldsymbol{\delta}^t,\mathbf{f}^t,\mathbf{P}^t)$

V. Simulation Results and Discussions

A. Simulation Setting

For the performance evaluation, we consider $L=2$, $B=4$ and $\mathrm{U}=48$. We let each sBS maintain 2 VCs, where each VC has 6 UEs. In other words, we have $U_{j,k,l}=6,\forall j,k$ and l, $V_{k,l}=2,\forall k$ and l, and $B_l=2,\forall l$. We assume $\omega=1$ megahertz (MHz). We randomly generate maximum transmission power $P^{\max}$, energy budget for each VC aggregation round e^{th}, CPU frequency $\mathrm{f}^{\max}$ and required CPU cycle to process per-bit data c, respectively, from $[23,30]$ dBm, $[10,13]$ Joules, $[1.8,2.8]$

$$\nabla_{\delta_i^t}\left[A(\delta_i^{t,q},\mathrm{f}_i^{t,q},\mathrm{P}_i^{t,q})\right] = \{\ln(2)2^{\chi_i^{t,q}} d\mathrm{f}_i^{t,q} A(\delta_i^{t,q},\mathrm{f}_i^{t,q},\mathrm{P}_i^{t,q})(\omega\zeta^2+\tilde{I}_{i,k}^t)[bnc_iD_i(\rho(\mathrm{FPP}+1)-\kappa_0)- \mathrm{f}_i^{t,q}\mathrm{t}^{\mathrm{th}}(\mathrm{FPP}+1)]\}/\{\omega\mathrm{P}_i^{t,q}d_{i,k}^{-\alpha}\times[\mathrm{f}_i^{t,q}\mathrm{t}^{\mathrm{th}}-bnc_iD_i(\rho+\kappa_0(1-\delta_i^{t,q}))]^2\}. \quad (47)$$

$$\nabla_{\mathrm{f}_i}\left[A(\delta_i^{t,q},\mathrm{f}_i^{t,q},\mathrm{P}_i^{t,q})\right] = \{-\ln(2)2^{\chi_i^{t,q}} bnc_i d D_i A(\delta_i^{t,q},\mathrm{f}_i^{t,q},\mathrm{P}_i^{t,q})(\omega\zeta^2+\tilde{I}_{i,k}^t)[(1-\delta_i^{t,q})(\mathrm{FPP}+1)+1]\times (\rho+\kappa_0[1-\delta_i^{t,q}])\}/\{\omega\mathrm{P}_i^{t,q}d_{i,k}^{-\alpha}\times[\mathrm{f}_i^{t,q}\mathrm{t}^{\mathrm{th}}-bnc_iD_i(\rho+\kappa_0(1-\delta_i^{t,q}))]^2\}. \quad (48)$$

$$\nabla_{P_i}\left[A(\delta_i^{t,q},\mathrm{f}_i^{t,q},\mathrm{P}_i^{t,q})\right] = -A(\delta_i^{t,q},\mathrm{f}_i^{t,q},\mathrm{P}_i^{t,q})(2^{\chi_i^{t,q}}-1)(\omega\zeta^2+\tilde{I}_{i,k}^t)/[(P_i^{t,q})^2 d_{i,k}^{-\alpha}]. \quad (49)$$

$$\mathrm{t}_i^{\mathrm{up}} \approx \frac{d\left(2+\mathrm{FPP}-(1+\mathrm{FPP})\delta_i^t\right)}{\omega\log_2(1+\mathrm{P}_i^{t,q}h_{i,k}d_{i,k}^{-\alpha}/[\omega\zeta^2+\tilde{I}_{i,k}^t])} + \frac{-\ln(2)dh_{i,k}d_{i,k}^{-\alpha}[(1-\delta_i^{t,q})(\mathrm{FPP}+1)+1]\times(\mathrm{P}_i^t-\mathrm{P}_i^{t,q})}{\omega\{\ln(1+[\mathrm{P}_i^{t,q}h_{i,k}d_{i,k}^{-\alpha}]/[\omega\zeta^2+\tilde{I}_{i,k}^t])\}^2(\omega\zeta^2+\tilde{I}_{i,k}^t+\mathrm{P}_i^{t,q}h_{i,k}d_{i,k}^{-\alpha})} := \tilde{\mathrm{t}}_i^{\mathrm{up}}. \quad (50)$$

$$\mathrm{e}_i^{\mathrm{up}} \approx \{d\mathrm{P}_i^{t,q}[(\mathrm{FPP}+2)-(\mathrm{FPP}+1)\delta_i^t]\}/\{\omega\log_2(1+\mathrm{P}_i^{t,q}h_{i,k}d_{i,k}^{-\alpha}/(\omega\zeta^2+\tilde{I}_{i,k}^t))\}+ \frac{d[(1-\delta_i^{t,q})(\mathrm{FPP}+1)+1]\left[\log_2\left(1+\frac{\mathrm{P}_i^{t,q}h_{i,k}d_{i,k}^{-\alpha}}{\omega\zeta^2+\tilde{I}_{i,k}^t}\right)-\frac{\mathrm{P}_i^{t,q}h_{i,k}d_{i,k}^{-\alpha}}{\ln(2)(\omega\zeta^2+\tilde{I}_{i,k}^t+\mathrm{P}_i^{t,q}h_{i,k}d_{i,k}^{-\alpha})}\right]}{\omega\{\log_2(1+\mathrm{P}_i^{t,q}h_{i,k}d_{i,k}^{-\alpha}/(\omega\zeta^2+\tilde{I}_{i,k}^t))\}^2}(\mathrm{P}_i^t-\mathrm{P}_i^{t,q}) = \tilde{\mathrm{e}}_i^{\mathrm{up}}. \quad (51)$$

gigahertz (GHz) and $[20,25]$ for these two VCs. Therefore, all UEs in a VC have the above randomly generated system configurations[8]. Moreover, as described earlier in Section II, our proposed PHFL has 4 tiers, namely (1) UE-VC, (2) VC-sBS, (3) sBS-mBS, and (4) mBS-central server.

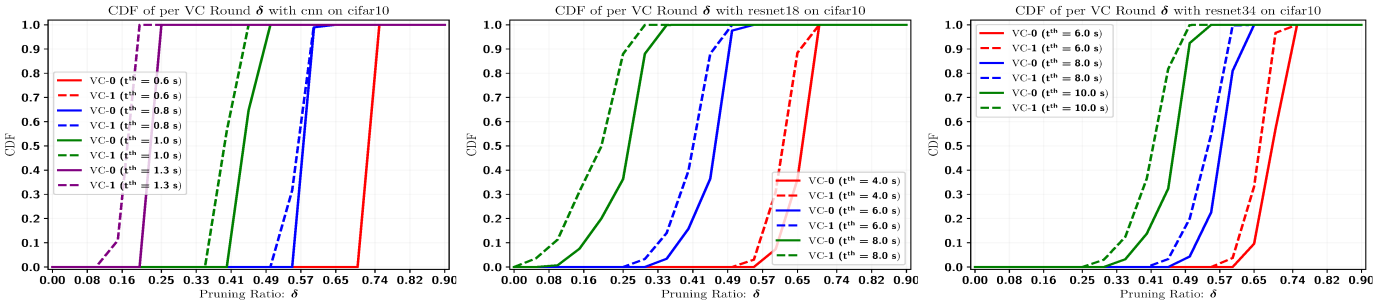
For our ML task, we use image classification with the popular CIFAR-10 and CIFAR-100 datasets [33] for performance evaluation. We use symmetric Dirichlet distribution $\mathrm{Dir}(\bar{\alpha})$ with concentration parameter $\bar{\alpha}$ for the non-IID data distribution as commonly used in literature [4], [27]. Besides, we use 1) convolutional neural network (CNN), 2) residual network (ResNet)-18 [34] and 3) ResNet-34 [34]. The CNN model has the following architecture: Conv2d(3,128), MaxPool2d, Conv2d(128, 64), MaxPool2d, Linear(256, 256), Linear(256, *#Labels*), whereas the ResNets have a similar architecture as in the original paper [34]. Moreover, the total number of trainable parameters depends on various configurations, such as the input/output shapes, kernel sizes, strides, etc. In our implementation, the original CNN, ResNet-18 and ResNet-34 models, respectively, have $151,882$; $6,992,138$ and $12,614,794$ trainable parameters on CIFAR-10, and $175,012$; $7,038,308$ and $12,660,964$ trainable parameters on CIFAR-100. Besides, with $\mathrm{FPP}=32$, we have a wireless payload of about 5.01 megabits (Mbs), 230.7 Mbs and 416.3 Mbs for CIFAR-10, and 5.8 Mbs, 232.3 Mbs and 417.8 Mbs for CIFAR-100 datasets for the respective three original models.

B. Performance Study

First, we investigate the pruning ratios δ_i^t's in different VCs. When the system configurations remain the same, the pruning ratio depends on the deadline threshold t^{th}. More specifically, a larger deadline allows the client to prune fewer model parameters, given that the energy constraint is satisfied. Intuitively, less pruning leads to a bulky model that takes longer training time. The CNN model is shallower compared to the ResNets. More specifically, the original non-pruned ResNet-18 and ResNet-34 models have about 46 times and 83 times the trainable parameters of the CNN model, respectively, on CIFAR-10. Therefore, the clients require a larger t^{th} to perform their local training and trained model offloading as the trainable parameters increase.

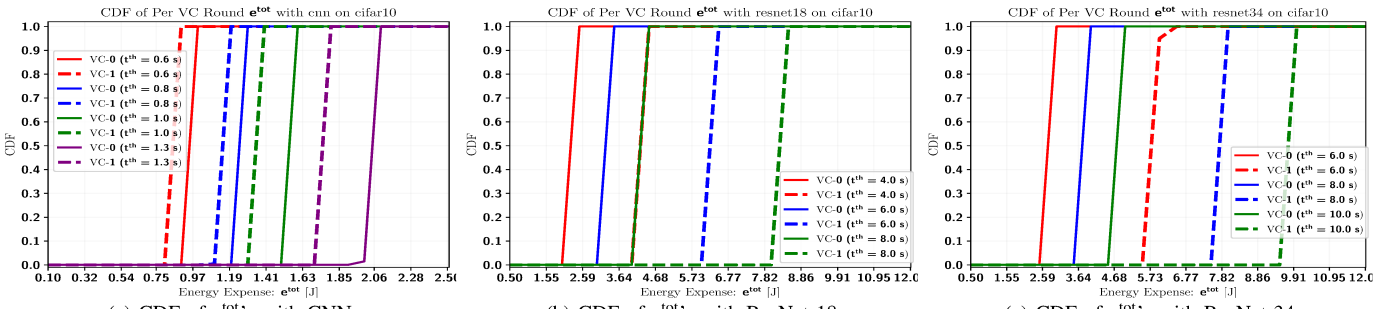
Intuitively, given a fixed t^{th}, the clients need to prune more model parameters for a bulky model in order to meet the deadline and energy constraints. Our simulation results also show that this general intuition holds in determining the δ_i^t's, as shown in Fig. 2, which show the cumulative distribution function (CDF) of the δ_i^t's in different VCs. It is worth noting that the pruning ratios δ_i^t's in each VC aggregation rounds are not deterministic due to the randomness of the wireless channels. We know the optimal variables once we solve the optimization problem in (55), which depends on the realizations of the wireless channels. Then, for a given VC j, we generate the plot by calculating $\frac{\sum_{l=1}^{L}\sum_{k=1}^{B_l}\sum_{i=1}^{U_{j,k,l}}\mathbf{1}(\delta_i^t\le\delta)}{\sum_{l=1}^{L}\sum_{k=1}^{B_l}\sum_{i=1}^{U_{j,k,l}}i}$, where $\mathbf{1}(\delta_i^t\le\delta)$ is an indicator function that takes value 1 if $\delta_i^t\le\delta$ and 0 otherwise. With the CNN model, about 50% clients have a δ_i^t less than 0.23, 0.43, 0.58 and 0.72 in VC-0 in all cells, for 1.3s, 1s, 0.8s and 0.6s deadline thresholds, respectively, in Fig. 2a. Note that we use $\delta^{\mathrm{th}}=0.9$, i.e., the clients can prune up to 90% of the neurons. Moreover, we consider $\mathrm{t}^{\mathrm{th}}=4$s and $\mathrm{t}^{\mathrm{th}}=6$s, to make the problem feasible for all clients for the ResNet-18 and ResNet-34 models, respectively. Furthermore, from Fig. 2a - Fig. 2c, it is quite clear that the UEs in VC-1 have to prune slightly lesser model parameters than the UEs in VC-0, even though the maximum CPU frequency $\mathrm{f}^{\max}$ of the UEs in VC-0 is 2.58 GHz, which is about 6.22% higher than the UEs in VC-1. However, due to the wireless payloads in the offloading phase, the transmission powers of the clients can also influence the δ_i^t's. In our setting, the UEs' maximum transmission powers are 0.35 Watt and 0.95 Watt, respectively, in VC-0 and VC-1. As such, with a similar wireless channel, the UEs in VC-1 can offload much faster than the UEs in VC-0. The above observations, thus,

[8]Our approach can easily be extended where all clients can have random $\mathrm{f}_i^{\max}$'s, $\mathrm{e}_i^{\mathrm{th}}$'s and $P_i^{\max}$'s. Our approach is practical since these parameters depend on the clients' manufacturers and their specific models.



(a) CDF of δ_u^t's with CNN (b) CDF of δ_u^t's with ResNet-18 (c) CDF of δ_u^t's with ResNet-34

Fig. 2: CDF of clients' pruning ratios in different VCs for different t^{th} with different ML models



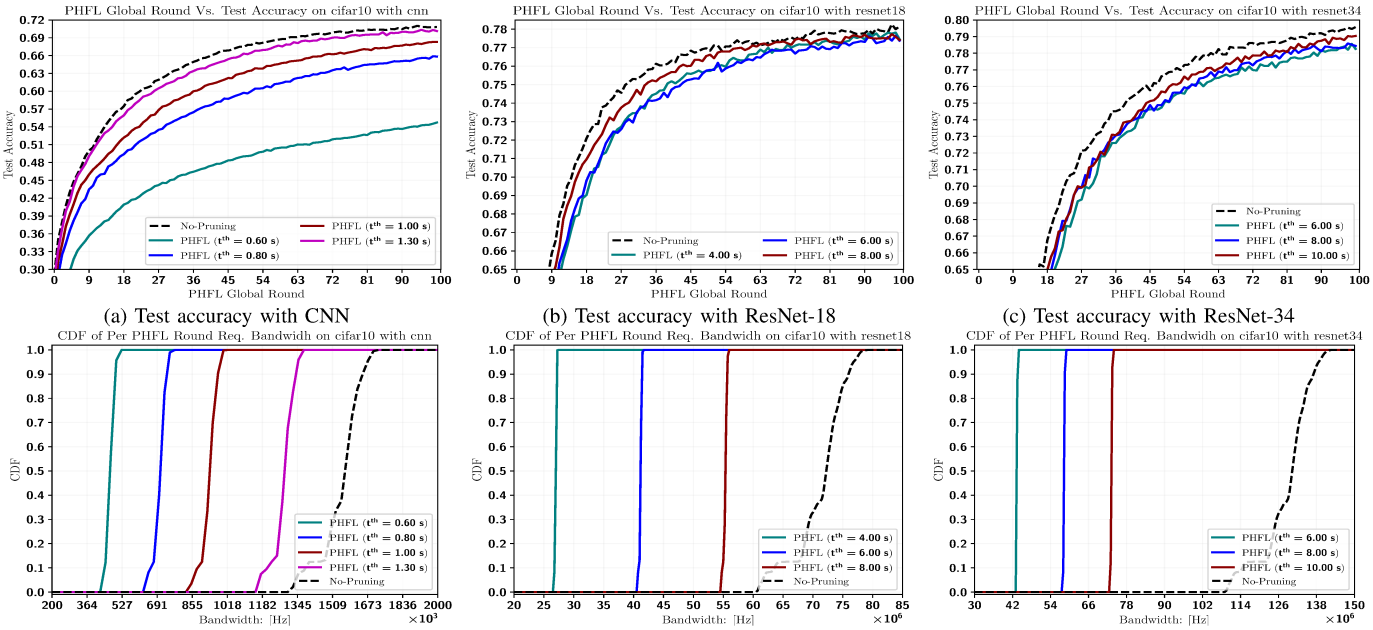
(a) CDF of e_i^{tot}'s with CNN (b) CDF of e_i^{tot}'s with ResNet-18 (c) CDF of e_i^{tot}'s with ResNet-34

Fig. 3: CDF of clients' e_i^{tot}'s in different VCs for different t^{th} with different ML models

point out that trained model offloading time t_i^{up} dominates the total time to finish one VC round t_i^{tot}.

When it comes to energy expense, from (39) and (40), it is quite obvious that a high δ_i^t shall lead to less energy consumption for both the training and offloading. However, the total energy expense of the clients boils down to the dominating factor between the required energy for computation and trained model offloading due to the interplay between the wireless and the learning parameters. Particularly, with $\omega = 1$ MHz, the clients can offload the CNN model fast, leading the computational energy consumption to be the dominating factor. The ResNets, on the other hand, have huge wireless overheads, leading the offloading time and energy be the dominating factors. This is also observed in our results in Fig. 3, which is the CDF plot of the energy expense e_i^{tot}, calculated in (41), of the clients in each VC and is generated following a similar strategy as in Fig. 2. When the CNN model is used, the total energy cost of the clients in VC-0 is larger, even though they prune more parameters, compared to the clients in VC-1 since larger f_i^{max}'s of the clients in VC-0 leads to a higher computational energy cost. This, however, changes for the ResNets since the wireless communication burden dominates the computation burden. The clients in VC-1 can use their higher P_i^{max}'s for the offloading time reduction when they determine the pruning ratios δ_i^t's. As such, the total energy expenses of the clients in VC-1 are much larger than the ones in VC-0. Our simulation results in Fig. 3a and Figs. 3b-3c also reveal the same trends.

Now, we observe the impact of δ_i^t's on the test accuracies and required bandwidth for trained model offloading. Intuitively, if the model is shallow, pruning further makes it shallower. Therefore, the test performance can exacerbate if δ_i^t's increases for a shallow model. On the other hand, for a bulky model, pruning may have a less severe effect. Specifically, under the deadline and energy constraints, pruning may eventually help because pruning a few neurons leads to a shallower but still reasonably well-constructed model that can be trained more efficiently. Moreover, our convergence bound in (31) clearly shows that the increasing δ_i^t's decreases the convergence speed. However, the wireless payload is directly related to δ_i^t's as shown in (35). Particularly, the wireless payload is an increasing function of the δ_i^t's. As such, increasing the deadline threshold t^{th} should decrease the δ_i^t's but significantly increase the wireless payload size. Our simulation results also reveal these trends in Fig. 4. Particularly, we observe that the CNN model is largely affected by a small t^{th} because that leads to a large δ_i^t, which eventually prunes more neurons of the already shallow model. On the other hand, the bulky ResNets exhibit small performance degradation when t^{th} decreases. Moreover, compared to the original non-pruned counterparts, the performance difference is small if t^{th} is selected appropriately, as shown in Fig. 4a to Fig. 4c. However, even a slight increase in d_p shall reduce the wireless payload size, which can significantly save a large portion of the bandwidth, as observed in Fig. 4d to Fig. 4f. For example, if the CNN model is used, with $t^{th} = 1.3$s, the performance degradation on test accuracy is about 0.97% after 100 PHFL round, while the per PHFL round bandwidth saving for at least 70% of the clients is about 18.3%. Similarly, if $t^{th} = 6$s, the test accuracy degradation is about 0.99% and 1.65% after 100 global rounds, while the per PHFL round bandwidth saving for at least 70% of the clients are about 43.8% and 67.2%, for ResNet-18 and ResNet-34, respectively.



(a) Test accuracy with CNN (b) Test accuracy with ResNet-18 (c) Test accuracy with ResNet-34

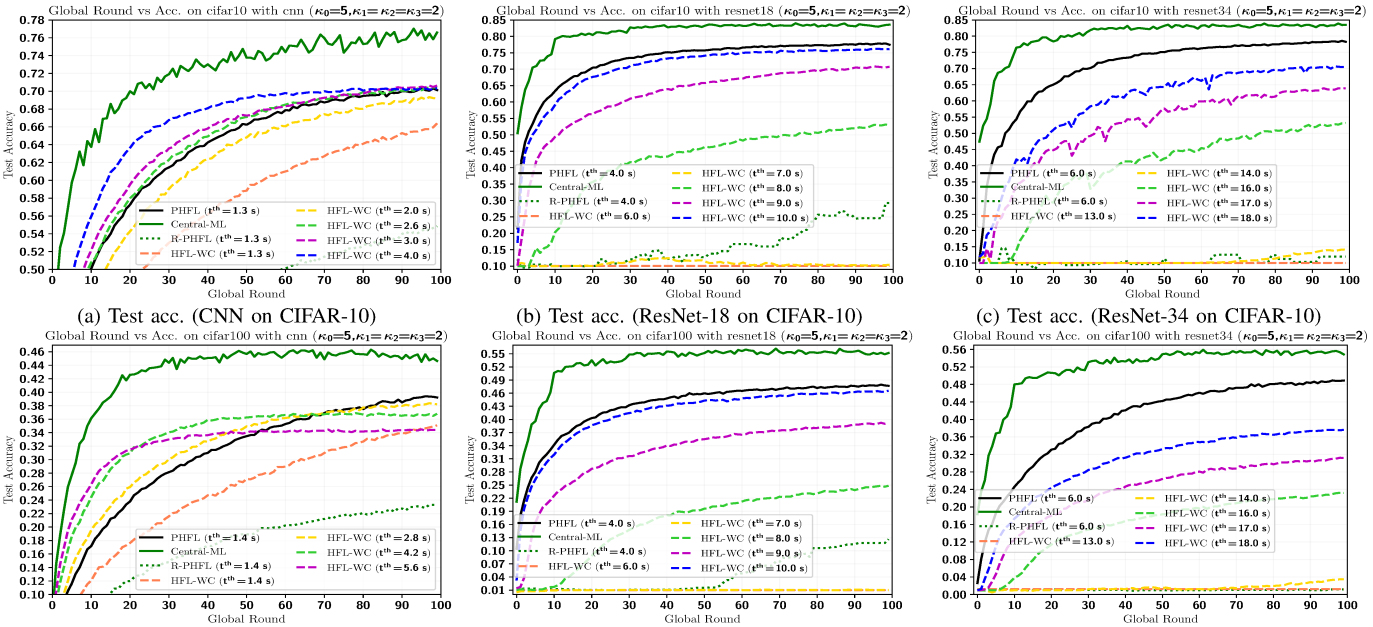
(d) Required bandwidth with CNN (e) Required bandwidth with ResNet-18 (f) Required bandwidth with ResNet-34

Fig. 4: Trade-offs between test accuracies and required bandwidth for different t^{th}'s with different ML models

C. Baseline Comparisons

We now focus on performance comparisons. First, we consider the existing HFL [9], [10] algorithm that does not consider model pruning or any energy constraints. Besides, [9], [10] only considered two levels, UE-BS and BS-cloud/server. For a fair comparison, we adapt HFL into three levels 1) UE-sBS, 2) sBS-mBS and 3) mBS-cloud/server. Furthermore, we enforce the energy and deadline constraints in each UE-sBS aggregation round and name this baseline HFL with constraints (HFL-WC). Moreover, since [9], [10] did not have any VC and we have κ_1 VC aggregation rounds before the sBS aggregation round, we have adapted the deadline accordingly for HFL-WC to make our comparison fair. Furthermore, we consider a random PHFL (R-PHFL) scheme, which has the same system model as ours and allows model pruning. In R-PHFL, the pruning ratios, δ_i^t's, are randomly selected between 0 and δ^{th} to satisfy constraint (45e). Moreover, in both HFL-WC and R-PHFL, a common κ_0 that satisfies both deadline and energy constraints for all clients in all VCs leads to poor test accuracy. As such, we determine the local iterations of the UEs within a VC by selecting the maximum possible number of iterations that all clients within that VC can perform without violating the delay and energy constraints. For our proposed PHFL, we choose $\kappa_0 = 5$ and $\kappa_1 = 2$. Moreover, we let $\kappa_2 = \kappa_3 = 2$ for both HFL-WC and PHFL. We also consider centralized SGD to show the performance gap of PHFL with the ideal case where all training data samples are available centrally.

From our above discussion, it is expected that pruning will likely not have the edge over the HFL-WC with a shallow model. Moreover, one may not need pruning for a shallow model in the first place. However, for a bulky model, due to a large number of training parameters and a huge wireless payload, pruning can be a necessity under extreme resource constraints. Furthermore, it is crucial to jointly optimize δ_i^t's, f_i^t's and P_i^t's in order to increase the test accuracy. The simulation results in Fig. 5 also validate these claims. We observe that when the t^{th} increases, HFL-WC's performance improves with the CNN model. Moreover, when HFL-WC has κ_1 times the deadline of PHFL, the performance is comparable, as shown in Fig. 5a and Fig. 5d. More specifically, the maximum performance degradation of the proposed PHFL algorithm is about 0.71% on CIFAR-10 when HFL-WC has 2.31 times the deadline of PHFL. However, for the ResNets model, HFL-WC requires a significantly longer deadline threshold to make the problem feasible. Particularly, with ResNet-18, $t^{th} \leq 6$s does not allow the UEs to perform even a single local iteration, leading to the same initial model weights and, thus, the same test accuracy in HFL-WC. Moreover, when the deadline threshold is κ_1 times the t^{th} of the PHFL, HFL-WC's test accuracy significantly lags, as shown in Fig. 5b and Fig. 5e. Particularly, after $T = 100$ rounds, our proposed solution with $t^{th} = 4$s provides about 46.43%, 9.5% and 1.78%, and about 92.03%, 21.6% and 2.38% better test accuracy on CIFAR-10 and on CIFAR-100, respectively, than the HFL-WC with $t^{th} = 8$s, $t^{th} = 9$s and $t^{th} = 10$s. For the ResNet-34 model, the clients require $t^{th} \geq 14$s for performing some local training in HFL-WC, whereas our proposed solution can achieve significantly better performance with only $t^{th} = 6$s. For example, our proposed solution with $t^{th} = 6$s yields about 47.19%, 22.43% and 10.99%, and about 110.73%, 56.7% and 30.06% better test accuracy than the HFL-WC with $t^{th} = 16$s, $t^{th} = 17$s and $t^{th} = 18$s on CIFAR-10 and CIFAR-100, respectively. Moreover, our proposed solution provides about 165% and 278.11% and about 553.26% and 4078.63% better test accuracy than R-PHFL on CIFAR-10 and CIFAR-100 with the ResNet-18 and ResNet-34 models, respectively. The gap with the ideal centralized ML is also expected since



(a) Test acc. (CNN on CIFAR-10) (b) Test acc. (ResNet-18 on CIFAR-10) (c) Test acc. (ResNet-34 on CIFAR-10)

(d) Test acc. (CNN on CIFAR-100) (e) Test acc. (ResNet-18 on CIFAR-100) (f) Test acc. (ResNet-34 on CIFAR-100)

Fig. 5: Test accuracies for different t^{th}'s with different ML models on different datasets

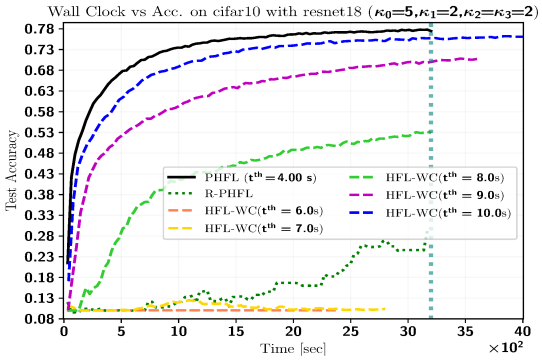


Fig. 6: Wall clock vs test accuracy on CIFAR-10 with ResNet-18

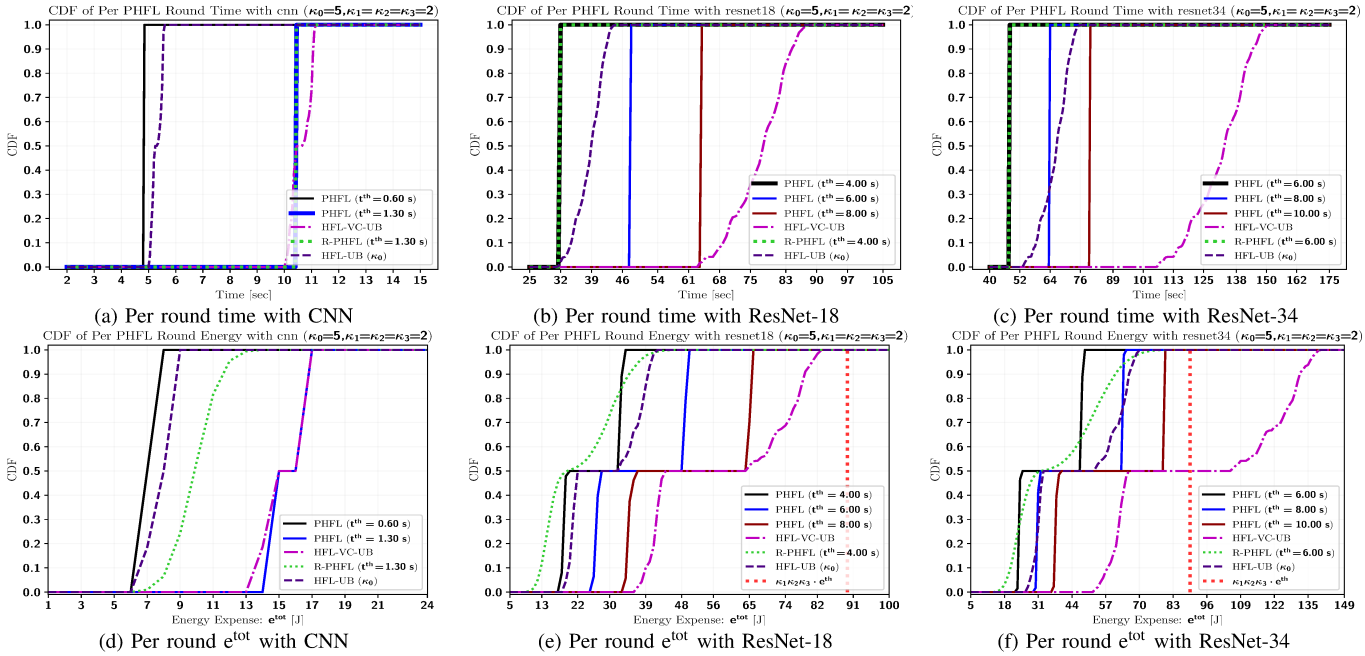
FL suffers from data and system heterogeneity.

We also examine the performance of the proposed method in terms of wall clock time. Since HFL-WC does not have the VC tier, the wall clock time to run M global rounds for HFL-WC with a deadline smaller than $\kappa_1 \times t^{th}$ will be lower than the proposed PHFL's wall clock time to run the same number of global rounds. However, that does not necessarily guarantee a higher test accuracy than our PHFL since training and offloading the original bulky model may take a long time, allowing only a few local SGD rounds of the clients. Besides, any deadline greater than $\kappa_1 \times t^{th}$ for the HFL-WC will require a longer wall clock time than our proposed PHFL solution. Our simulation results in Fig. 6 clearly shows these trends. We observe that when HFL-WC has a deadline $\kappa_1 \times t^{th}$, i.e., 2×4s = 8s, the test accuracies are about 48% and 73%, respectively, for HFL-WC and PHFL when the wall clock reaches 1800 seconds. Even with $t^{th} > 4\kappa_1$ seconds, the HFL-WC algorithm performs worse than our proposed PHFL solution.

From the above results and discussion, it is quite clear that R-PHFL yields poor test accuracy due to the random selection of δ_i^t's. Besides, HFL-WC cannot deliver reasonable performance when the model has a large number of training parameters. Furthermore, our proposed PHFL's performance is comparable to the non-pruned HFL-WC performance with the shallow model. As such, in the following, we only consider the upper bound (UB) of the HFL baseline, which does not consider the constraints. Moreover, to show how pruning degrades test accuracy, we also consider the UB, called HFL-VC-UB, by inheriting the same system model described in Section II, but without the model pruning and the constraints. In other words, we inherit the same underlying four levels, UE-VC, VC-sBS, sBS-mBS and mBS-cloud/server aggregation policy with the original non-pruned model.

First, we illustrate the required computation time and the corresponding energy expenses for the baselines and our proposed PHFL solution with different deadline thresholds. Naturally, if we increase the t^{th} for each VC aggregation round, our proposed solution will take longer time and energy to perform $T = 100$ global rounds. Moreover, since there are κ_1 VC aggregation rounds in our proposed system model, the original model training and offloading with HFL-VC-UB is expected to take significantly longer and consume more energy than the HFL-UB baseline. Therefore, the effectiveness, in terms of time and energy consumption, of PHFL is largely dependent upon the deadline threshold t^{th}. While R-PHFL requires the same time as our proposed PHFL, the energy requirements for R-PHFL can vary significantly due to the random selection of the δ_i^t's that lead to different model sizes and different local training episodes in different VCs.

Our results in Figs. 7a-7c and Figs. 7d-7f clearly show these trade-offs with time and energy consumption, respectively, for the three models. It is worth pointing out that we adopted the popular lottery ticket hypothesis [26] for finding the winning ticket that requires performing ρ iterations on the initial model



(a) Per round time with CNN (b) Per round time with ResNet-18 (c) Per round time with ResNet-34

(d) Per round e^{tot} with CNN (e) Per round e^{tot} with ResNet-18 (f) Per round e^{tot} with ResNet-34

Fig. 7: CDF of per PHFL round time and energy consumption with different ML models on CIFAR-10

TABLE II: Test Accuracy with Trained $\mathbf{w}^T$ on CIFAR-10 dataset with $\kappa_0 = 5, \kappa_1 = \kappa_2 = \kappa_3 = 2$ and $T = 100$

Energy Budget	Methods	Dir($\tilde{\alpha}$)	With CNN Model			With ResNet-18 Model			With ResNet-34 Model		
			Acc	Req T [s]	Req E [J]	Acc	Req T [s]	Req E [J]	Acc	Req T [s]	Req E [J]
429612	PHFL (Ours)	0.5	0.6867	1040	73172	0.7576	3200	122870	0.7662	4800	171079
		0.9	0.7013		73170	0.7746		122872	0.7826		171079
		10	0.7117		73171	0.7930		122871	0.7989		171080
	HFL-VC (UB) (Ours)	0.5	0.6928	1112.21	73837	0.7712	8712.91	277699	0.7761	14959.43	445248
		0.9	0.7081	1112.42	73837	0.7815	8728.63	277697	0.7957	14965.79	445249
		10	0.7226	1112.07	73837	0.7982	8710.87	277725	0.8028	14967.85	445281
	R-PHFL	0.5	0.5435	1040	47512	0.2559	3200	108302	0.1447	4800	183015
		0.9	0.5483		47448	0.2923		108170	0.1198		183821
		10	0.6067		47400	0.3562		107963	0.1262		183129
	HFL (UB) - κ_0	0.5	0.6617	556.28	36918	0.7469	4360.46	138854	0.7406	7486.32	222615
		0.9	0.6745	556.3	36919	0.7719	4368.01	138852	0.7622	7488.2	222613.54
		10	0.6949	556.03	36919	0.7853	4358.69	138863	0.7826	7485.64	222644

TABLE III: Test Accuracy with Trained $\mathbf{w}^T$ on CIFAR-10 dataset with $\kappa_0 = 10, \kappa_1 = \kappa_2 = \kappa_3 = 2$ and $T = 100$

Energy Budget	Methods	Dir($\tilde{\alpha}$)	With CNN Model			With ResNet-18 Model			With ResNet-34 Model		
			Acc	Req T [s]	Req E [J]	Acc	Req T [s]	Req E [J]	Acc	Req T [s]	Req E [J]
429612	PHFL (Ours)	0.5	0.6922	1600	116082	0.7643	3200	134505	0.776	4800	182445
		0.9	0.6983		116081	0.7821		134506	0.7945		182450
		10	0.7099		116081	0.7955		134511	0.8063		182446
	HFL-VC (UB) (Ours)	0.5	0.6947	2055.76	143148	0.7672	9661.69	347012	0.7827	15913.01	514544
		0.9	0.7108	2055.62	143147	0.7879	9657.30	346984	0.7978	15924.50	514579
		10	0.7185	2055.62	143147	0.7934	9658.81	347011	0.8119	15922.98	514558
	HFL (UB)- κ_0	0.5	0.6885	1028.03	71574	0.7653	4832.60	173501	0.7711	7960.03	257292
		0.9	0.6984	1027.83	71574	0.7817	4829.18	173494	0.7849	7962.99	257280
		10	0.7197	1027.91	71574	0.7929	4828.19	173508	0.7979	7971.59	257295

with full parameter space as described in Section II-B. This incurs additional t_i^{cpd} time and e_i^{cpd} energy overheads, as calculated in (33) and (38), respectively. As such, when the t^{th} is large, if the UEs have sufficient energy budgets, they can prune only a few neurons. Therefore, the total time and energy consumption for the original model computation for getting the winning ticket, training the pruned model and offloading the trained pruned model parameters can become slightly larger than the HFL-VC-UB. This is observed in Fig. 7a and Fig. 7d for CNN model when $t^{th} = 1.3$s, and also in Fig. 7b when $t^{th} = 10$s for the ResNet-18 model. Besides, the dashed vertical lines in Fig. 7e and Fig. 7f show the mean energy budget of the clients for each PHFL global round. While all UEs are able to perform the learning and offloading within this mean energy budget with CNN and ResNet-18 models, clearly, when the ResNet-34 model is used, more than 50% of the clients will fail to perform the HFL-VC-UB due to their limited energy budgets.

Finally, we show the impact of κ_0 and κ_1 for different dataset heterogeneity levels on CIFAR-10 dataset in Table II - Table IV, where $t^{th} = 4$s and $t^{th} = 6$s is used in our proposed PHFL algorithms for ResNet-18 and ResNet-34 models, respectively. Besides, for the CNN model, we used $t^{th} = 1.3$s and 2s, respectively, in our PHFL algorithm when $\kappa_0 = 5$ and $\kappa_0 = 10$, respectively, due to the facts that pruning exacerbates test performance for a shallow model and computational time dominates the offloading delay. Similarly, we considered $t^{th} = 1.4$s and $t^{th} = 2.5$s, respectively, for $\kappa_0 = 5$

TABLE IV: Test Accuracy with Trained $\mathbf{w}^T$ on CIFAR-10 dataset with $\kappa_0 = 5, \kappa_1 = 4, \kappa_2 = \kappa_3 = 2$ and $T = 100$

Energy Budget	Methods	Dir($\bar{\alpha}$)	With CNN Model			With ResNet-18 Model			With ResNet-34 Model		
			Acc	Req T [s]	Req E [J]	Acc	Req T [s]	Req E [J]	Acc	Req T [s]	Req E [J]
859224	PHFL (Ours)	0.5	0.6988	2080	146341	0.7690	6400	245740	0.7809	9600	342161
		0.9	0.7099		147675	0.7822		245743	0.7986		342157
		10	0.7192		147674	0.7937		245741	0.8094		342158
	HFL-VC (UB) (Ours)	0.5	0.7038	2224.42	147673	0.7667	17438.07	555371	0.7853	29915.05	890465
		0.9	0.7128	2224.42	147675	0.7849	17432.27	555439	0.8014	29928.13	890540
		10	0.7177	2224.69	147674	0.7951	17439.30	555384	0.8108	29921.44	890519
	R-PHFL	0.5	0.6066	2080	94998	0.4262	6400	215254	0.1005	9600	366584
		0.9	0.6196		147673	0.4147		216364	0.1262		366109
		10	0.6499		94684	0.5220		215769	0.1106		367140

TABLE V: Test Accuracy with Trained $\mathbf{w}^T$ on CIFAR-100 dataset with $\kappa_0 = 5, \kappa_1 = \kappa_2 = \kappa_3 = 2$ and $T = 100$

Energy Budget	Methods	Dir($\bar{\alpha}$)	With CNN Model			With ResNet-18 Model			With ResNet-34 Model		
			Acc	Req T [s]	Req E [J]	Acc	Req T [s]	Req E [J]	Acc	Req T [s]	Req E [J]
429612	PHFL (Ours)	0.5	0.3894	1120	77960	0.4726	3200	122775	0.4729	4800	171032
		0.9	0.3922		77960	0.4768		122780	0.4889		171029
		10	0.3938		77961	0.4816		122778	0.4824		171031
	HFL-VC (UB) (Ours)	0.5	0.3982	1138.04	74527	0.4798	8772.6	279067	0.4909	15026.08	446611
		0.9	0.3966	1138	74526	0.4835	8762.76	279092	0.4893	15029.89	446616
		10	0.3950	1137.88	74527	0.4781	8764.94	279068	0.4894	15021.95	446578
	R-PHFL	0.5	0.2040	1120	50732	0.0916	3200	108627	0.0105	4800	183279
		0.9	0.2326		50779	0.1261		108232	0.0117		183446
		10	0.2258		50663	0.1426		108217	0.0101		183644
	HFL (UB) - κ_0	0.5	0.3487	569.17	37263	0.4747	4391.65	139543	0.4633	7517.81	223324
		0.9	0.3520	569.10	37263	0.4820	4383.76	139545	0.4755	7515.77	223287
		10	0.3573	569.03	37263	0.4834	4385.32	139528	0.4815	7517	223257

TABLE VI: Test Accuracy with Trained $\mathbf{w}^T$ on CIFAR-100 dataset with $\kappa_0 = 10, \kappa_1 = \kappa_2 = \kappa_3 = 2$ and $T = 100$

Energy Budget	Methods	Dir($\bar{\alpha}$)	With CNN Model			With ResNet-18 Model			With ResNet-34 Model		
			Acc	Req T [s]	Req E [J]	Acc	Req T [s]	Req E [J]	Acc	Req T [s]	Req E [J]
429612	PHFL (Ours)	0.5	0.3722	2000	144127	0.4691	3200	134359	0.4797	4800	182362
		0.9	0.3669		144127	0.4721		134358	0.4853		182363
		10	0.3702		144128	0.4780		134356	0.4750		182361
	HFL-VC (UB) (Ours)	0.5	0.3796	2081.32	143837	0.4717	9710.81	348364	0.4816	15947.54	515916
		0.9	0.3669	2081.38	143837	0.4682	9705.78	348379	0.4893	15959.39	515991
		10	0.3831	2081.24	143837	0.4722	9710.21	348364	0.4825	15955.30	515940
	HFL (UB) - κ_0	0.5	0.3928	1040.75	71919	0.4676	4860.13	174177	0.4750	7983.24	257939
		0.9	0.3892	1040.68	71918	0.4738	4858.64	174186	0.4873	7987.34	258013
		10	0.3917	1040.59	71918	0.4770	4858.41	174172	0.4796	7984.84	257969

TABLE VII: Test Accuracy with Trained $\mathbf{w}^T$ on CIFAR-100 dataset with $\kappa_0 = 5, \kappa_1 = 4, \kappa_2 = \kappa_3 = 2$ and $T = 100$

Energy Budget	Methods	Dir($\bar{\alpha}$)	With CNN Model			With ResNet-18 Model			With ResNet-34 Model		
			Acc	Req T [s]	Req E [J]	Acc	Req T [s]	Req E [J]	Acc	Req T [s]	Req E [J]
859224	PHFL (Ours)	0.5	0.3857	2240	155919	0.4677	6400	245556	0.4935	9600	342061
		0.9	0.3774		155920	0.4861		245556	0.4898		342063
		10	0.3878		155920	0.4771		245555	0.4954		342063
	HFL-VC (UB) (Ours)	0.5	0.3948	2275.84	149053	0.4714	17534.56	558164	0.4994	30043.73	893273
		0.9	0.39	2275.75	149052	0.4808	17531.96	558155	0.5007	30058.09	893273
		10	0.3932	2275.71	149052	0.4806	17539.31	558167	0.4979	30040.23	893296
	R-PHFL	0.5	0.2363	2240	101571	0.1817	6400	216838	0.0168	9600	367556
		0.9	0.2206		101771	0.1972		217151	0.0140		367535
		10	0.2433		101791	0.2185		216750	0.0203		366833

and $\kappa_0 = 10$ on CIFAR-100 datasets for the CNN model. The performance comparisons for different $\bar{\alpha}$'s are shown in Table V - Table VII. From the tables, it is quite clear that pruning helps with negligible performance deviation from its original non-pruned counterparts. Besides, for the shallow CNN model, the performance gain, in terms of test accuracy, of our proposed PHFL is insignificant compared to the bulky ResNets. Moreover, increasing κ_0 or κ_1 generally improves the test accuracy. However, if the same t^{th} is to be used, it is beneficial to increase κ_0 compared to increasing κ_1.

VI. Conclusion

This work proposed a model pruning solution to alleviate bandwidth scarcity and limited computational capacity of wireless clients in heterogeneous networks. Using the convergence upper-bound, pruning ratio, computation frequency and transmission power of the clients were jointly optimized to maximize the convergence rate. The performances were evaluated on two popular datasets using three popular machine learning models of different total training parameter sizes. The results suggest that pruning can significantly reduce training time, energy expense and bandwidth requirement while incurring negligible test performance.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2017.

[2] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. IEEE INFOCOM*, 2019.

[3] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1935–1949, 2020.

[4] R. Jin, X. He, and H. Dai, "Communication efficient federated learning with energy awareness over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 7, pp. 5204–5219, Jan. 2022.

[5] Z. Chen, W. Yi, and A. Nallanathan, "Exploring representativity in device scheduling for wireless federated learning," *IEEE Trans. Wireless Commun.*, pp. 1–1, 2023.
[6] T. Zhang, K.-Y. Lam, J. Zhao, and J. Feng, "Joint device scheduling and bandwidth allocation for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, pp. 1–1, 2023.
[7] J. Wang, S. Wang, R.-R. Chen, and M. Ji, "Demystifying why local aggregation helps: Convergence analysis of hierarchical sgd," in *Proc. AAAI*, 2022.
[8] S. Hosseinalipour, S. S. Azam, C. G. Brinton, N. Michelusi, V. Aggarwal, D. J. Love, and H. Dai, "Multi-stage hybrid federated learning over large-scale d2d-enabled fog networks," *IEEE/ACM Trans. Network.*, vol. 30, no. 4, pp. 1569–1584, 2022.
[9] B. Xu, W. Xia, W. Wen, P. Liu, H. Zhao, and H. Zhu, "Adaptive hierarchical federated learning over wireless networks," *IEEE Trans. Vehicular Technol.*, vol. 71, no. 2, pp. 2070–2083, 2021.
[10] S. Liu, G. Yu, X. Chen, and M. Bennis, "Joint user association and resource allocation for wireless hierarchical federated learning with iid and non-iid data," *IEEE Trans. Wireless Commun.*, vol. 21, no. 10, pp. 7852–7866, 2022.
[11] 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, 2020.
[12] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE ICC*, 2020.
[13] C. Feng, H. H. Yang, D. Hu, Z. Zhao, T. Q. S. Quek, and G. Min, "Mobility-aware cluster federated learning in hierarchical wireless networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 10, pp. 8441–8458, Oct. 2022.
[14] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in *Proc. ICASSP*. IEEE, 2020.
[15] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.
[16] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proc. MLSys*, vol. 2, pp. 429–450, 2020.
[17] H. Yang, X. Zhang, P. Khanduri, and J. Liu, "Anarchic federated learning," in *Proc. ICML*. PMLR, 2022, pp. 25 331–25 363.
[18] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *Proc. NeurIPS*, vol. 33, pp. 7611–7623, 2020.
[19] T. Lin, S. U. Stich, L. Barba, D. Dmitriev, and M. Jaggi, "Dynamic model pruning with feedback," in *Proc. ICLR*, 2020.
[20] Y. Jiang, S. Wang, V. Valls, B. J. Ko, W.-H. Lee, K. K. Leung, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–13, Apr. 2022.
[21] Z. Zhu, Y. Shi, J. Luo, F. Wang, C. Peng, P. Fan, and K. B. Letaief, "Fedlp: Layer-wise pruning mechanism for communication-computation efficient federated learning," *arXiv preprint arXiv:2303.06360*, 2023.
[22] S. Liu, G. Yu, R. Yin, and J. Yuan, "Adaptive network pruning for wireless federated learning," *IEEE Wireless Commun. Lett.*, vol. 10, no. 7, pp. 1572–1576, 2021.
[23] S. Liu, G. Yu, R. Yin, J. Yuan, L. Shen, and C. Liu, "Joint model pruning and device selection for communication-efficient federated edge learning," *IEEE Trans. Commun.*, vol. 70, no. 1, pp. 231–244, Jan. 2022.
[24] J. Ren, W. Ni, and H. Tian, "Toward communication-learning trade-off for federated learning at the network edge," *IEEE Commun. Lett.*, vol. 26, no. 8, pp. 1858–1862, Aug. 2022.
[25] "*3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Proximity-services in 5G System protocol aspects; Stage 3*," 3GPP TS 24.554 V18.0.0, Rel. 18, Mar. 2023.
[26] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proc. ICLR*, 2019.
[27] M. F. Pervej, R. Jin, and H. Dai, "Resource constrained vehicular edge federated learning with highly mobile connected vehicles," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 6, pp. 1825–1844, June 2023.
[28] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified sgd with memory," in *Proc. NeurIPS*, 2018.
[29] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.
[30] S. Diamond and S. Boyd, "Cvxpy: A python-embedded modeling language for convex optimization," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2909–2913, 2016.
[31] E. Che, H. D. Tuan, and H. H. Nguyen, "Joint optimization of cooperative beamforming and relay assignment in multi-user wireless relay networks," *IEEE Trans. Wireless Comm.*, vol. 13, no. 10, pp. 5481–5495, 2014.
[32] Y. Sun, D. Xu, D. W. K. Ng, L. Dai, and R. Schober, "Optimal 3d-trajectory design and resource allocation for solar-powered uav communication systems," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4281–4298, 2019.
[33] A. Krizhevsky, "Learning multiple layers of features from tiny images," Apr. 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf
[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, 2016.

Md Ferdous Pervej (M'23) received the B.Sc. degree in electronics and telecommunication engineering from the Rajshahi University of Engineering and Technology, Rajshahi, Bangladesh, in 2014, and the M.S. and Ph.D. degrees in electrical engineering from Utah State University, Logan, UT, USA, in 2019 and North Carolina State University, Raleigh, NC, USA, in 2023, respectively.

He was with Mitsubishi Electric Research Laboratories, Cambridge, MA, in summer 2021, and with Futurewei Wireless Research and Standards, Schaumburg, IL from May to December 2022. He is currently a Postdoctoral Scholar – Research Associate in the Ming Hsieh Department of Electrical and Computer Engineering at the University of Southern California, Los Angeles, CA, USA. His primary research interests are wireless networks, distributed machine learning, vehicle-to-everything communication, edge caching/computing, and machine learning for wireless networks.

Richeng Jin (M'21) received the B.S. degree in information and communication engineering from Zhejiang University, Hangzhou, China, in 2015, and the Ph.D. degree in electrical engineering from North Carolina State University, Raleigh, NC, USA, in 2020.

He was a Postdoctoral Researcher in electrical and computer engineering at North Carolina State University, Raleigh, NC, USA, from 2021 to 2022. He is currently a faculty member of the department of information and communication engineering with Zhejiang University, Hangzhou, China. His research interests are in the general area of wireless AI, game theory, and security and privacy in machine learning/artificial intelligence and wireless networks.

Huaiyu Dai (F'17) received the B.E. and M.S. degrees in electrical engineering from Tsinghua University, Beijing, China, in 1996 and 1998, respectively, and the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ in 2002.

He was with Bell Labs, Lucent Technologies, Holmdel, NJ, in summer 2000, and with AT&T Labs-Research, Middletown, NJ, in summer 2001. He is currently a Professor of Electrical and Computer Engineering with NC State University, Raleigh, holding the title of University Faculty Scholar. His research interests are in the general areas of communications, signal processing, networking, and computing. His current research focuses on machine learning and artificial intelligence for communications and networking, multilayer and interdependent networks, dynamic spectrum access and sharing, as well as security and privacy issues in the above systems.

He has served as an area editor for IEEE Transactions on Communications, a member of the Executive Editorial Committee for IEEE Transactions on Wireless Communications, and an editor for IEEE Transactions on Signal Processing. Currently he serves as an area editor for IEEE Transactions on Wireless Communications. He was a co-recipient of best paper awards at 2010 IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2010), 2016 IEEE INFOCOM BIGSECURITY Workshop, and 2017 IEEE International Conference on Communications (ICC 2017). He received Qualcomm Faculty Award in 2019.