Parallel Successive Learning for Dynamic Distributed Model Training Over Heterogeneous Wireless Networks

Seyyedali Hosseinalipour, *Member, IEEE*, Su Wang, *Student Member, IEEE*, Nicolò Michelusi, *Senior Member, IEEE*, Vaneet Aggarwal, *Senior Member, IEEE*, Christopher G. Brinton, *Senior Member, IEEE*, David J. Love, *Fellow, IEEE*, and Mung Chiang, *Fellow, IEEE*

***Abstract*— Federated learning (FedL) has emerged as a popular technique for distributing model training over a set of wireless devices, via iterative local updates (at devices) and global aggregations (at the server). In this paper, we develop *parallel successive learning* (PSL), which expands the FedL architecture along three dimensions: (i) *Network*, allowing decentralized cooperation among the devices via device-to-device (D2D) communications. (ii) *Heterogeneity*, interpreted at three levels: (ii-a) Learning: PSL considers heterogeneous number of stochastic gradient descent iterations with different mini-batch sizes at the devices; (ii-b) Data: PSL presumes a *dynamic environment* with data arrival and departure, where the distributions of local datasets evolve over time, captured via a new metric for *model/concept drift*. (ii-c) Device: PSL considers devices with different computation and communication capabilities. (iii) *Proximity*, where devices have different distances to each other and the access point. PSL considers the realistic scenario where global aggregations are conducted with *idle times* in-between them for resource efficiency improvements, and incorporates *data dispersion* and *model dispersion with local model condensation* into FedL. Our analysis sheds light on the notion of *cold* vs. *warmed up* models, and model *inertia* in distributed machine learning. We then propose *network-aware dynamic model tracking* to optimize the model learning vs. resource efficiency tradeoff, which we show is an NP-hard signomial programming problem. We finally solve this problem through proposing a general optimization solver. Our numerical results reveal new findings on the interdependencies between the idle times in-between the global aggregations, model/concept drift, and D2D cooperation configuration.**

***Index Terms*— Cooperative federated learning, device-to-device communications, network optimization, dynamic machine learning.**

Manuscript received 9 February 2022; revised 27 September 2022 and 13 March 2023; accepted 25 May 2023; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Ioannidis. Date of publication 10 July 2023; date of current version 16 February 2024. This work was supported in part by Cisco Inc., in part by NSF under Grant CNS-2146171 and CNS-2129615, in part by ONR under Grant N000142212305, and in part by DARPA under Grant D22AP00168-00. *(Corresponding author: Seyyedali Hosseinalipour.)*

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

Su Wang, Christopher G. Brinton, David J. Love, and Mung Chiang are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906 USA (e-mail: wang2506@purdue.edu; cgb@purdue.edu; djlove@purdue.edu; chiang@purdue.edu).

Nicolò Michelusi is with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: nicolo.michelusi@asu.edu).

Vaneet Aggarwal is with the Schools of Industrial Engineering and Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906 USA (e-mail: vaneet@purdue.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNET.2023.3286987, provided by the authors.

Digital Object Identifier 10.1109/TNET.2023.3286987

I. Introduction

DISTRIBUTED machine learning (ML) over wireless networks has attracted significant attention recently, for applications ranging from keyboard next word prediction to autonomous driving [1], [2]. Distributed ML is an alternative to centralized ML, which requires a central node, e.g., a server, coexisting with the dataset. This alternative is of particular interest since in many applications the dataset is collected in a distributed fashion across a set of wireless devices, e.g., through their sensing equipment or the users' input, where the transfer of data to the cloud incurs significant energy consumption and long delays.

Federated learning (FedL) is the most recognized distributed ML technique, with the premise of keeping the devices' datasets local [3], [4]. Its conventional architecture resembles a *star* topology of device-server interactions (Fig. 1). Each model training round of FedL consists of (i) *local updating*, where devices update their local models based on their datasets and the global model, e.g., via stochastic gradient descent (SGD), and (ii) *global aggregation*, where the server aggregates the local models to a new global model and broadcasts it.

A. Related Work

Researchers have considered the effects of limited/imperfect communications in wireless networks (e.g., channel fading, packet loss) on the performance of FedL [5], [6], [7], [8], [9]. Also, quantization [10] and sparsification [11] techniques have been studied to facilitate FedL implementation.

Researchers have also considered the computation aspects of FedL over wireless networks [6], [7], [12], [13], [14]. Part of this literature has focused on the impact of straggler nodes, i.e., when some nodes have low computation capability, on model training [6], [12], [14], [15]. Another emphasis has been reducing the computation requirements of model training, e.g., through coding techniques [13], intelligent data offloading between devices [14], and device sampling [16].

Other research has focused on extending the *star* topology of FedL. Hierarchical FedL has proposed a *tree* structure between edge devices and the main server, e.g., [17]. This literature has mostly focused on specific use cases of two-tiered network structures above wireless cellular devices, e.g., with edge clouds at base stations. Additionally, there is a literature on *fully decentralized* FedL over *mesh* network architectures without a centralized server, where device-server communications are replaced with device-to-device (D2D) communications [18], [19]. Building upon this, *semi-decentralized architectures* for FedL have also been proposed, where D2D communications are exploited in

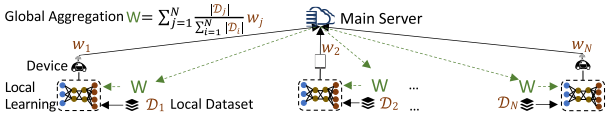

Fig. 1. "Star" learning topology architecture of conventional FedL.

conjunction with device-server interactions to improve the model training performance [20], [21]. In this literature, D2D communications are solely used for distributed model parameter aggregation across the nodes.

B. FedL Shortcomings and Solution Overview

1) Limitations of FedL Over Heterogeneous Wireless Networks: Consider a wireless network consisting of a set of battery limited devices, e.g., smart phones or sensors. Assume that a base station (BS) aims to train an ML model using the data gathered by the devices. There are multiple challenges faced in utilizing conventional FedL in this environment:

1) There might exist some devices with high data quality/quantity that suffer from low computation capability, resulting in their data being neglected during training.
2) There might exist some devices with high computation capability suffering from low data quantity/quality, resulting in idle processing resources.
3) There might exist some devices with high computation capability and good data quality/quantity, but poor channel conditions to the BS, making their participation in the model aggregation step challenging.
4) There might exist some devices with low computation capability and low data quality/quantity that have good channel conditions to the BS, which will be unused.

Hence, there are several conditions under which conventional FedL will result in poor performance. Roughly speaking, FedL models are biased towards devices with good channel conditions that have large amounts of data, which might encompass only a small portion of the overall network.

2) Enabling Data Sharing in FedL: Most of the FedL literature to date has assumed that users have strict data privacy concerns and never share their data. While this is true in some applications, e.g., healthcare systems, there are also applications where the data privacy is not strictly regulated, e.g., model training over sensor networks to detect abrupt environmental changes. Also, economic incentives (e.g., rewards, gas credit, or cash back) can be developed to encourage data sharing in many applications such as autonomous driving. Further, research on privacy preserving representation learning, which aims to obfuscate sensitive attributes of raw data, can expand the types of data which can be shared [22], [23]. These factors have motivated recent initial investigations of data offloading in FedL. Specifically, the recent work [24] proposes data offloading for edge-assisted FedL in vehicular networks. Also, researchers in [25] use data offloading to alleviate the impact of non-i.i.d. data in FedL. Finally, works [26], [27] introduce data offloading for federated and distributed learning in heterogeneous networks.

3) Parallel Successive Learning (PSL): Motivated by the above challenges, we propose a novel methodology for distributing ML over wireless networks that leverages the following properties, which constitute the pillars of *parallel successive learning* (PSL). PSL is a foremost realization of *fog learning* paradigm introduced by us in [26], which enables both parameter and data offloading among the devices.

1) Modern wireless devices are capable of device-to-device (D2D) communications [26], [28], [29], which are often considerably low power consuming. D2D communications can also be carried out in the *out-band* mode that does not occupy the licensed spectrum. There is an existing set of literature on D2D communications for various ad-hoc networks [30], [31], [32]. Motivated by this, we exploit D2D communications among the edge devices in PSL.
2) Devices with high data quantity/quality and low computation capability can transfer a portion of their data to those with better computation resources via D2D communications. Then, the devices with more computation resources will train on larger datasets, while the rest train on smaller datasets. To accomplish this, in PSL, we introduce a *data dispersion* mechanism among the devices.
3) Devices with high computation capabilities and bad channel conditions to the BS can execute model training and offload their trained models/gradients to those with better channels. Thus, we introduce a *model/gradient dispersion* mechanism in PSL to transfer the models/gradients among the devices through D2D communications. Further, given the heterogeneity among the devices and time-varying nature of datasets, in PSL, we consider local model training via *non-uniform SGD with stratified data sampling* with various mini-batch sizes and number of iterations across the devices.
4) Devices with good channel conditions to the BS and low computation capability can act as *aggregators*, receiving models from neighboring devices and conducting a *local aggregation* followed by uplink transmission to the BS. Subsequently, we introduce a *local condensation* method in PSL for conducting these device-side model aggregations.

C. Summary of Contributions

Our main contributions can be summarized as follows:

- We develop PSL, a distributed ML technique that introduces a new degree of freedom into model training, which is *idle times* in between global aggregations. PSL further extends FedL in the following three dimensions:
  - I. Network: PSL considers the local D2D network among the devices and allows direct device cooperation for data and parameter dispersion. This migrates away from the star topology of FedL and paves the road to more decentralized distributed ML architectures. This approach is complementary to recent works that utilize D2D in distributed ML to conduct model consensus [18], [19], [20], [21].
  - II. Heterogeneity: PSL considers and addresses three types of heterogeneity in wireless distributed ML:
    - Device: PSL assumes different computation and communication capabilities across devices. This is reflected in different CPU cycles ranges, chipset coefficients, and transmit powers in D2D vs. uplink transmissions.
    - Learning: PSL adapts device participation in model training according to their capabilities. In particular, it considers heterogeneous number of local SGD iterations with various mini-batch sizes at the devices.
    - Data: PSL considers a *dynamic environment* with data arrival and departure at the devices, where the distribution of the local datasets are non-i.i.d and evolving over time. We interpret this as the *drift* of the local loss over time.

III. Proximity: PSL considers D2D and device-to-server proximities to determine efficient transfers of both ML models and data across the network.

- We introduce *resource pooling*, where device resources are cooperatively orchestrated to facilitate ML model training. This is realized through a sequence of steps we develop and analyze: *data dispersion*, *local computation*, and *model/gradient dispersion with local condensation*.
- We analytically characterize the convergence behavior of PSL, through which we quantify the joint effect of (i) SGD with non-uniform local data sampling, (ii) non-uniform numbers of local SGD iterations across the devices, and (iii) dynamic data at the devices captured via *model/concept drift*. We leverage this to formulate the *network-aware PSL* problem, which optimizes over the tradeoffs between energy consumption, delay, and model performance for heterogeneous wireless networks.
- We show that network-aware the PSL problem is highly non-convex and NP-hard. We then propose a tractable approach based on posynomial approximation and constraint correction to solve the problem through a sequence of convex problems, which enjoys convergence guarantees. The proposed optimization transformation technique, given the generality of the analysis and the problem formulation, sheds light on the solution of a broader range of problems under the umbrella of *network-aware distributed learning*.

II. System Model

PSL in a nutshell. PSL is a general distributed ML paradigm, with its cornerstone built on FedL. It conducts distributed model training via *resource pooling* and coordinates device resources to operate in a cooperative manner. It conducts each round of model training via five steps: (i) global model broadcasting, (ii) data dispersion, (iii) local computation, (iv) model/gradient dispersion with local condensation, and (v) global model aggregation, which are illustrated in Fig. 2.

Henceforth, we present the PSL model tracking in Sec. II-A, describe the data management of PSL in Sec. II-B, and discuss the local model training of PSL in Sec. II-C. We then introduce model training phases experienced through PSL in Sec. II-D, and model the device orchestration in PSL in Sec. II-E.

A. Dynamic/Online Model Tracking Problem in PSL

We consider a network of N devices $\mathcal{N} = \{1, 2, \cdots, N\}$ coexisting with a server located at a BS, where ML model training is conducted through a series of global aggregations indexed by $k \in \mathbb{N}$. In contrast to most existing works in FedL that assume a stationary data distribution, PSL considers *dynamic ML* characterized by *data variations*. In particular, in PSL, the size and distribution of devices' datasets are assumed to be time-varying, i.e., changing across global aggregations. This is more realistic for real-world applications of distributed ML. For instance, in online product recommendation systems, user preferences may change from day to night and from season to season [33], and in keyword next word prediction, word choices are affected by trending news [34].

At global iteration k, each device $n \in \mathcal{N}$ is associated with a dataset $\mathcal{D}_n^{(k)}$, which has $D_n^{(k)} \triangleq |\mathcal{D}_n^{(k)}|$ data points. Each data point $d \in \mathcal{D}_n^{(k)}$ contains a feature vector, denoted by $\boldsymbol{d}$, and a label. For example, in image classification, the feature may be the RGB colors of all pixels in the image, and the label may be the location where the image was taken.

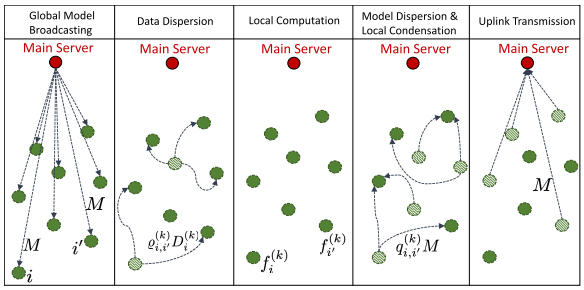

Fig. 2. A schematic of the learning architecture of PSL with five steps: (i) global model broadcasting, where the devices receive the global parameter from the server, (ii) data dispersion, where the devices conduct partial dataset offloading, (iii) local computation, where the devices compute their local models, (iv) model/gradient dispersion with local condensation, where devices conduct partial model/gradient transfer and perform local aggregations, (v) uplink transmission, where some devices transmit their models to the BS.

In PSL, the model training is started with an initial global model broadcast among the nodes, i.e., $\mathbf{w}^{(0)} \in \mathbb{R}^M$, where M is the *model dimension*. For each data point $d \in \mathcal{D}_n^{(k)}$, the ML model is associated with a *loss function* $f(\mathbf{w}, d)$ that quantifies the error of parameter $\mathbf{w} \in \mathbb{R}^M$. We refer to Table I in [7] for a list of ML loss functions. For an arbitrary $\mathbf{w}$, during the k-th global aggregation, let $F_n^{(k)}(\mathbf{w}) \triangleq F_n(\mathbf{w}|\mathcal{D}_n^{(k)})$ denote the local loss at node n, $F_n(\mathbf{w}|\mathcal{D}_n^{(k)}) = \sum_{d \in \mathcal{D}_n^{(k)}} f(\mathbf{w}, d)/D_n^{(k)}$.[1] Then, the global loss of the ML model is given by

$$F^{(k)}(\mathbf{w}) \triangleq F(\mathbf{w}|\mathcal{D}^{(k)}) = \frac{1}{D^{(k)}} \sum_{n \in \mathcal{N}} D_n^{(k)} F_n^{(k)}(\mathbf{w}), \quad (1)$$

where $D^{(k)} \triangleq |\mathcal{D}^{(k)}|$ is the cardinality of $\mathcal{D}^{(k)}$, the collection of data points across devices. We model the evolution of data through a new definition of *model drift* in Sec. III (Definition 2).

Due to the temporal variations in the distributions of local datasets, the optimal global model is time varying. In particular, for a training duration of K global iterations, the optimal global models can be represented as a sequence $\{\mathbf{w}^{(k)^\star}\}_{k=1}^K$, where

$$\mathbf{w}^{(k)^\star} = \arg\min_{\mathbf{w} \in \mathbb{R}^M} F^{(k)}(\mathbf{w}), \ \forall k, \quad (2)$$

which may not be unique in the case of loss functions which are not strongly convex, e.g., neural networks.

B. Data Heterogeneity and Management in PSL

We propose *partitioning* the dataset of each device into a *disjoint* set of sub-datasets, called stratum (see Fig. 3).[2] At global iteration k, we assume that dataset of device n consists of set $\mathcal{S}_n^{(k)} \triangleq \{\mathcal{S}_{n,1}^{(k)}, \mathcal{S}_{n,2}^{(k)}, \cdots\}$ of $S_n^{(k)} \triangleq |\mathcal{S}_n^{(k)}|$ strata, each with size $S_{n,j}^{(k)} \triangleq |\mathcal{S}_{n,j}^{(k)}|$. We also let $\widetilde{\sigma}_{n,j}^{(k)} = \sqrt{\frac{1}{S_{n,j}^{(k)}-1} \sum_{d \in \mathcal{S}_{n,j}^{(k)}} \|\boldsymbol{d} - \widetilde{\mu}_{n,j}^{(k)}\|_2^2}$ and $\widetilde{\mu}_{n,j}^{(k)} = \sum_{d \in \mathcal{S}_{n,j}^{(k)}} \boldsymbol{d}/S_{n,j}^{(k)}$ denote the sample (total) standard deviation and the mean of data inside stratum $\mathcal{S}_{n,j}^{(k)}$. We will exploit the means of

[1] Note that since the same ML model (e.g., neural network) architecture is trained across the node, f is not indexed by n.

[2] We use "stratum" as singular form and "strata" as plural form.

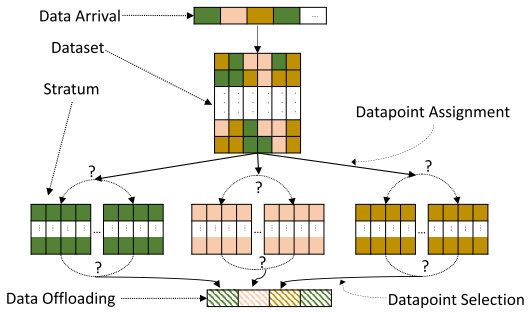

Fig. 3. A schematic of the data stratification for PSL. The figure represents the dataset at a device. The device faces two dilemmas: (i) how to assign the arriving data to the strata, and (ii) which data points to choose to offload.

the strata for data management as explained in the following, and the variance for optimal local data sampling for SGD in Sec. III (Proposition 1). We assume that data points in each stratum possess the same label.

This data partitioning has three advantages: (i) it provides effective data management upon data arrival/departure, (ii) it leads to tractable techniques to track the local dataset evolution, (iii) it opens the door to effective non-uniform data sampling to reduce the noise of SGD. We describe the first two advantages below and defer the explanation of the third one to Sec. II-C.

PSL considers two types of data movement: (i) inter-node data arrival/departure and (ii) intra-node data arrival/departure. In case (i), the arrival/departure of data at the devices is due to local data offloading, facilitated by device-to-device (D2D) communications. In case (ii), arrival/departure is caused by devices collecting and abandoning data. When new data arrives, the device has control on how to assign it to the existing strata. Also, in case (i), the devices have control on which data points from which strata to transfer (see Fig. 3).

Data arrival. If data points with new labels arrive, the device will form new strata and assign the respective data points to them. Otherwise, the device assigns each arriving data point to the stratum with the closest mean among those with the same label. In particular, given the current means of the strata at device n, $\widetilde{\mu}_{n,j}^{(k)}$, $\forall j$, the arriving data point d, with feature vector $\boldsymbol{d}$, gets assigned to stratum $\mathcal{S}_{n,j^\star}^{(k)}$, where $\mathcal{S}_{n,j^\star}^{(k)} = \arg\min_{\mathcal{S}_{n,j}^{(k)} \in \mathcal{S}_n^{(k)}} \|\widetilde{\mu}_{n,j}^{(k)} - \boldsymbol{d}\|$. This promotes homogeneity among the data points within each stratum. After the size of a stratum reaches a predefined maximum size $s_{\max}$, the stratum is further partitioned into two strata with equal sizes.[3] In addition, if the size of the stratum falls bellow a threshold $s_{\min}$, the stratum is assumed to merged with the strata containing data with the same label if such strata exists.

Data offloading. When a device offloads data, we can imagine two potential strategies for data point selection: (i) choosing from strata with higher variances, which results in a smaller sampling error for SGD, and (ii) choosing from strata with more data points, which reduces local model bias. Since data across the devices is non-iid in PSL, strategy (i) can increase the divergence across local datasets, which can significantly reduce the performance of the global model. We thus rely on strategy (ii), where for offloading a data point, device n chooses the stratum $\mathcal{S}_{n,j\star}^{(k)}$ with largest size and offloads the data point d that has the closest distance to the mean of strata, i.e., $d = \arg\min_{d \in \mathcal{S}_{n,j\star}^{(k)}} \|\widetilde{\mu}_{n,j}^{(k)} - \boldsymbol{d}\|$. In this way, we minimize the impact an offloading device experiences on its local data distribution.

Remark 1: In general, the optimal data offloading strategy may be a hybrid of the two mentioned strategies, since the impact of data offloading on the divergence of the local models from the global model is difficult to quantify in environments with unknown local data distributions. In this work, we propose the first steps towards smart data management and leave further investigations to future work.

Tracking of local data statistics. To cope with the dynamics of local datasets, we exploit an *online data statistics tracking* technique. We assume that each device has computed or otherwise gained knowledge of the initial mean and variance of its local data strata. Upon arrival or departure of each data point (or groups of data points), the device updates the mean and variance of its strata in an online manner through the following lemma.

Lemma 1 (Online Tracking of Strata Mean and Variance): Let $\mathcal{S}$ denote a set of $|\mathcal{S}|$ (vector) data points with mean $\boldsymbol{\mu}_{\text{old}}$ and sample variance σ_{old}^2. Let $\mathcal{A}$ denote a set of new data points that are added to $\mathcal{S}$ with mean and variance of $\boldsymbol{\mu}_{\mathcal{A}}$ and $\sigma_{\mathcal{A}}^2$, respectively, and $\mathcal{D}$ denote a set of data points departing from $\mathcal{S}$ with mean and variance of $\boldsymbol{\mu}_{\mathcal{D}}$ and $\sigma_{\mathcal{D}}^2$, respectively. Then, the new mean and variance of $\mathcal{S}$ are given by (3) and (4), as shown at the bottom of the next page.

Proof: The proof is provided in Appendix A. ∎

The method described in Lemma 1 requires the computation of mean and variance over the entire local dataset of each device only once at the beginning of ML model training.

C. Local Data Sampling and Model Training Iterations

As compared to SGD with uniform sampling, commonly used in FedL literature, we exploit SGD with non-uniform sampling. Our technique, inspired by *stratified sampling* in statistics [36], is advantageous to uniform sampling techniques when the distribution of data in each stratum is homogeneous while between strata is heterogeneous. The initial allocation of data points across the strata in each device can be conducted as in centralized SGD [37]; we are focused on the benefits of this technique to distributed ML.

To solve (2) in a distributed manner, within each global training round, devices conduct local model training through successive mini-batch SGD updates. However, device heterogeneity leads to varying contributions to model training. PSL assumes that devices utilize SGD with (i) different numbers of local iterations, (ii) different mini-batch sizes, and (iii) non-uniform data sampling. Formally, at global iteration k, device n performs $e_n^{(k)}$ iterations of SGD over its local dataset. The evolution of local model parameters is then given by[4]

$$\mathbf{w}_n^{(k),e} = \mathbf{w}_n^{(k),e-1} - \frac{\eta_k}{D_n^{(k)}} \sum_{j=1}^{S_n^{(k)}} \sum_{d \in \mathcal{B}_{n,j}^{(k),e}} \frac{S_{n,j}^{(k)} \nabla f(\mathbf{w}_n^{(k),e-1}, d)}{B_{n,j}^{(k)}}, \quad (5)$$

where $e \in \{1, \cdots, e_n^{(k)}\}$ denotes the local iteration index, η_k denotes the step-size, and $\mathbf{w}_i^{(k),0} = \mathbf{w}^{(k)}$ is the previously received global parameter from the BS. The nested sum in (5) indicates the overall gradient is computed by calculating the

[3] $s_{\max}$ is assumed to be an even number without loss of generality.

[4] The *complexity* of ML model training using different choices of neural networks has been discussed in [38].

gradient over samples from strata. In particular, $\mathcal{B}_{n,j}^{(k),e}$ denotes the data batch, sampled at the e-th iteration from stratum $\mathcal{S}_{n,j}^{(k)}$.

We conduct data sampling uniformly at random within each stratum, with the number of samples collected *varying* by strata, leading to an overall *non-uniform* sampling procedure (see Proposition 1). We assume that the number of sampled data points from each stratum does not change during each training interval, i.e., $B_{n,j}^{(k)} = |\mathcal{B}_{n,j}^{(k),e}|$, $\forall e$. We also define $B_n^{(k)} = \sum_{j=1}^{S_n^{(k)}} B_{n,j}^{(k)}$ as the SGD mini-batch size. Both $B_n^{(k)}$ and $B_{n,j}^{(k)}$ are design variables that should be tuned according to device capability and dataset heterogeneity, discussed in Sec. III and IV.

After each device n performs its last iteration of local model training, i.e., $e_n^{(k)}$, it computes the accumulated gradient $\overline{\nabla F}_n^{(k)} = \left(\mathbf{w}^{(k)} - \mathbf{w}_n^{(k),e_n^{(k)}}\right)/\eta_k$, offloaded either to its neighbors or to the BS. The global update is carried out at the BS as

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \eta_k \overline{\nabla F}^{(k)}, \tag{6}$$

where $\overline{\nabla F}^{(k)}$ is the normalized accumulated gradient of the devices, factoring in the heterogeneous number of local SGD iterations [39], [40], given by

$$\overline{\nabla F}^{(k)} = \sum_{n' \in \mathcal{N}} \frac{D_{n'}^{(k)} e_{n'}^{(k)}}{D^{(k)}} \sum_{n \in \mathcal{N}} \frac{D_n^{(k)}}{D^{(k)} e_n^{(k)}} \overline{\nabla F}_n^{(k)}. \tag{7}$$

Global model $\mathbf{w}^{(k+1)}$ will be then used to synchronize the devices for the next round of local updates. The process of recovering $\overline{\nabla F}^{(k)}$ at the server through gradient dispersion and local condensation among the devices is discussed in Sec. II-E.3.

D. Dynamic Model Training With Idle Times: "Cold" vs. "Warmed Up" Model, and Model "Inertia"

Many distributed ML applications (e.g., keyboard next word prediction or online cloth recommendation system) call for model training to be executed across long time periods (e.g., multiple weeks or seasons). In such settings, it is unrealistic to assume that the global aggregations are conducted continuously back to back, where the devices are always engaged in local model training, since this will require prohibitively high resource consumption. Thus, as compared to current art, PSL further introduces a new design parameter $\Omega^{(k)} \in \mathbb{Z}^+ \cup \{0\}$ that captures the *idle time* between the end of global aggregation $k-1$ and the beginning of k, during which the devices are not engaged in model training ($\Omega^{(0)} \triangleq 0$). This parameter captures the frequency of engagement of the devices in model training.

Initially, when the global model is launched from a "cold start", i.e., it is not well-trained, conducting model training results in significant improvements in model performance, calling for rapid global training rounds, i.e., small $\Omega^{(k)}$-s. After several global training rounds, the model training at the devices starts with a "warmed up" model, which marginalizes the reward in terms of model performance gains. In this regime, if the data at the devices changes rapidly, to track the changes, fast global rounds are required; otherwise, model training can be delayed, i.e., large $\Omega^{(k)}$-s, to save energy and network resources. In particular, given a warmed up global model, the model training should be *triggered* when sufficient changes in the local data distributions is occurred. We call this phenomenon the *inertia* of the global model, since it resembles the same notion in physics. Initially, the model has a lower inertia. During model training, the inertia of the global model increases (i.e., it becomes reluctant to changes) and it takes large shifts in the data distribution to trigger a new model training round. A key contribution of our work is in characterizing this notion of inertia in distributed ML.

E. Cooperative Resource Pooling in PSL

In PSL, the resource of the devices are utilized cooperatively, where both data and model parameters can be transferred in D2D mode. Optimizing these transfers requires consideration of their resource requirements.

At global aggregation k, for each device $n \in \mathcal{N}$, we let $h_n^{(k)}$ denote the channel gain of the device to the BS. Consequently, the data rate of device n to the BS is given by[5]

$$r_n^{(k)} = B^{\mathsf{U}} \log\left(1 + |h_n^{(k)}|^2 p_n^{\mathsf{U}} / N^{\mathsf{U}}\right), \tag{8}$$

where B^{U} denotes the uplink bandwidth given to each device, p_n^{U} is the uplink transmit power of the device, and $N^{\mathsf{U}} = N_0 B^{\mathsf{U}}$ is the noise power with N_0 denoting the noise spectral density.[6]

Similarly, with D2D communications, for two devices $n, m \in \mathcal{N}$, we define the data rate at device m achieved via transmission from device n in D2D mode as follows:

$$r_{n,m}^{(k)} = B^{\mathsf{D}} \log\left(1 + |h_{n,m}^{(k)}|^2 p_n^{\mathsf{D}} / N^{\mathsf{D}}\right), \tag{9}$$

where B^{D} denotes the D2D bandwidth given to each user pair, $h_{n,m}^{(k)}$ denotes the channel gain among the respective nodes, p_n^{D} denotes the D2D transmit power of device n, and $N^{\mathsf{D}} = N_0 B^{\mathsf{D}}$.

[5] $\log(.)$ denotes logarithm with base 2 unless otherwise stated and the data rate expressions are measured at the instance of data/gradient transmissions.

[6] An alternative method is to use the average/expected data rate expressions obtained under average fading power. This would lead to closed form expressions for data rates (e.g., see [41]) that can be directly used in our subsequent optimization formulation. Note that upon using the instantaneous data rates as in (8), if the data transmission time exceeds the channel coherence time, the instantaneous data rate may change during the data transmission. Using the average/expected data rate will also resolve this issue.

$$\boldsymbol{\mu}_{\mathsf{new}} = \frac{|\mathcal{S}|\boldsymbol{\mu}_{\mathsf{old}} + |\mathcal{A}|\boldsymbol{\mu}_{\mathcal{A}} - |\mathcal{D}|\boldsymbol{\mu}_{\mathcal{D}}}{|\mathcal{S}| + |\mathcal{A}| - |\mathcal{D}|}, \tag{3}$$

$$\sigma^2_{\mathsf{new}} = \frac{(|\mathcal{S}|-1)\sigma^2_{old} + (|\mathcal{A}|-1)\sigma^2_{\mathcal{A}} - (|\mathcal{D}|-1)\sigma^2_{\mathcal{D}}}{|\mathcal{A}| + |\mathcal{S}| - |\mathcal{D}| - 1} + \frac{\left(\frac{|\mathcal{A}||\mathcal{S}|}{|\mathcal{A}|+|\mathcal{S}|-|\mathcal{D}|}\right)\|\boldsymbol{\mu}_{\mathsf{old}} - \boldsymbol{\mu}_{\mathcal{A}}\|^2 - \left(\frac{|\mathcal{S}||\mathcal{D}|}{|\mathcal{A}|+|\mathcal{S}|-|\mathcal{D}|}\right)\|\boldsymbol{\mu}_{\mathsf{old}} - \boldsymbol{\mu}_{\mathcal{D}}\|^2}{|\mathcal{A}| + |\mathcal{S}| - |\mathcal{D}| - 1} - \frac{\left(\frac{|\mathcal{A}||\mathcal{D}|}{|\mathcal{A}|+|\mathcal{S}|-|\mathcal{D}|}\right)\|\boldsymbol{\mu}_{A} - \boldsymbol{\mu}_{\mathcal{D}}\|^2}{|\mathcal{A}| + |\mathcal{S}| - |\mathcal{D}| - 1}. \tag{4}$$

Remark 2: Since the physical layer details are abstracted in this paper, for simplicity, it is assumed that the devices utilize a multi-access protocol such as FDMA [42] that avoids interference of the devices in uplink transmissions. The same holds for D2D communications. Investigating the effect of interference is left as a topic for future works.

We next formalize the data dispersion, local computations, and local parameter aggregations with local model condensation processes in PSL visualized in Fig. 2:

1) Data Dispersion: In PSL, each device can offload a portion of its data in D2D mode. We define a continuous variable $\varrho_{n,m}^{(k)} \in [0,1]$, where $\sum_{m\in\mathcal{N}} \varrho_{n,m}^{(k)} = 1,\ \forall n,k$, as the fraction of data of node n offloaded to node m at global iteration k, with $\rho_{n,n}^{(k)}$ being the amount of data kept locally.[7] We refer to $\boldsymbol{\varrho}^{(k)} = \left[\varrho_{n,m}^{(k)}\right]_{1\le n,m\le N}$ as the *data dispersion matrix*. The *data reception time* at device m from n is thus given by

$$T_{n,m}^{\mathsf{DR},(k)} = \varrho_{n,m}^{(k)} D_n^{(k)} b^{\mathsf{D}} / r_{n,m}^{(k)}, \tag{10}$$

where b^{D} denotes the number of bits representing one data point. Assuming data dispersion occurs in parallel (see Remark 2), the total data reception time at node m is[8]

$$T_m^{\mathsf{DR},(k)} = \max_{n\in\mathcal{N}\setminus\{m\}} \left\{T_{n,m}^{\mathsf{DR},(k)}\right\}. \tag{11}$$

2) Local Computation: We assume that the devices are equipped with different processing units. Let us define a_n as the number of CPU cycles that are used to process one data sample at device n. Subsequently, the *computation time* at device n to execute $e_n^{(k)}$ local SGD iterations based on (5) at global iteration k is given by

$$T_n^{\mathsf{C},(k)} = e_n^{(k)} a_n B_n^{(k)} / f_n^{(k)}, \tag{12}$$

where $f_n^{(k)}$ denotes the CPU frequency of the device, and the mini-batch size satisfies $B_n^{(k)} \le \sum_{m\in\mathcal{N}} \varrho_{m,n}^{(k)} D_m^{(k)}$.

3) Model Parameter/Gradient Dispersion and Local Condensation: PSL introduces a scenario in which after performing local computations, each device can partition its vector of local model/accumulated gradient into different *chunks* and disperse the chunks among the neighboring devices in D2D mode, which we call *model/gradient dispersion* (see Fig. 4). Considering the update rule in (6)&(7), at each device n, this can be carried out via either dispersing the (normalized) latest local model $\frac{D_n^{(k)}}{D^{(k)} e_n^{(k)}} \mathbf{w}^{(k),e_n^{(k)}}$, or via dispersing the (normalized accumulated) gradient $\frac{D_n^{(k)}}{D^{(k)} e_n^{(k)}} \overline{\nabla F}_n^{(k)}$, both of which are vectors with size M. In the following, we focus on the latter approach without loss of generality. In this work, we consider that in each aggregation period, a device either

[7]We assume that the offloaded data points of each device are no longer used by that device in its local model training. This assumption is made to alleviate the global model becoming biased towards the data distributions of devices offloading samples, i.e., to avoid double-counting them in the SGD process of the transmitting and the receiving devices.

[8]Interference-avoidance multiple access techniques other than FDMA would also be compatible with our methodology, so long as the expression in (11) is changed accordingly. For example, with TDMA, which is used in [12], $\max_{n\in\mathcal{N}\setminus\{m\}} \left\{T_{n,m}^{\mathsf{DR},(k)}\right\}$ in (11) should be replaced with $\sum_{n\in\mathcal{N}\setminus\{m\}} \left\{T_{n,m}^{\mathsf{DR},(k)}\right\}$, which our optimization methodology in Sec. IV can easily incorporate. The same holds for other parts of the paper, where simultaneous information exchange under FDMA is presumed.

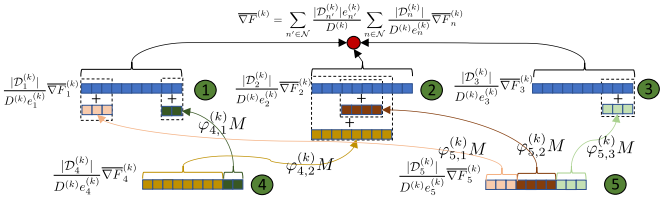

Fig. 4. A schematic of gradient dispersion with local condensation for a network of 5 nodes. Each node n computes $\overline{\nabla F}_n^{(k)}$ (with size M), and normalizes it with respect to its number of data points and SGD iterations. Nodes 4 and 5 partition the resulting vector to multiple chunks and disperse the chunks across their neighbors. The recipients sum the received vectors with their own vectors and transmit the results (vector of size M) to the main server, which computes $\overline{\nabla F}^{(k)}$ (see (7)) and conducts global aggregation (see (6)).

(i) completely disperses its local gradient across its neighbors in D2D mode or (ii) keeps its local gradient, receives gradient chucks from its neighbors, and engages in uplink transmission.

To alleviate uplink transmission of multiple gradients, when a device receives gradient chunks, it conducts a local aggregation, i.e., summing the received chunks with its local gradient, and only sends the resulting vector to the BS. We call this *local model/gradient condensation* since it results in uplink transmission of a vector with size M at each node regardless of the number of received chunks from its neighbors (see Fig. 4).

To model this process, we define a continuous variable $\varphi_{n,m}^{(k)} \in [0,1]$ to denote the fraction of gradient parameters (i.e., the fraction of indices of the gradient vector) offloaded from device n to m during global iteration k, where $\sum_{m\in\mathcal{N}} \varphi_{n,m}^{(k)} = 1,\ \forall n,k$. If node n does not share its gradient with any neighbor, $\varphi_{n,n}^{(k)} = 1$. We define $\boldsymbol{\varphi}^{(k)} = [\varphi_{n,m}^{(k)}]_{1\le n,m\le N}$ as the *gradient dispersion matrix*. The time required for the reception of gradient elements at node m from node n is given by

$$T_{n,m}^{\mathsf{RG},(k)} = \varphi_{n,m}^{(k)} M b^{\mathsf{G}} / r_{n,m}^{(k)}, \tag{13}$$

where b^{G} denotes the number of bits required to represent one element of the gradient vector with size M. Since the reception of gradient occurs in a parallel manner, the total time required for the reception of gradient parameters at node m is given by

$$T_m^{\mathsf{RG},(k)} = \max_{n\in\mathcal{N}\setminus\{m\}} \left\{T_{n,m}^{\mathsf{RG},(k)}\right\}. \tag{14}$$

Finally, the uplink transmission time at node n is given by

$$T_n^{\mathsf{GT},(k)} = M b^{\mathsf{G}} / r_n^{(k)}. \tag{15}$$

As explained earlier, each node either disperses its gradient or keeps it entirely local, and thus $\varphi_{n,n}^{(k)}$ is a binary variable, $\forall n$. Thus, (15) can be expressed as $T_n^{\mathsf{GT},(k)} = M b^{\mathsf{G}} \varphi_{n,n}^{(k)} / r_n^{(k)}$.

4) Energy Consumption: At global iteration k, the total energy consumption $E_n^{(k)}$ at each device n is given by

$$E_n^{(k)} = E_n^{\mathsf{DD},(k)} + E_n^{\mathsf{C},(k)} + E_n^{\mathsf{GD},(k)} + E_n^{\mathsf{GT},(k)}, \tag{16}$$

where $E_n^{\mathsf{DD},(k)}$, $E_n^{\mathsf{C},(k)}$, $E_n^{\mathsf{GD},(k)}$, $E_n^{\mathsf{GT},(k)}$ denote the energy used for data dispersion, local computation, gradient dispersion, and gradient transmission to the BS, given as

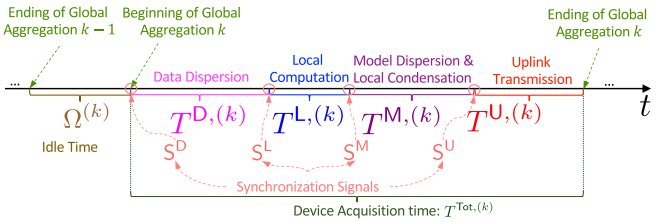

Fig. 5. A schematic of PSL operations during global round k. There is an *idle time* in between global aggregations. During the *device acquisition time*, a series of synchronization signals are broadcast by the BS.

follows:

$$E_n^{\mathsf{DD},(k)} = \sum_{m\in\mathcal{N}\setminus\{n\}} p_n^{\mathsf{D}} \varrho_{n,m}^{(k)} D_n^{(k)} b^{\mathsf{D}} / r_{n,m}^{(k)}, \tag{17}$$

$$E_n^{\mathsf{C},(k)} = (\alpha_n/2) a_n e_n^{(k)} B_n^{(k)} \left(f_n^{(k)}\right)^2, \tag{18}$$

$$E_n^{\mathsf{GD},(k)} = \sum_{m\in\mathcal{N}\setminus\{n\}} p_n^{\mathsf{D}} \varphi_{n,m}^{(k)} M b^{\mathsf{G}} / r_{n,m}^{(k)}, \tag{19}$$

$$E_n^{\mathsf{GT},(k)} = p_n^{\mathsf{U}} M b^{\mathsf{G}} \varphi_{n,n}^{(k)} / r_n^{(k)}, \tag{20}$$

In (18), $\alpha_n/2$ is the effective chipset capacitance of $n \in \mathcal{N}$ [6].

5) PSL Under Synchronization Signaling: To synchronize the processes involved in PSL, we assume that the BS ochestrates the devices through a sequence of signals. In particular, the BS starts each global training round via broadcasting signal S^{D}, which begins the data dispersion phase among the devices. Afterward, the BS dictates the start of local model training via broadcasting signal S^{L}. Then, it starts the model/gradient dispersion phase with a signal S^{M}. Finally, the global aggregation round ends with the BS broadcasting a signal S^{U}, at which time the devices start uplink transmissions. Let $T^{\mathsf{D},(k)}$, $T^{\mathsf{L},(k)}$, $T^{\mathsf{M},(k)}$, $T^{\mathsf{U},(k)}$ denote the corresponding time interval associated with data dispersion, local model training, model/gradient dispersion, and uplink transmissions, respectively. The total delay of global training round k is $T^{\mathsf{Tot},(k)} = T^{\mathsf{D},(k)} + T^{\mathsf{L},(k)} + T^{\mathsf{M},(k)} + T^{\mathsf{U},(k)}$, which we refer to as the *device acquisition time*. Fig. 5 depicts a schematic of the timeline of PSL during a global training round.

III. Convergence Analysis of PSL

We first make two standard assumptions, which are common in literature [6], [7], [14], [40], [43], to conduct convergence analysis. Henceforth, notation $\|.\|$ denotes the 2-norm.

Assumption 1 (Smoothness of the Global and Local Loss Functions): Local loss function $F_n^{(k)}$ is β-smooth, $\forall n \in \mathcal{N}, k$:

$$\|\nabla F_n^{(k)}(\mathbf{w}) - \nabla F_n^{(k)}(\mathbf{w}')\| \leq \beta \|\mathbf{w} - \mathbf{w}'\|, \forall \mathbf{w}, \mathbf{w}' \in \mathbb{R}^M, \tag{23}$$

which implies the β-smoothness of the global loss function F.

Assumption 2 (Bounded Dissimilarity of Local Loss Functions): There exist finite constants $\zeta_1 \geq 1$, $\zeta_2 \geq 0$, for any set of coefficients $\{a_n \geq 0\}_{n\in\mathcal{N}}$, where $\sum_{n\in\mathcal{N}} a_n = 1$, such that

$$\sum_{n\in\mathcal{N}} a_n \|\nabla F_n^{(k)}(\mathbf{w})\|^2 \leq \zeta_1 \left\| \sum_{n\in\mathcal{N}} a_n \nabla F_n^{(k)}(\mathbf{w}) \right\|^2 + \zeta_2, \forall k, \mathbf{w}. \tag{24}$$

The two parameters ζ_1 and ζ_2 introduced in Assumption 2 measure the level of heterogeneity (i.e., non-i.i.d-ness) across the devices' datasets. If the device's datasets are homogeneous (i.e., i.i.d. across the devices), we will have $\zeta_1 = 1, \zeta_2 = 0$, and these values will increase as the heterogeneity of data across the devices increases.

We next define *local data variability* to further measure the level of heterogeneity at the devices' local datasets:

Definition 1 (Local Data Variability): The local data variability at each device n is measured via $\Theta_n \geq 0$, which $\forall \mathbf{w}, k$ satisfies

$$\|\nabla f(\mathbf{w}, d) - \nabla f(\mathbf{w}, d')\| \leq \Theta_n \|d - d'\|, \forall d, d' \in \mathcal{D}_n^{(k)}. \tag{25}$$

We further define $\Theta = \max_{n\in\mathcal{N}}\{\Theta_n\}$.

We next quantify the dynamics of data variations at the devices via introducing a versatile measure that connects the variation in the data to the performance of the ML model:

Definition 2 (Model/Concept Drift): Let $D_n(t)$ and $D(t)$ denote the instantaneous number of data points at device n and total number of data points at wall clock time t (measured in seconds) during PSL. For each device n, we measure the online model/concept drift for two consecutive time instances $t-1$ and t during which the device does not conduct ML model training by $\Delta_n(t) \in \mathbb{R}$, which captures the variation of the local loss for any model parameter, $\forall \mathbf{w} \in \mathbb{R}^M$:

$$\frac{D_n(t)}{D(t)} F_n\left(\mathbf{w} \middle| \mathcal{D}_n(t)\right) - \frac{D_n(t-1)}{D(t-1)} F_n\left(\mathbf{w} \middle| \mathcal{D}_n(t-1)\right) \leq \Delta_n(t). \tag{26}$$

$\Delta_n^{(t)}$ is assumed to be measured only at discrete time instances $t \in \mathbb{Z}^+$ and is presumed to be fixed in the continues time interval $(t-1, t]$ (i.e., for the duration of 1 second).

A larger value for the model/concept drift, i.e., $\Delta_n \gg 0$, implies a larger local loss variation and a harder tracking of the optimal model parameters for the ML training. Also, the above definition encompasses the case where due to model drift the old model becomes more fit to the current data (when $\Delta_n < 0$). Our definition of model drift is different from other few definitions in current art [39], [44] from two aspects. First, our definition connects the data variations to the model performance and can be used in scenarios where major variations in some dimensions of data (i.e., some of the features) does not affect the performance. Second, our proposed drift is estimable (i.e., it is not defined with respect to the variations in the optimal model as in [39] and [44] which is by itself unknown a priori).

We next obtain the convergence behavior of PSL for non-convex loss functions. We use $\widehat{\mathcal{D}}_n^{(k)}$ to denote the set of data points at device n used during global round k obtained after the model dispersion phase, $\widehat{D}_n^{(k)} = |\widehat{\mathcal{D}}_n^{(k)}| = \sum_{m\in\mathcal{N}} \varrho_{m,n}^{(k)} D_m^{(k)}$. Also, with slight abuse of notations, we use $S_{n,j}^{(k)}$ in the bounds to denote the number of data points in respective stratum and $\widetilde{\sigma}_{n,j}^{(k)}$ to denote its variance during local model training of global round k which are obtained via Lemma 1.

Theorem 1 (General Convergence Behavior of PSL): Assume

$$\eta_k \leq \min\left\{ \frac{1}{2\beta} \sqrt{\frac{\Lambda^{(k)}}{\zeta_1(\beta^2 + \Lambda^{(k)}) \left(e_{\mathsf{max}}^{(k)} \left(e_{\mathsf{max}}^{(k)} - 1\right)\right)}}, \left(2\beta \sum_{n\in\mathcal{N}} \frac{D_n^{(k)} e_n^{(k)}}{D^{(k)}}\right)^{-1} \right\},$$

where $\Lambda^{(k)} < 1$ is a constant and $e_{\mathsf{max}}^{(k)} = \max_{n\in\mathcal{N}}\{e_n^{(k)}\}$. Also, define $\Gamma^{(k)} = \frac{\eta_k}{2} \sum_{n\in\mathcal{N}} \frac{D_n^{(k)} e_n^{(k)}}{D^{(k)}}$, and $\Delta^{(k)} =$

$\sum_{n\in\mathcal{N}}\widehat{\Delta}_n^{(k)}$, *where* $\widehat{\Delta}_n^{(k)}=\max_{t\in T^{\mathsf{Idle},(k)}}\Delta_n(t)$. *Then, the cumulative average of the gradient of the global loss function over the training period of PSL satisfies the upper bound in (21), as shown at the bottom of the page.*

Proof: Refer to Appendix B for the detailed proof. ■

In (21), $F^{(k-1)}(\mathbf{w}^{(k)})=F(\mathbf{w}^{(k)}|\mathcal{D}^{(k-1)})$ denotes the loss under which the k-th global model training ends where $\mathbf{w}^{(k)}$ is obtained and $F^{(k)}(\mathbf{w}^{(k+1)})=F(\mathbf{w}^{(k+1)}|\mathcal{D}^{(k)})$ denotes the loss under which the $k+1$-th global model training ends where $\mathbf{w}^{(k+1)}$ is obtained. Also, $F^{(-1)}(\mathbf{w}^{(0)})$ denotes the *initial* loss of the algorithm before the start of any model training, i.e., $\Omega^{(1)}$ seconds before the first model training round.

Main Takeaways. The bound in (21) captures the effect of the ML-related parameters on the performance of PSL. Term (a) captures the effect of consecutive loss function gains during ML training. Term (b) captures the effect of model drift (via $\Delta^{(k+1)}$). Terms (c) and (e) are both concerned with the data stratification and mini-batch sizes used at the devices. Term (d) captures the effect of model dissimilarity (via ζ_2) and the number of SGD iterations (via $e_{\mathsf{max}}^{(k)}$). Larger model dissimilarity leads to smaller step size choices (ζ_1 incorporated in the condition on η_k) and larger upper bound (ζ_2 in (d)). Also, larger local data variability implies a larger bound (Θ in (c) and (e)). Further, terms (a) and (b) are inversely proportional to η_k (incorporated in $\Gamma^{(k)}$), terms (c) and (d) are quadratically proportional to η_k, and term (e) is linearly proportional to it (incorporated in $\Gamma^{(k)}$). The bound also provides further insights: (i) upon having $e_n^{(k)}=1$, $\forall n,k$, terms (c) and (d) become zero and the bound demonstrate 1-epoch distributed ML with non-uniform SGD sampling; (ii) upon sampling all the data points, i.e., $B_{n,j}^{(k)}=S_{n,j}^{(k)}$, $\forall j,n,k$, terms (c) and (e) become zero and the bound reveals the convergence of full-batch local gradient descents, (iii) upon having uniform sampling across strata, $B_{n,j}^{(k)}=B_n^{(k)}S_{n,j}^{(k)}/D_n^{(k)}$, the bound demonstrates the convergence upon uniform data sampling using SGD with mini-batch size $B_n^{(k)}$ at each device n; (iv) the effect of offloading is reflected in $S_{n,j}^{(k)}$ and $\widehat{D}_n^{(k)}$. In particular, considering terms (c) and (e), increasing $S_{n,j}^{(k)}$-s (i.e., data reception at device n), with everything else being constant, needs to be met via increasing $B_{n,j}^{(k)}$ (i.e., increasing the mini-batch size) to keep the bound value fixed. Similarly, upon data offloading, the device can use a smaller mini-batch size. Thus, the bound promotes offloading data from devices with low computation capability to those with higher capability.

The bound in (21) reveals that reaching convergence is attainable under certain circumstances, which we aim to obtain:

Corollary 1 (Convergence Under Proper Choice of Step Size and Bounded Local Iterations): In addition to the conditions in Theorem 1, further assume that (i) $\eta_k=\alpha/\sqrt{Ke_{\mathsf{sum}}^{(k)}/N}$ *with a finite positive constant* α *chosen to satisfy the condition on* η_k *in Theorem 1, where* $e_{\mathsf{sum}}^{(k)}=$

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\nabla F^{(k)}(\mathbf{w}^{(k)})\right\|^2\right]\leq\frac{1}{K}\Bigg[\underbrace{\sum_{k=0}^{K-1}\frac{\mathbb{E}\left[F^{(k-1)}(\mathbf{w}^{(k)})-F^{(k)}(\mathbf{w}^{(k+1)})\right]}{\Gamma^{(k)}(1-\Lambda^{(k)})}}_{(a)}+\underbrace{\sum_{k=0}^{K-1}\frac{\Omega^{(k+1)}\Delta^{(k+1)}}{\Gamma^{(k)}(1-\Lambda^{(k)})}}_{(b)}$$
$$+\sum_{k=0}^{K-1}\frac{1}{(1-\Lambda^{(k)})}\underbrace{\left(8\beta^2\Theta^2\eta_k^2\sum_{n\in\mathcal{N}}\frac{\widehat{D}_n^{(k)}}{D^{(k)}}(e_n^{(k)}-1)\sum_{j=1}^{S_n^{(k)}}\left(1-\frac{B_{n,j}^{(k)}}{S_{n,j}^{(k)}}\right)\frac{S_{n,j}^{(k)}}{\left(D_n^{(k)}\right)^2}\frac{(S_{n,j}^{(k)}-1)\left(\widetilde{\sigma}_{n,j}^{(k)}\right)^2}{B_{n,j}^{(k)}}\right)}_{(c)}$$
$$+\underbrace{8\zeta_2\eta_k^2\beta^2\left(e_{\mathsf{max}}^{(k)}\right)\left(e_{\mathsf{max}}^{(k)}-1\right)}_{(d)}+\underbrace{8\Theta^2\beta\Gamma^{(k)}\sum_{n\in\mathcal{N}}\left(\frac{\widehat{D}_n^{(k)}}{D^{(k)}\sqrt{e_n^{(k)}}}\right)^2\sum_{j=1}^{S_n^{(k)}}\left(1-\frac{B_{n,j}^{(k)}}{S_{n,j}^{(k)}}\right)\frac{S_{n,j}^{(k)}}{\left(D_n^{(k)}\right)^2}\frac{(S_{n,j}^{(k)}-1)\left(\widetilde{\sigma}_{n,j}^{(k)}\right)^2}{B_{n,j}^{(k)}}}_{(e)}\Bigg] \tag{21}$$

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\nabla F^{(k)}(\mathbf{w}^{(k)})\right\|^2\right]$$
$$\leq 2\sqrt{\widehat{e}_{\mathsf{max}}}\frac{F^{(-1)}(\mathbf{w}^{(0)})-F^{(K)\star}}{\overline{e}_{\mathsf{min}}\alpha\sqrt{NK}(1-\Lambda_{\mathsf{max}})}+\frac{2\sqrt{\widehat{e}_{\mathsf{max}}}}{\overline{e}_{\mathsf{min}}\alpha\sqrt{NK}}\sum_{k=0}^{K-1}\frac{\Omega^{(k+1)}\Delta^{(k+1)}}{1-\Lambda_{\mathsf{max}}}+\frac{1}{K}\sum_{k=0}^{K-1}\frac{1}{(1-\Lambda_{\mathsf{max}})}\Bigg(\frac{8\beta^2\Theta^2\alpha^2N}{Ke_{\mathsf{sum}}^{(k)}}$$
$$\times\sum_{n\in\mathcal{N}}\frac{\widehat{D}_n^{(k)}}{D^{(k)}}(e_n^{(k)}-1)\sum_{j=1}^{S_n^{(k)}}\left(1-\frac{B_{n,j}^{(k)}}{S_{n,j}^{(k)}}\right)\frac{S_{n,j}^{(k)}}{\left(\widehat{D}_n^{(k)}\right)^2}\frac{(S_{n,j}^{(k)}-1)\left(\widetilde{\sigma}_{n,j}^{(k)}\right)^2}{B_{n,j}^{(k)}}+\frac{8\zeta_2\alpha^2\beta^2N}{Ke_{\mathsf{sum}}^{(k)}}\left(e_{\mathsf{max}}^{(k)}\right)\left(e_{\mathsf{max}}^{(k)}-1\right)\Bigg)$$
$$+\frac{1}{K}\sum_{k=0}^{K-1}\frac{4\overline{e}_{\mathsf{max}}\alpha\Theta^2\beta\sqrt{N}}{(1-\Lambda_{\mathsf{max}})\sqrt{e_{\mathsf{sum}}^{(k)}}\sqrt{K}}\sum_{n\in\mathcal{N}}\left(\frac{\widehat{D}_n^{(k)}}{D^{(k)}\sqrt{e_n^{(k)}}}\right)^2\sum_{j=1}^{S_n^{(k)}}\left(1-\frac{B_{n,j}^{(k)}}{S_{n,j}^{(k)}}\right)\frac{S_{n,j}^{(k)}}{\left(\widehat{D}_n^{(k)}\right)^2}\frac{(S_{n,j}^{(k)}-1)\left(\widetilde{\sigma}_{n,j}^{(k)}\right)^2}{B_{n,j}^{(k)}} \tag{22}$$

$\sum_{n\in\mathcal{N}} e_n^{(k)}$, *(ii)* $\widehat{e}_{\mathsf{min}} \leq e_{\mathsf{sum}}^{(k)} \leq \widehat{e}_{\mathsf{max}}$ *for two finite positive constants* $\widehat{e}_{\mathsf{min}}$ *and* $\widehat{e}_{\mathsf{max}}$, $\forall k$, *(iii)* $\max_k \left\{\Lambda^{(k)}\right\} \leq \Lambda_{\mathsf{max}} < 1$, *(iv)* $\overline{e}_{\mathsf{min}} \leq e_{\mathsf{avg}}^{(k)} \leq \overline{e}_{\mathsf{max}}$ *for two finite positive constants* $\overline{e}_{\mathsf{min}}$ *and* $\overline{e}_{\mathsf{max}}$, $\forall k$, *where* $e_{\mathsf{avg}}^{(k)} = \sum_{n\in\mathcal{N}} D_n^{(k)} e_n^{(k)} / D^{(k)}$. *Then, the cumulative average of the global loss function gradient for PSL satisfies (22), as shown at the bottom of the previous page.*

Proof: Refer to Appendix C for the detailed proof. ■

Corollary 2 (Convergence Under Unified Upperbounds on the Sampling Noise): Under the conditions specified in Corollary 1, further assume (i) bounded stratified sampling noise $\max_{k,n} \left\{ \sum_{j=1}^{S_n^{(k)}} \left(1 - \frac{B_{n,j}^{(k)}}{S_{n,j}^{(k)}}\right) \frac{S_{n,j}^{(k)}}{\left(D_n^{(k)}\right)^2} \frac{\left(S_{n,j}^{(k)} - 1\right)\left(\widetilde{\sigma}_{n,j}^{(k)}\right)^2}{B_{n,j}^{(k)}} \right\} \leq \sigma_{\mathsf{max}}$, $\forall n, k$, *(ii) bounded local iterations* $\max_k \{e_{\mathsf{max}}^{(k)}\} \leq e_{\mathsf{max}}$ *for positive constant* e_{max}, *and (iii) bounded idle period as* $\Omega^{(k)} \leq \left[\frac{\gamma}{K\Delta^{(k)}}\right]^+$, $\forall k$, *for a finite non-negative* γ, *where* $[a]^+ \triangleq \max\{a, 0\}$, $\forall a \in \mathbb{R}$. *Then, the cumulative average of the global loss function gradient for PSL satisfies (27), as shown at the bottom of the next page, implying* $\frac{1}{K}\sum_{k=0}^{K-1} \left\|\nabla F^{(k)}(\mathbf{w}^{(k)})\right\|^2 \leq \mathcal{O}(1/\sqrt{K})$.

Proof: Refer to Appendix D for the detailed proof. ■

One of the key findings of Corollary 2 is that, to have a guaranteed convergence behavior, the idle times must be inversely proportional to the model drift, i.e., $\Omega^{(k)} \leq \left[\frac{\gamma}{K\Delta^{(k)}}\right]^+$, $\forall k$. This implies that larger model drift requires rapid global aggregations (i.e., small idle times), while smaller model drift requires less frequent global aggregations.

We next obtain the data sampling technique that needs to be utilized under a certain data sampling budget at each device:

Proposition 1 (PSL Under Optimal Local Data Sampling): Considering the assumptions made in Theorem 1. For a given mini-batch size at each device n, *i.e.,* $B_n^{(k)}$, *tuning the number of sampled points from strata according to Neyman's sampling technique represented with respect to the variance of the data described by* $B_{n,j}^{(k)} = \left(B_n^{(k)} \widetilde{\sigma}_{n,j}^{(k)} S_{n,j}^{(k)}\right) \left(\sum_{i=1}^{S_n^{(k)}} \widetilde{\sigma}_{n,i}^{(k)} S_{n,i}^{(k)}\right)^{-1}$ *minimizes the bound in (21). Further, if* $\eta_k = \alpha / \sqrt{K e_{\mathsf{sum}}^{(k)} / N}$ *with a finite positive constant* α *chosen to satisfy the condition on* η_k *in Theorem 1 the cumulative average of global gradient under PSL satisfies the bound in (28), as shown at the bottom of the next page.*

Proof: Refer to Appendix E for the detailed proof. ■

The choice of $B_{n,j}^{(k)}$ in Proposition 1 advocates sampling more data points from those strata with higher variance to reduces the SGD noise. This technique is particularly effective when the local datasets are non-i.i.d., e.g., devices possess unbalanced number of data points from different labels.

The bound obtained in Proposition 1 is rather general. In particular, the bound is not obtained under conditions in Corollary 2, which were considered to prove the convergence. This is intentionally done to give a generalized bound that describes the PSL convergence under arbitrary choice of idle times and sampling errors. We thus build our optimization with respect to this bound, making our optimization solver general and applicable to a wide variety of scenarios. We obtain the convergence of PSL when the conditions of Corollaries 1 and 2 are imposed on Proposition 1 in Appendix F (Corollary 3 and 4).

IV. Network-Aware Parallel Successive Learning

In *network-aware PSL*, we aim to jointly optimize the *macro decisions* of the system, e.g., timing of the synchronization signals and idle times between global aggregations, and the *micro decisions* of the system, e.g., local mini-batch sizes and data/parameter offloading ratios, which is among the most general formulations in literature. We formulate the network-aware PSL as the following optimization problem:

$$(\boldsymbol{\mathcal{P}}): \min \frac{1}{K}\left[\sum_{k=0}^{K-1} c_1 E^{\mathsf{Tot},(k)} + c_2 T^{\mathsf{Tot},(k)}\right] + c_3 \underbrace{\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}\left[\left\|\nabla F^{(k)}(\mathbf{w}^{(k)})\right\|^2\right]}_{=\Xi\left(\widehat{\boldsymbol{D}}^{(k)}, \boldsymbol{B}^{(k)}, \Omega^{(k)}, \Delta^{(k)}\right) \text{ given by (28)}} \tag{29}$$

$$\text{s.t. } T^{\mathsf{Tot},(k)} = T^{\mathsf{D},(k)} + T^{\mathsf{L},(k)} + T^{\mathsf{M},(k)} + T^{\mathsf{U},(k)}, \tag{30}$$

$$E^{\mathsf{Tot},(k)} = \sum_{n\in\mathcal{N}} E_n^{(k)}, \tag{31}$$

$$\sum_{k=1}^{K} T^{\mathsf{Tot},(k)} + \Omega^{(k)} = T^{\mathsf{ML}}, \tag{32}$$

$$\max_{n\in\mathcal{N}} \left\{T_n^{\mathsf{DR},(k)}\right\} \leq T^{\mathsf{D},(k)}, \tag{33}$$

$$\max_{n\in\mathcal{N}} \left\{T_n^{\mathsf{C},(k)}\right\} \leq T^{\mathsf{L},(k)}, \tag{34}$$

$$\max_{n\in\mathcal{N}} \left\{T_n^{\mathsf{RG},(k)}\right\} \leq T^{\mathsf{M},(k)}, \tag{35}$$

$$\max_{n\in\mathcal{N}} \left\{T_n^{\mathsf{GT},(k)}\right\} \leq T^{\mathsf{U},(k)}, \tag{36}$$

$$\sum_{m\in\mathcal{N}} \varrho_{n,m}^{(k)} = 1, \ n \in \mathcal{N}, \tag{37}$$

$$\sum_{m\in\mathcal{N}} \varphi_{n,m}^{(k)} = 1, \quad n \in \mathcal{N}, \tag{38}$$

$$\varphi_{n,n}^{(k)} \sum_{m\in\mathcal{N}\setminus\{n\}} \varphi_{n,m}^{(k)} \leq 0, \ n \in \mathcal{N}, \tag{39}$$

$$(1 - \varphi_{n,n}^{(k)}) \sum_{m\in\mathcal{N}\setminus\{n\}} \varphi_{m,n}^{(k)} \leq 0, \ n \in \mathcal{N}, \tag{40}$$

$$f_n^{\mathsf{min}} \leq f_n^{(k)} \leq f_n^{\mathsf{max}}, \ 1 \leq B_n^{(k)} \leq \sum_{m\in\mathcal{N}} \varrho_{m,n}^{(k)} D_m^{(k)}, n \in \mathcal{N}, \tag{41}$$

$$\varrho_{n,m}^{(k)}, \varphi_{n,m}^{(k)} \geq 0, \quad n, m \in \mathcal{N},$$

Variables:

$$K, \left\{\mathbf{e}^{(k)}, \mathbf{f}^{(k)}, \mathbf{B}^{(k)}, \boldsymbol{\varrho}^{(k)}, \boldsymbol{\varphi}^{(k)}, T^{\mathsf{D},(k)}, T^{\mathsf{L},(k)}, T^{\mathsf{M},(k)}, T^{\mathsf{U},(k)}, \Omega^{(k)}\right\}_{k=1}^{K} \tag{42}$$

Objective and variables. $\boldsymbol{\mathcal{P}}$ aims to identify the number of global aggregations K, and the value of the following variables at each global aggregation k: the number of SGD iterations $\mathbf{e}^{(k)} = [e_n^{(k)}]_{n\in\mathcal{N}}$, frequency cycles of the devices $\boldsymbol{f}^{(k)} = [f_n^{(k)}]_{n\in\mathcal{N}}$, mini-batch sizes $\boldsymbol{B}^{(k)} = [B_n^{(k)}]_{n\in\mathcal{N}}$ (given the mini-bath size, the sample size of strata, i.e., $[B_{n,j}^{(k)}]_{n\in\mathcal{N}}, \forall j$, is given by Proposition 1), data offloading ratios $\boldsymbol{\varrho}^{(k)} = [\varrho_{n,m}^{(k)}]_{n,m\in\mathcal{N}}$, model parameter offloading ratios $\boldsymbol{\varphi}^{(k)} = [\varphi_{n,m}^{(k)}]_{n,m\in\mathcal{N}}$, synchronization periods (i.e.,

$T^{\mathsf{D},(k)}, T^{\mathsf{L},(k)}, T^{\mathsf{M},(k)}, T^{\mathsf{U},(k)}$ defined in Sec. II-E.5), and idle times between the global aggregations $\Omega^{(k)}$. The objective function of $\boldsymbol{\mathcal{P}}$ draws a tradeoff between the total energy consumption, device acquisition time/cost, and the ML training performance of the global model. The latter is captured via Ξ, which is characterized by the bound in (28). These (possibly) competing objectives are weighted via non-negative coefficients c_1, c_2, c_3.[9]

Constraints. $T^{\mathsf{Tot},(k)}$ in (30) denotes the device acquisition time and $E^{\mathsf{Tot},(k)}$ in (31) denotes the total energy consumption during global round k, where $E_n^{(k)}$ is given by (16). Constraint (32) ensures that the accumulated time spent during the model training and the idle times equals to the ML model training time T^{ML}. Constraints (33), (34), (35), (36) ensure that the time interval used for data dispersion (see (11)), local computation (see (12)), model dispersion (see (13)), and uplink transmission (see (15)) are chosen to ensure the operation of the system without conflict. Constraints (37) and (38) ensure the proper dispersion of the data and model parameters. Constraints (39) and (40) together ensure that each device either disperses its model parameter or keeps it local and engages in model condensation, i.e., $\phi_{n,n}$ is binary and takes the value of 0 if any portion of the local model is offloaded; and 1 otherwise. Finally, (41), (42) are the feasibility constraints.

Main takeaways. In $\boldsymbol{\mathcal{P}}$, if $c_3 = 0$, and $c_1, c_2 > 0$, model training never occurs ($K = 0$) and devices always remain in the idle state. As c_3 increases, the solution favors a lower model loss via increasing the number of global rounds K and decreasing the idle times $\Omega^{(k)}$-s. Ξ is a function of concept drift ($\Delta_n^{(k)}$-s), especially upon having small concept drifts, once the global model reaches a relatively low loss (i.e., it is *warmed up* and has high *inertia*), performing global aggregations result in marginal performance gains, and thus the frequency of global rounds will be decreased. Also, considering Ξ behavior in (28), the optimization favors larger mini-batch sizes at the devices with higher number of data points and larger data variance. Also, the data is often offloaded from the devices with limited computation resource to those with abundant resources, while the model parameters are often transferred from the devices with poor BS channel conditions to those with better channels.

Behavior of $\boldsymbol{\mathcal{P}}$. Except integer K, all the variables are continuous. Given that $1 \leq K \leq T^{\mathsf{ML}}$, with all the rest of the variables known as a function of K, K can be obtained with an exhaustive search. We thus focus on obtaining the rest of variables for a given K. In $\boldsymbol{\mathcal{P}}$, multiplication between optimization variables appear in multiple places. For example, in $E_n^{\mathsf{C},(k)}$, encapsulated in $E_n^{(k)}$ (see (16)) in (31), multiplication between $e_n^{(k)}$, $B_n^{(k)}$, and $f_n^{(k)}$ exist (see (18)). Similar phenomenon exist in $T_n^{\mathsf{C},(k)}$ (see (12)) in (34). More importantly, the definition of Ξ in (28) consists of multiple terms with multiplication of variables some of which with negative coefficients. In particular, the problem belongs to the category of *Signomial Programming*, which are highly non-convex and NP-hard [45].

[9]We trivially defined $E^{\mathsf{Tot},(0)} = T^{\mathsf{Tot},(0)} = 0$.

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\nabla F^{(k)}(\mathbf{w}^{(k)})\right\|^2\right] \leq 2\sqrt{\widehat{e}_{\mathsf{max}}}\frac{F^{(-1)}(\mathbf{w}^{(0)}) - F^{(K)\star}}{\overline{e}_{\mathsf{min}}\alpha\sqrt{NK}(1-\Lambda_{\mathsf{max}})} + \frac{2\sqrt{\widehat{e}_{\mathsf{max}}}\gamma}{\overline{e}_{\mathsf{min}}\alpha\sqrt{NK}(1-\Lambda_{\mathsf{max}})} + \frac{4\overline{e}_{\mathsf{max}}\alpha\Theta^2\beta\sqrt{N}}{(1-\Lambda_{\mathsf{max}})\sqrt{\widehat{e}_{\mathsf{min}}}\sqrt{K}}\sigma_{\mathsf{max}}$$
$$+ \frac{1}{K}\frac{1}{(1-\Lambda_{\mathsf{max}})}\left(8\beta^2\Theta^2\alpha^2 N(e_{\mathsf{max}}-1)\sigma_{\mathsf{max}}/\widehat{e}_{\mathsf{min}} + 8\zeta_2\alpha^2 N\beta^2\left(e_{\mathsf{max}}\right)(e_{\mathsf{max}}-1)/\widehat{e}_{\mathsf{min}}\right) \tag{27}$$

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla F^{(k)}(\mathbf{w}^{(k)})\|^2\right] \leq \Xi\left(\widehat{\boldsymbol{D}}^{(k)}, \boldsymbol{B}^{(k)}, \Omega^{(k)}, \Delta^{(k)}\right) \triangleq \underbrace{\frac{2\sqrt{\widehat{e}_{\mathsf{max}}}\left(F^{(-1)}(\mathbf{w}^{(0)}) - F^{(K)\star}\right)}{\alpha\overline{e}_{\mathsf{min}}\sqrt{NK}(1-\Lambda_{\mathsf{max}})}}_{(a)} + \sum_{k=0}^{K-1}\frac{2\sqrt{e_{\mathsf{sum}}^{(k)}}\Omega^{(k+1)}\Delta^{(k+1)}}{\alpha e_{\mathsf{avg}}^{(k)}\sqrt{NK}(1-\Lambda^{(k)})}$$
$$+ \sum_{k=0}^{K-1}\frac{1}{(1-\Lambda^{(k)})}\left(\frac{8\beta^2\Theta^2\alpha^2 N}{e_{\mathsf{sum}}^{(k)}K^2}\sum_{n\in\mathcal{N}}\frac{\widehat{D}_n^{(k)}}{D^{(k)}}(e_n^{(k)}-1)\frac{1}{\left(\widehat{D}_n^{(k)}\right)^2}\left[\frac{1}{B_n^{(k)}}\left(\sum_{j=1}^{S_n^{(k)}}\widetilde{\sigma}_{n,j}^{(k)}S_{n,j}^{(k)}\right)^2 - \underbrace{\sum_{j=1}^{S_n^{(k)}}S_{n,j}^{(k)}\left(\widetilde{\sigma}_{n,j}^{(k)}\right)^2}_{(b)}\right]\right.$$
$$\left. + \frac{8\zeta_2\alpha^2\beta^2 N}{e_{\mathsf{sum}}^{(k)}K^2}\left(e_{\mathsf{max}}^{(k)}\right)\left(e_{\mathsf{max}}^{(k)}-1\right)\right) + \sum_{k=0}^{K-1}\frac{4e_{\mathsf{avg}}^{(k)}\alpha\Theta^2\beta\sqrt{N}}{2\sqrt{e_{\mathsf{sum}}^{(k)}}K\sqrt{K}(1-\Lambda^{(k)})}\sum_{n\in\mathcal{N}}\frac{1}{\left(D^{(k)}\sqrt{e_n^{(k)}}\right)^2}$$
$$\times\left[\frac{1}{B_n^{(k)}}\left(\sum_{j=1}^{S_n^{(k)}}\widetilde{\sigma}_{n,j}^{(k)}S_{n,j}^{(k)}\right)^2 - \underbrace{\sum_{j=1}^{S_n^{(k)}}S_{n,j}^{(k)}\left(\widetilde{\sigma}_{n,j}^{(k)}\right)^2}_{(c)}\right] \tag{28}$$

In the following, we provide a tractable technique to solve $\boldsymbol{\mathcal{P}}$.[10] Although our approach is developed for $\boldsymbol{\mathcal{P}}$, it can be applied to a broader category of problems that we call *network-aware distributed ML*, where the formulations are mostly concerned with optimizing the performance of the ML training under network constraints. We are among the first to introduce these optimization techniques to distributed ML literature.

A related problem format in literature to $\boldsymbol{\mathcal{P}}$ is *geometric programming* (GP), to understand which the knowledge of monomial and posynomials is necessary.

Definition 3: A ***monomial*** *is defined as a function* $f: \mathbb{R}^n_{++} \rightarrow \mathbb{R}$: $f(\boldsymbol{y}) = z y_1^{\alpha_1} y_2^{\alpha_2} \cdots y_n^{\alpha_n}$, *where* $z \geq 0$, $\boldsymbol{y} = [y_1, \cdots, y_n]$, *and* $\alpha_j \in \mathbb{R}$, $\forall j$. *Further, a* ***posynomial*** g *is defined as a sum of monomials:* $g(\boldsymbol{y}) = \sum_{m=1}^{M} z_m y_1^{\alpha_m^{(1)}} y_2^{\alpha_m^{(2)}} \cdots y_n^{\alpha_m^{(n)}}$, $z_m \geq 0, \forall m$.

A GP in its standard form admits a posynomials objective function subject to inequality constraints on posynomials and equality constraints on monomials (see Appendix G-A). With a logarithmic change of variables, GP in its standard form can be transformed into a convex optimization that can be efficiently solved using well-known software, e.g., CVXPY [46]. Nevertheless, $\boldsymbol{\mathcal{P}}$ does not admit the format of GP. In particular, the bound in (28) appearing in the objective function consists of terms with negative sign that violate the definition of posynomials. Furthermore, constraints (32), (37) and (38) are equalities on posynomials, which GP does not admit. To tackle these violating terms, we first consider the constraints in the form of equality on posynomials, and use the method of penalty functions and auxiliary variables [47]. To this end, we consider each equality on a posynomial in the format of $g(\boldsymbol{x}) = c$ as two inequality constraints: (c-i) $g(\boldsymbol{x}) \leq c$, and (c-ii) $1/(Ag(\boldsymbol{x})) \leq c$, where $A \geq 1$ is an auxiliary variable, which will later be forced $A \downarrow 1$ via being added to the objective function with a penalty coefficient.[11] Inequality (c-i) is an inequality on a posynomial, which GP admits. However, (c-ii) is an inequality on a non-posynomial (division of a monomial/posynomial by a posynomial is not a posynomial). One way to transform (c-ii) to an inequality on a posynomial is to approximate its denominator by a monomial (division of a posynomial by a monomial is a posynomial). To this end, we exploit the arithmetic-geometric mean inequality.

Lemma 2 (Arithmetic-Geometric Mean Inequality [48]): Consider a posynomial function $g(\boldsymbol{y}) = \sum_{i=1}^{i'} u_i(\boldsymbol{y})$, *where* $u_i(\boldsymbol{y})$ *is a monomial,* $\forall i$. *The following inequality holds*:

$$g(\boldsymbol{y}) \geq \hat{g}(\boldsymbol{y}) \triangleq \prod_{i=1}^{i'} \left(u_i(\boldsymbol{y})/\alpha_i(\boldsymbol{z})\right)^{\alpha_i(\boldsymbol{z})}, \tag{49}$$

where $\alpha_i(\boldsymbol{z}) = u_i(\boldsymbol{z})/g(\boldsymbol{z})$, $\forall i$, *and* $\boldsymbol{z} > 0$ *is a fixed point.*

In Appendix G, we explain all the steps taken to solve the optimization problem $\boldsymbol{\mathcal{P}}$. Due to space limitations, we provide a high level discussion in the following.

Our technique to solve $\boldsymbol{\mathcal{P}}$ is an iterative approach, where at each iteration ℓ, we use the aforementioned method of penalty functions based on (c-i) and (c-ii). The corresponding posynomials in (c-ii) for (32), (37) and (38) are approximated using (43), (44), and (45), as shown at the bottom of the next page, respectively. Furthermore, since $\widehat{D}_n^{(k)}$, $e_{\mathsf{sum}}^{(k)}$ and $e_{\mathsf{avg}}^{(k)}$ appear in multiple places in (28), for tractability, we treat them as optimization variables and add the following constraints to $\boldsymbol{\mathcal{P}}$: $(\widehat{D}_n^{(k)})^{-1} \sum_{m \in \mathcal{N}} \varrho_{m,n}^{(k)} D_m^{(k)} = 1, n \in \mathcal{N}$, $\sum_{n \in \mathcal{N}} e_n^{(k)}/e_{\mathsf{sum}}^{(k)} = 1, \sum_{n \in \mathcal{N}} \widehat{D}_n^{(k)} e_n^{(k)}/(e_{\mathsf{avg}}^{(k)} D^{(k)}) = 1$, which are all equality constraint on posynomials. We thus use the method of penalty functions with approximations given in (46), (47) and (48), as shown at the bottom of the next page, to transform them. It is easy to verify that (43)-(48) are in fact the best local monomial approximations to their corresponding posynomials near fixed point $\boldsymbol{x}^{[\ell-1]}$ in terms of the first-order Taylor approximation (vector $\boldsymbol{x}$ encapsulates all the optimization variables in all the expressions).

We next tackle the complex term Ξ in the objective function of $\boldsymbol{\mathcal{P}}$, i.e., (28). In (28), we first upper bound $F(\mathbf{w}^{(0)}) - F^{(K)^\star}$ (inside the term in the first line) with $F(\mathbf{w}^{(0)})$ and $e_n^{(k)} - 1$ with $e_n^{(k)}$ (inside the term in the second line), and $e_{\mathsf{max}}^{(k)} - 1$ with $e_{\mathsf{max}}^{(k)}$ (inside the term in the third line) since these do not impose a notable difference in the optimization solution. Also, to have a tractable solution, we assume that (i) the relative size of the strata to the size of the local dataset is upper bounded throughout the learning period, and let $s_{n,j} \leq 1$ denote the upper bound of the relative size of stratum $\mathcal{S}_{n,j}^{(k)}$ to the local dataset, i.e., $S_{n,j}^{(k)}/\widehat{D}_n^{(k)} \leq s_{n,j}$, $\forall k$, and (ii) the optimizer optimizes the ML bound for an upper bound on the variance of the local strata (i.e., $\widetilde{\sigma}_{n,j}^{(k)}$, $\forall k, n, j$, is upper bounded via historical data for the optimizer); however, during the PSL model training each node uses Lemma 1 to track the variance of its strata, based on which it conducts non-uniform data sampling according to the rule obtained in Proposition 1. These upper bounds and assumptions are inherently assumed in Proposition 2. Note that in (28), all the terms contain the summation over global aggregation index k, i.e., $\sum_{k=1}^{K}$, expect the first term (term (a)), which in turn can be upper bounded as $\sum_{k=1}^{K} \frac{2\sqrt{\widehat{e}_{\mathsf{max}}} F(\mathbf{w}^{(0)})}{\alpha \overline{e}_{\mathsf{min}} K \sqrt{K}(1-\Lambda_{\mathsf{max}})}$. We thus consider Ξ in (28) as $\Xi = \sum_{k=1}^{K} \sigma_+^{(k)}(\boldsymbol{x}) - \sigma_{-,1}^{(k)}(\boldsymbol{x}) - \sigma_{-,2}^{(k)}(\boldsymbol{x})$, where $\sigma_+^{(k)}(\boldsymbol{x})$ contains all the terms with positive coefficients, while $\sigma_{-,1}^{(k)}(\boldsymbol{x})$, $\sigma_{-,2}^{(k)}(\boldsymbol{x})$ are two terms with negative sign (terms (b) and (c) and their coefficients). We next replace the term $\gamma\Xi$ in the objective function of $\boldsymbol{\mathcal{P}}$ with $\gamma \sum_{k=0}^{K-1} \chi^{(k)}$, which auxiliary variable $\chi^{(k)}$ is the upperbound of summand in Ξ, which we add it to the constraints as $\sigma_+^{(k)}(\boldsymbol{x}) - \sigma_{-,1}^{(k)}(\boldsymbol{x}) - \sigma_{-,2}^{(k)}(\boldsymbol{x}) \leq \chi^{(k)}$, $\forall k$. Now we focus on this constraint, which can be written as follows:

$$\sigma_+^{(k)}(\boldsymbol{x}) \Big/ \left(\chi^{(k)} + \sigma_{-,1}^{(k)}(\boldsymbol{x}) + \sigma_{-,2}^{(k)}(\boldsymbol{x})\right) \leq 1. \tag{51}$$

Considering the fraction in (51), all the terms encapsulated in $\sigma_+^{(k)}(\boldsymbol{x})$ are posynomials, making its numerator a posynomial. However, its denominator is also a posynomial, making it a non-posynomial fraction. We thus focus on the approximation of the denominator with a monomial, for which we exploit Lemma 2 (see Appendix G for the detailed steps), the result of which is given by (50), as shown at the bottom of the next page. After the conducted approximations, we obtain a problem in which the objective function is a posynomial and all the constraints are inequalities on posynomials admitting

[10] To be able to solve the problem, we rely on strict positive optimization variables and replace constraint (42) with $\varrho_{n,m}^{(k)}, \varphi_{n,m}^{(k)} > 0, \forall n, m$. Accordingly, inequalities in (39) and (40) in the form of $A(\boldsymbol{x}) \leq 0$ are replaced with $A(\boldsymbol{x}) < \vartheta$, where $\vartheta > 0$ is an optimization variable. ϑ is then added to the objective function with a penalty term to ensure $\vartheta \downarrow 0$ at the final solution.

[11] $A \downarrow 1$ is an equivalent representation of $A \rightarrow 1^+$.

the GP format (see the problem in (186)-(210) in Appendix G). We provide the pseudo-code of our optimization solver in Algorithm 1. We next show the convergence guarantees of our solver.

Proposition 2 (Optimization Solver): For each K, Algorithm 1 generates a sequence of solutions for $\boldsymbol{\mathcal{P}'}$ using the approximations (43)-(48),(50) that converge to $\boldsymbol{x}_K^\star$ satisfying the Karush-Kuhn-Tucker (KKT) conditions of $\boldsymbol{\mathcal{P}}$.

Proof: It can be shown that $\boldsymbol{\mathcal{P}'}$ (given by (186)-(210)) is an *inner approximation* [49] of $\boldsymbol{\mathcal{P}}$. Thus, it is sufficient to examine the three characteristics mentioned in [49] for Algorithm 1 to prove the convergence, which can be done using the methods in [47] and [50] and omitted for brevity. ■

V. Simulation Results

We next evaluate the effectiveness of PSL. We incorporate the effect of fading in $h_n^{(k)}$, $h_{n,m}^{(k)}$ (in (8) and (9)). For the uplink channel $h_n^{(k)} = \sqrt{\beta_n^{(k)}} u_n^{(k)}$, where $u_n^{(k)} \sim \mathcal{CN}(0,1)$ captures Rayleigh fading, and $\beta_n^{(k)} = \beta_0 - 10\widetilde{\alpha}\log_{10}(d_n^{(k)}/d_0)$ [51]. Here, $\beta_0 = -30$dB, $d_0 = 1$m, $\widetilde{\alpha} = 3$, and $d_n^{(k)}$ is the instantaneous distance between node n and the BS. We use the same formula to describe the D2D channels but choose $\widetilde{\alpha} = 3.2$ since D2D are more prone to excessive loss. Channels are realized with coherence time of 50 ms. The devices are randomly placed in a circle area with radius of 25m with a BS in the center. Our simulations were implemented using Pytorch [52] and run on three Nvidia Tesla V100 GPUs

$$H(\boldsymbol{x}) = \sum_{k=1}^{K} T^{\mathsf{Tot},(k)} + \Omega^{(k)} \geq \widehat{H}(\boldsymbol{x};\ell) \triangleq \prod_{k=1}^{K} \left(\frac{T^{\mathsf{D},(k)} H([\boldsymbol{x}]^{\ell-1})}{\left[T^{\mathsf{D},(k)}\right]^{\ell-1}}\right)^{\frac{\left[T^{\mathsf{D},(k)}\right]^{\ell-1}}{H([\boldsymbol{x}]^{\ell-1})}} \left(\frac{T^{\mathsf{L},(k)} H([\boldsymbol{x}]^{\ell-1})}{\left[T^{\mathsf{L},(k)}\right]^{\ell-1}}\right)^{\frac{\left[T^{\mathsf{L},(k)}\right]^{\ell-1}}{H([\boldsymbol{x}]^{\ell-1})}} \left(\frac{T^{\mathsf{M},(k)} H([\boldsymbol{x}]^{\ell-1})}{\left[T^{\mathsf{M},(k)}\right]^{\ell-1}}\right)^{\frac{\left[T^{\mathsf{M},(k)}\right]^{\ell-1}}{H([\boldsymbol{x}]^{\ell-1})}} \left(\frac{T^{\mathsf{U},(k)} H([\boldsymbol{x}]^{\ell-1})}{\left[T^{\mathsf{U},(k)}\right]^{\ell-1}}\right)^{\frac{\left[T^{\mathsf{U},(k)}\right]^{\ell-1}}{H([\boldsymbol{x}]^{\ell-1})}} \left(\frac{\Omega^{(k)} H([\boldsymbol{x}]^{\ell-1})}{\left[\Omega^{(k)}\right]^{\ell-1}}\right)^{\frac{\left[\Omega^{(k)}\right]^{\ell-1}}{H([\boldsymbol{x}]^{\ell-1})}} \tag{43}$$

$$G(\boldsymbol{x}) = \sum_{m\in\mathcal{N}} \varrho_{n,m}^{(k)} \geq \widehat{G}(\boldsymbol{x};\ell) \triangleq \prod_{m\in\mathcal{N}} \left(\frac{\varrho_{n,m}^{(k)} G([\boldsymbol{x}]^{\ell-1})}{\left[\varrho_{n,m}^{(k)}\right]^{\ell-1}}\right)^{\frac{\left[\varrho_{n,m}^{(k)}\right]^{\ell-1}}{G([\boldsymbol{x}]^{\ell-1})}} \tag{44}$$

$$J(\boldsymbol{x}) = \sum_{m\in\mathcal{N}} \varphi_{n,m}^{(k)} \geq \widehat{J}(\boldsymbol{x};\ell) \triangleq \prod_{m\in\mathcal{N}} \left(\frac{\varphi_{n,m}^{(k)} J([\boldsymbol{x}]^{\ell-1})}{\left[\varphi_{n,m}^{(k)}\right]^{\ell-1}}\right)^{\frac{\left[\varphi_{n,m}^{(k)}\right]^{\ell-1}}{J([\boldsymbol{x}]^{\ell-1})}} \tag{45}$$

$$I(\boldsymbol{x}) = \sum_{m\in\mathcal{N}} \varrho_{m,n}^{(k)} D_m^{(k)} \geq \widehat{I}(\boldsymbol{x};\ell) \triangleq \prod_{m\in\mathcal{N}} \left(\frac{\varrho_{m,n}^{(k)} I([\boldsymbol{x}]^{\ell-1})}{[\varrho_{m,n}^{(k)}]^{\ell-1}}\right)^{\frac{D_m^{(k)}[\varrho_{m,n}^{(k)}]^{\ell-1}}{I([\boldsymbol{x}]^{\ell-1})}} \tag{46}$$

$$R(\boldsymbol{x}) = \sum_{n\in\mathcal{N}} e_n^{(k)} \geq \widehat{R}(\boldsymbol{x};\ell) \triangleq \prod_{n\in\mathcal{N}} \left(\frac{e_n^{(k)} R([\boldsymbol{x}]^{\ell-1})}{[e_n^{(k)}]^{[\ell-1]}}\right)^{\frac{[e_n^{(k)}]^{[\ell-1]}}{R([\boldsymbol{x}]^{\ell-1})}} \tag{47}$$

$$V(\boldsymbol{x}) = \sum_{n\in\mathcal{N}} \widehat{D}_n^{(k)} e_n^{(k)} \geq \widehat{V}(\boldsymbol{x};\ell) \triangleq \prod_{n\in\mathcal{N}} \prod_{m\in\mathcal{N}} \left(\frac{\varrho_{m,n}^{(k)} e_n^{(k)} V([\boldsymbol{x}]^{\ell-1})}{\left[\varrho_{m,n}^{(k)} e_n^{(k)}\right]^{[\ell-1]}}\right)^{\frac{D_m^{(k)}\left[\varrho_{m,n}^{(k)} e_n^{(k)}\right]^{[\ell-1]}}{V([\boldsymbol{x}]^{\ell-1})}} \tag{48}$$

$$W(\boldsymbol{x}) = \chi^{(k)} + \frac{1}{(1-\Lambda^{(k)})} \frac{8\beta^2\Theta^2\alpha^2 N}{K^2} (e_{\mathsf{sum}}^{(k)})^{-1} \sum_{n\in\mathcal{N}} \frac{e_n^{(k)}}{D^{(k)}} \widehat{Z}_n^{(k)} + \frac{4\Theta^2\beta\alpha\sqrt{N}}{K\sqrt{K}(1-\Lambda^{(k)})} \left(e_{\mathsf{avg}}^{(k)}\right) \left(e_{\mathsf{sum}}^{(k)}\right)^{-1/2} \sum_{n\in\mathcal{N}} \frac{\widehat{D}_n^{(k)}}{\left(D^{(k)}\right)^2 e_n^{(k)}} \widehat{Z}_n^{(k)}$$
$$\geq \widehat{W}(\boldsymbol{x};\ell) \triangleq \left(\frac{\chi^{(k)} W([\boldsymbol{x}]^{\ell-1})}{[\chi^{(k)}]^{\ell-1}}\right)^{\frac{[\chi^{(k)}]^{\ell-1}}{W([\boldsymbol{x}]^{\ell-1})}} \times \prod_{n\in\mathcal{N}} \prod_{q=1}^{2} \left(\frac{\delta_q(\boldsymbol{x},n) W([\boldsymbol{x}]^{\ell-1})}{\delta_q([\boldsymbol{x}]^{\ell-1},n)}\right)^{\frac{\delta_q([\boldsymbol{x}]^{\ell-1},n)}{W([\boldsymbol{x}]^{\ell-1})}}, \tag{49}$$

$$\delta_1(\boldsymbol{x},n) = \frac{1}{(1-\Lambda^{(k)})} \frac{8\beta^2\Theta^2\alpha^2 N}{K^2} \frac{e_n^{(k)}}{e_{\mathsf{sum}}^{(k)} D^{(k)}} \widehat{Z}_n^{(k)}, \quad \delta_2(\boldsymbol{x},n) = \frac{4\Theta^2\beta\alpha\sqrt{N}}{K\sqrt{K}(1-\Lambda^{(k)})} \frac{e_{\mathsf{avg}}^{(k)} \widehat{D}_n^{(k)}}{\sqrt{e_{\mathsf{sum}}^{(k)}} \left(D^{(k)}\right)^2 e_n^{(k)}} \widehat{Z}_n^{(k)}, \quad \widehat{Z}_n^{(k)} = \sum_{j=1}^{S_n^{(k)}} s_{n,j} \left(\widetilde{\sigma}_{n,j}^{(k)}\right)^2 \tag{50}$$

Algorithm 1 Optimization Solver for Problem $\mathcal{P}$

input : Convergence criterion, model training duration T^{ML}.
1 **for** $K = 1$ *to* T^{ML} **do**
2 Set the iteration count $\ell = 0$.
3 Choose a feasible point $\boldsymbol{x}^{[0]}$.
4 Obtain the monomial approximations (43)-(48),(50) given $\boldsymbol{x}^{[\ell]}$.
5 Replace the results in the approximation of Problem $\mathcal{P}$ (i.e., $\mathcal{P}'$ given by (186)-(210) in Appendix G).
6 With logarithmic change of variables, transform the resulting GP problem to a convex problem (as described in Appendix G-A).
7 $\ell = \ell + 1$
8 Obtain the solution of the convex problem using current art solvers (e.g., CVXPY [46]) to determine $\boldsymbol{x}^{[\ell]}$.
9 **if** *two consecutive solutions* $\boldsymbol{x}^{[\ell-1]}$ *and* $\boldsymbol{x}^{[\ell]}$ *do not meet the specified convergence criterion* **then**
10 Go to line 4 and redo the steps using $\boldsymbol{x}^{[\ell]}$.
11 **else**
12 Set the solution of the iterations as $\boldsymbol{x}_K^\star = \boldsymbol{x}^{[\ell]}$.
13 $\boldsymbol{x}^\star = \min_{\{\boldsymbol{x}_K^\star\}_{1 \le K \le T^{\mathsf{ML}}}} \{\text{Objective of } \mathcal{P} \text{ evaluated at } \boldsymbol{x}_K^\star\}$

TABLE I
PSL NETWORK SAVINGS VS THE BASELINE METHOD (FED-NOVA [40]). WE MEASURE NETWORK SAVINGS TO REACH ACCURACY THRESHOLDS BY THE COMBINATION OF ENERGY AND DEVICE ACQUISITION TIME (DAT)

	Acc	Static Datasets		Time Varying Datasets	
		Energy (kJ)	DAT (s)	Energy (kJ)	DAT (s)
FMNIST	40%	11.7 (50%)	730 (50%)	11.7 (33%)	730 (33%)
	50%	23.3 (40%)	1461 (40%)	17.5 (30%)	1095 (30%)
	60%	64.1 (42%)	4017 (42%)	58.3 (38%)	3652 (38%)
MNIST	60%	35.0 (86%)	2191 (86%)	35.0 (86%)	2191 (86%)
	70%	64.1 (85%)	4017 (85%)	64.1 (85%)	4017 (85%)
	80%	140 (92%)	8765 (92%)	117 (91%)	7304 (91%)

with 32 GB VRAM each. We used CVXPY [46] to obtain the solutions of the convex problems obtained through Alg. 1.

A. Dynamic Model Training Under PSL: Proof-of-Concept

We compare the ML performance of PSL against the baseline method Fed-Nova [40], a state-of-the-art FedL method that also accounts for varying SGD iterations across the devices. Fed-Nova aims to obtain unbiased global models for FedL by normalizing devices' received gradients at the server with respect to their number of conducted local SGD iterations, which is shown to significantly outperform existing FedL methods including FedAvg, FedProx, and VRLSGD [40]. We consider classification tasks over MNIST [53] and Fashion-MNIST [54] datasets for a network of 10 devices in Fig. 6 and Table I.[12] Both datasets consist of 60K training samples and 10k testing samples, where each datum belongs to one of 10 labels. We consider both time-varying and static device datasets. In the dynamic case, devices obtain a *new* dataset of size drawn from a normal distribution $\mathcal{N}(1000, 125)$ after each global aggregation. In the static case, devices use a fixed dataset with size drawn from $\mathcal{N}(1000, 125)$. Device datasets are only composed of data from 3 distinct labels. To have a fair comparison, we isolate the ML performance and consider both methods with *no data offloading*. The maximum and minimum number of local SGDs are considered 25 and 1 respectively. Fig. 6 shows that PSL outperforms the baseline method in both cases owed to its local data management and non-uniform data sampling.

[12]The results in Fig. 6 and Table I are the averaged results obtained from 10 Monte-Carlo iterations of independent network realizations.

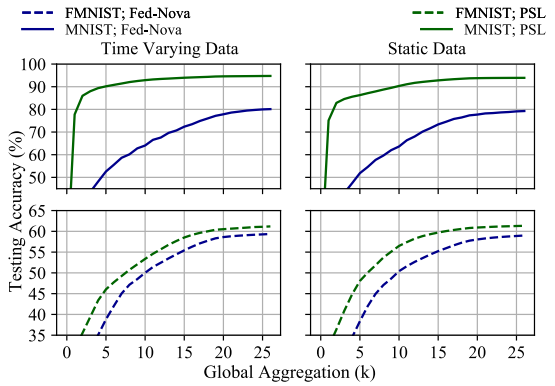

Fig. 6. Accuracy obtained using PSL vs. the baseline method (Fed-Nova [40]). Left subplot: time varying local datasets across the global aggregations. Right subplot: static local datasets across the global aggregations. The results are obtained using moving average with window size of 10.

We quantify the corresponding resource savings of PSL in Table I. The jumps in energy consumption when moving from 50% to 60% accuracy on FMNIST and 70% to 80% accuracy on MNIST are due to the natural saturation in training improvement of the federated ML methods upon reaching higher accuracies. For example, to improve from 40% to 50% accuracy on FMNIST, PSL uses 3 additional aggregations while Fed-Nova requires 5 additional aggregations. Meanwhile, to improve from 50% to 60% classification accuracy on FMNIST, PSL needs 10 further aggregations while Fed-Nova uses 16 further aggregations.

B. Network-Aware PSL: Ablation Study

Direct examination of $\mathcal{P}$ is difficult due to the complexity and entanglement of the optimization variables and elements of the bound in (28). As a result, we perform an ablation study – systematically evaluating the impacts of important optimization and scaling variables in isolation – to characterize $\mathcal{P}$ in detail. We use the set of network characteristics in Table II. The results in Sections V-B.1, V-B.3, V-B.5, V-B.7 are the averaged results over 10 Monte-Carlo iterations of independent network realizations. Sections V-B.2, V-B.4, V-B.6 focus on showing discretized numerical values and, in order to preserve the key intuition behind the results, we show the result for a single network realization.

1) Optimization Solver and Network Size: We first investigate the convergence of our optimization solver for various network sizes in Fig. 7, where $T^{\mathsf{ML}} = 1000$s, $K = 2$, and $N \in \{5, 10, 15, 20\}$. As can be seen, larger number of devices leads to slower convergence but a better final solution since it causes (i) processing higher number of data points across the devices that leads to a better ML performance, and (ii) more efficient D2D data/model transfer opportunity. Furthermore, Fig. 7 reveals the diminishing rewards of increasing the number of devices, where an initial increase from $N = 5$ to $N = 10$ results in a notable performance improvement; however, this effect is less notable as the number of devices increase. This implies that engaging more devices may allow for more efficient model training but there is a point after which the network energy consumption (due to data processing and model aggregations) overshadows the ML performance gains.

2) Model/Concept Drift vs. Idle Time: We investigate the effect of the model drift $\Delta^{(k)}$ on the system idle time in Fig. 8, where $T^{\mathsf{ML}} = 5000$s, and $K = 10$. The figure shows that our solution promotes rapid global aggregations when the value of model drift increases. Further, when the value of concept drift

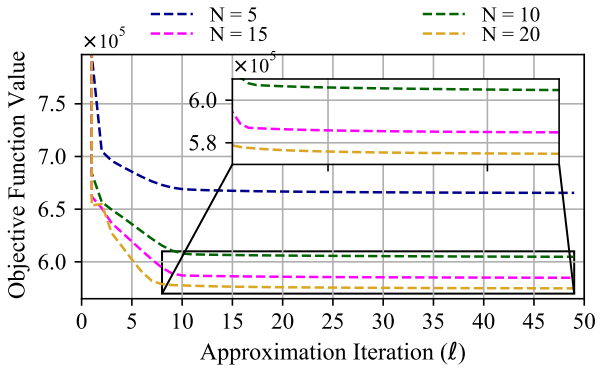

Fig. 7. Convergence of the objective function of $\mathcal{P}$ upon having varying number of devices N for $T^{\mathsf{ML}} = 1000s$, where the ML performance term in the objective function is prioritized over the energy and delay terms.

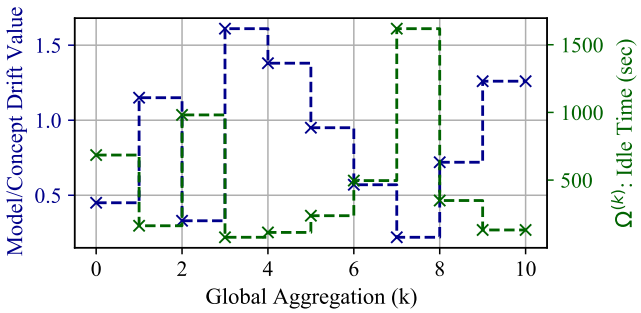

Fig. 8. Demonstration of the variation of the idle time in between the consecutive global aggregations (right y-axis) for a given configuration of model/concept drift (left y-axis).

TABLE II

NETWORK CHARACTERISTICS USED FOR ABLATION STUDY. THE EXPERIMENTS ALL USE THE NETWORK VALUES HEREIN, UNLESS INDICATED OTHERWISE

Param	Value	Param	Value	Param	Value
b^{D}	$32 \times 32 \times 4$	p_n^{U}	250mW	Δ	0.1
$M \times b^{\mathsf{G}}$	72000	p_n^{D}	100mW	Λ	0.9
$f^{\max}$	2.3GHz	N_0	-174dBm/Hz	α	0.1
$f^{\min}$	100KHz	B^{U}	1MHz	ζ_2	$1e-5$
α_n	$2e-13$	B^{D}	100kHz	Θ	3

is low, the global aggregations are carried out via higher idle times: since the data variations at devices is small, they can stay in idle state to save energy and device acquisition cost.

3) Model Dissimilarity vs. Local SGD Iterations: ζ_2 in (28) quantifies data dissimilarity, where higher data dissimilarity increases the chance of local model bias with increased local SGD iterations. In Fig. 9, we vary dissimilarity and depict the number of SGD iterations across the devices, where $T^{\mathsf{ML}} = 100$s. Each device n requires a_n CPU cycles to process each datum, thus a large a_n indicates a higher data processing cost. Fig. 9 shows that when ζ_2 is small, i.e., data is homogeneous across the devices, PSL maximizes ML performance by having more local SGD iterations at devices with small a_n ($n \in \{3,4,5\}$). As ζ_2 increases, uneven local SGD iterations can favor local models at devices with higher SGD iterations, so PSL reduces the variance of the number SGD iterations among the devices.

4) Cold vs. Warmed Up Model, and Model Inertia: Model training interval T^{ML} limits the time for all aspects of PSL (i.e., data/model transfer, local processing, and uplink transmissions). In Fig. 10, we depict the optimal number of global aggregations for a wide range of T^{ML} values. The left subplot shows the objective function value as a function of the number of global aggregations K. It can be seen that

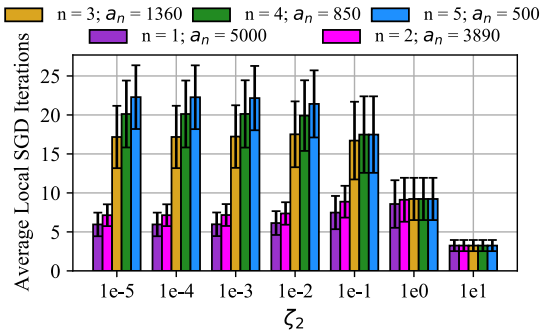

Fig. 9. Average local SGD iterations for different devices through the model training period upon having different degree of model dissimilarity ζ_2.

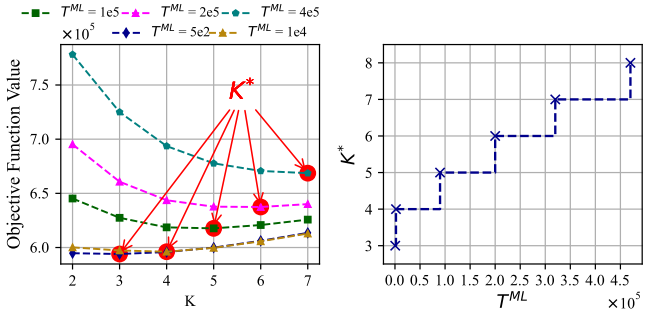

Fig. 10. Optimal number of global aggregations $K^\star$ for different period of model training intervals T^{ML}.

increasing the number of global aggregations K is not suitable for all scenarios. Large K implies that the system must spend more *time* on the data and model transmission processes and therefore has less time to run model training. As a result, for smaller T^{ML}, $K^\star$ is smaller, and $K^\star$ increases as T^{ML} increases. In the right subplot of Fig. 10, we analyze the relationship between T^{ML} and $K^\star$. Initially, i.e., having a cold model, $K^\star$ increases rapidly as a function of T^{ML}, however, as the model gets warmed up increasing $K^\star$ calls for larger increments in T^{ML} which signifies the model inertia (i.e., for a warmed up model, to trigger a new model training round larger data variations are needed so that ML model gains outweighs the network costs).

We next sequentially focus on the scaling of energy, acquisition time, and ML bound terms in problem $\mathcal{P}$.

5) Energy Scaling in Optimization Objective: c_1 in the objective of $\mathcal{P}$ controls the importance of the energy components. To demonstrate the effect of c_1, we focus on the data processing (measured via mini-batch size) and offloading (measured via net data received) in Fig. 11, where $T^{\mathsf{ML}} = 100$s and $\zeta_2 = 1e-3$. In the first column of Fig. 11, c_1 increases while is still small, which leads to less data processing at devices with high processing cost (i.e., $n = 2$) and more data offloading from them. The middle column accelerates the increment of c_1, showing rapid increase of data reception and processing at device $n = 5$. Here, even lower processing cost devices ($n = 4$) begin offloading data and processing less data. In the last column, data processing becomes almost prohibitive, and consequently, even device $n = 5$ processes only a subset of its data. Even though device $n = 5$ processes only a subset of its data, it is still beneficial for all devices to transfer data to device $n = 5$ in order to reduce their SGD noise (they choose mini-batch sizes of 1 so lower number of local data implies a lower SGD noise) and thereby reduce the ML term Ξ.

6) Temporal Importance in Optimization Objective: The cost of device acquisition, c_2, determines the balance of

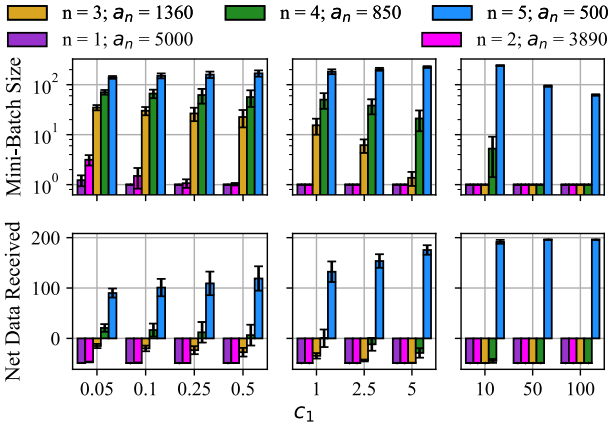

Fig. 11. Top row: the average mini-batch size through model training period. Bottom row: the net number of received data at the devices (negative values imply data offloading) through model training period. Different ranges of energy importance c_1 in $\mathcal{P}$ are considered.

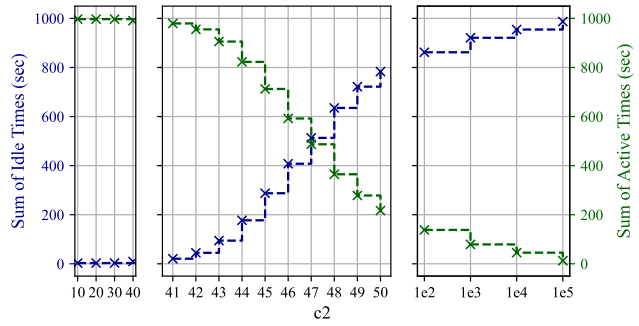

Fig. 12. Device acquisition time (right y-axis) and device idle times (left y-axis) for varying values of cost of device utilization c_2 in $\mathcal{P}$. As c_2 increases the device utilization time decreases and devices remain idle for longer times.

device acquisition and idle times. We investigate its influence in Fig. 12, where $T^{\mathsf{ML}} = 1000$s. It can be seen that a decrease in the device acquisition/active times corresponds to an increase in the idle times. This trade-off occurs in three regimes, segmented into three subplots. Initially, increasing c_2 has no effect until a network dependent threshold (in this case $c_2 = 40$) is reached. After this threshold, we enter a regime in which incremental increase of c_2 results in a notable increase in idle times and a proportional decrease in active times. Finally, in the third regime, the marginal reward of reducing device acquisition, which translates to processing less data in the ML bound, become prominent. To outweigh this effect, c_2 must increase by orders of magnitude to reduce the system active time.

7) ML Performance in Optimization Objective: Finally, we characterize the ML model training importance c_3 by its effect on the devices mini-batch size in Fig. 13 for $T^{\mathsf{ML}} = 1000$s. With increasing c_3, the importance of the model performance increases, and the solution prioritizes the third term in the objective by increasing the mini-batch size at the cost-efficient devices: we see that $n = 5$ begins processing more data, and, as c_3 increases, other devices also begin processing more data based on a combination of data offloading and processing cost.

VI. Conclusion

We introduced dynamic distributed learning via PSL. PSL extends federated learning through network, heterogeneity, and proximity. Also, it considers a cooperative distributed learning paradigm via incorporating *data dispersion* and *model/gradient dispersion with local condensation* across the devices to battle the innate shortcomings of federated learning. PSL further considers a realistic scenario in which the model training rounds are conducted with *idle times* in between, during which the data evolves across the devices. We modeled the processes conducted during PSL, introduced a new definition for concept/model drift, obtained the convergence characteristics of PSL, and formulated the network-aware PSL problem. We then proposed a general optimization solver to solve the problem with convergence guarantee. Multiple future work directions are also discussed in the paper.

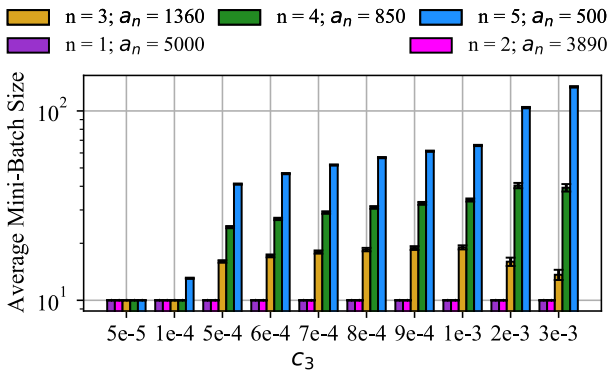

Fig. 13. Average mini-batch size of the devices for various values of ML model performance importance c_3 in the objective function of $\mathcal{P}$.

References

[1] C. Jiang, H. Zhang, Y. Ren, Z. Han, K. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Commun.*, vol. 24, no. 2, pp. 98–105, Apr. 2017.

[2] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *Proc. IEEE*, vol. 107, no. 11, pp. 2204–2239, Nov. 2019.

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

[4] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Toward an intelligent edge: Wireless communication meets machine learning," *IEEE Commun. Mag.*, vol. 58, no. 1, pp. 19–25, Jan. 2020.

[5] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," 2019, *arXiv:1909.07972*.

[6] 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 Conf. Comput. Commun.*, Apr. 2019, pp. 1387–1395.

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

[8] G. Zhu, Y. Wang, and K. Huang, "Broadband analog aggregation for low-latency federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 491–506, Jan. 2020.

[9] M. M. Amiri and D. Gündüz, "Federated learning over wireless fading channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3546–3557, May 2020.

[10] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, "Federated learning with quantization constraints," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 8851–8855.

[11] C. Renggli, S. Ashkboos, M. Aghagolzadeh, D. Alistarh, and T. Hoefler, "SparCML: High-performance sparse communication for machine learning," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2019, pp. 1–15.

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

[13] S. Dhakal, S. Prakash, Y. Yona, S. Talwar, and N. Himayat, "Coded federated learning," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2019, pp. 1–6.

[14] S. Wang, Y. Ruan, Y. Tu, S. Wagle, C. G. Brinton, and C. Joe-Wong, "Network-aware optimization of distributed learning for fog computing," *IEEE/ACM Trans. Netw.*, vol. 29, no. 5, pp. 2019–2032, Oct. 2021.

[15] V. Aggarwal, W. Wang, B. Eriksson, Y. Sun, and W. Wang, "Wide compression: Tensor ring nets," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9329–9338.
[16] S. Ji, W. Jiang, A. Walid, and X. Li, "Dynamic sampling and selective masking for communication-efficient federated learning," 2020, *arXiv:2003.09603*.
[17] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.
[18] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive IoT networks," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4641–4654, May 2020.
[19] H. Xing, O. Simeone, and S. Bi, "Decentralized federated learning via SGD over wireless D2D networks," in *Proc. IEEE 21st Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, May 2020, pp. 1–5.
[20] S. Hosseinalipour et al., "Multi-stage hybrid federated learning over large-scale D2D-enabled fog networks," *IEEE/ACM Trans. Netw.*, vol. 30, no. 4, pp. 1569–1584, Aug. 2022.
[21] F. P. Lin, S. Hosseinalipour, S. S. Azam, C. G. Brinton, and N. Michelusi, "Semi-decentralized federated learning with cooperative D2D local model aggregations," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3851–3869, Dec. 2021.
[22] M. Bertran et al., "Adversarially learned representations for information obfuscation and inference," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 614–623.
[23] S. S. Azam, T. Kim, S. Hosseinalipour, C. Joe-Wong, S. Bagchi, and C. Brinton, "Can we generalize and distribute private representation learning?" *Proc. 25th Int. Conf. Artif. Intell. Stat. (AISTATS)*, 2022, pp. 1–21.
[24] Z. Ji, L. Chen, N. Zhao, Y. Chen, G. Wei, and F. R. Yu, "Computation offloading for edge-assisted federated learning," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9330–9344, Sep. 2021.
[25] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*.
[26] S. Hosseinalipour, C. G. Brinton, V. Aggarwal, H. Dai, and M. Chiang, "From federated to fog learning: Distributed machine learning over heterogeneous wireless networks," *IEEE Commun. Mag.*, vol. 58, no. 12, pp. 41–47, Dec. 2020.
[27] S. Wang, M. Lee, S. Hosseinalipour, R. Morabito, M. Chiang, and C. G. Brinton, "Device sampling for heterogeneous federated learning: Theory, algorithms, and implementation," in *Proc. IEEE Conf. Comput. Commun.*, May 2021, pp. 1–10.
[28] M. N. Tehrani, M. Uysal, and H. Yanikomeroglu, "Device-to-device communication in 5G cellular networks: Challenges, solutions, and future directions," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 86–92, May 2014.
[29] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Wireless communications for collaborative federated learning," *IEEE Commun. Mag.*, vol. 58, no. 12, pp. 48–54, Dec. 2020.
[30] M. Abolhasan, T. Wysocki, and E. Dutkiewicz, "A review of routing protocols for mobile ad hoc networks," *Ad Hoc Netw.*, vol. 2, no. 1, pp. 1–22, Jan. 2004.
[31] S. Zeadally, R. Hunt, Y.-S. Chen, A. Irwin, and A. Hassan, "Vehicular ad hoc networks (VANETS): Status, results, and challenges," *Telecommun. Syst.*, vol. 50, no. 4, pp. 217–241, Aug. 2012.
[32] İ. Bekmezci, O. K. Sahingoz, and Ş. Temel, "Flying ad-hoc networks (FANETs): A survey," *Ad Hoc Netw.*, vol. 11, no. 3, pp. 1254–1270, May 2013.
[33] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surveys*, vol. 52, no. 1, pp. 1–38, Jan. 2020.
[34] D. Netburn. (Dec. 28, 2020). *From COVID to Curbside, 2020 Changed Our Vocabulary Too*. [Online]. Available: https://www.latimes.com/science/story/2020-12-28/from-covid-to-curbside-2020-changed-our-vocabulary-too
[35] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 53–65, Jan. 2018.
[36] S. L. Lohr, *Sampling: Design and Analysis: Design And Analysis*. Boca Raton, FL, USA: CRC Press, 2019.
[37] P. Zhao and T. Zhang, "Accelerating minibatch stochastic gradient descent using stratified sampling," 2014, *arXiv:1405.3080*.
[38] D. Bienstock, G. Muñoz, and S. Pokutta, "Principled deep neural network training through linear programming," 2018, *arXiv:1810.03218*.
[39] E. Rizk, S. Vlaski, and A. H. Sayed, "Dynamic federated learning," in *Proc. IEEE 21st Int. WRKSH Signal Process. Adv. Wireless Commun. (SPAWC)*, May 2020, pp. 1–5.
[40] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 33, Dec. 2020, pp. 7611–7623.
[41] Y. Chen, N. Zhao, Z. Ding, and M. Alouini, "Multiple UAVs as relays: Multi-hop single link versus multiple dual-hop links," *IEEE Trans. Wireless Commun.*, vol. 17, no. 9, pp. 6348–6359, Sep. 2018.
[42] J. Zhang, L. Yang, L. Hanzo, and H. Gharavi, "Advances in cooperative single-carrier FDMA communications: Beyond LTE-advanced," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 730–756, 2nd Quart., 2015.
[43] C. T. Dinh et al., "Federated learning over wireless networks: Convergence analysis and resource allocation," 2019, *arXiv:1910.13067*.
[44] Y. Ruan, X. Zhang, S.-C. Liang, and C. Joe-Wong, "Towards flexible device participation in federated learning," 2020, *arXiv:2006.06954*.
[45] M. Chiang, "Geometric programming for communication systems," *Found. Trends Commun. Inf. Theory*, vol. 2, nos. 1–2, pp. 1–154, Aug. 2005.
[46] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *J. Mach. Learn. Res.*, vol. 17, no. 83, pp. 1–5, Apr. 2016.
[47] G. Xu, "Global optimization of signomial geometric programming problems," *Eur. J. Oper. Res.*, vol. 233, no. 3, pp. 500–510, Mar. 2014.
[48] R. Duffin and E. Peterson, "Reversed geometric programs treated by harmonic means," *Indiana Univ. Math. J.*, vol. 22, no. 6, pp. 531–550, 1972.
[49] B. R. Marks and G. P. Wright, "A general inner approximation algorithm for nonconvex mathematical programs," *Oper. Res.*, vol. 26, no. 4, pp. 681–683, Aug. 1978.
[50] S. Wang, S. Hosseinalipour, M. Gorlatova, C. G. Brinton, and M. Chiang, "UAV-assisted online machine learning over multi-tiered networks: A hierarchical nested personalized federated learning approach," 2021, *arXiv:2106.15734*.
[51] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, 2005.
[52] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 8026–8037.
[53] L. Yan, C. Corinna, and C. J. Burges. *The MNIST Dataset of Handwritten Digits*. Accessed: Jun. 2023. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[54] H. Xiao, K. Rasul, and R. Vollgraf. *Fashion-MNIST*. Accessed: Jun. 2023. [Online]. Available: https://github.com/zalandoresearch/fashion-mnist
[55] J. Neyman, "On the two different aspects of the representative method: The method of stratified sampling and the method of purposive selection," in *Breakthroughs in Statistics*. Cham, Switzerland: Springer, 1992, pp. 123–150.
[56] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi, "A tutorial on geometric programming," *Optim. Eng.*, vol. 8, no. 1, p. 67, Mar. 2007.

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

Su Wang (Student Member, IEEE) received the B.Sc. degree in ECE from Purdue University in 2019, where he is currently pursuing the Ph.D. degree.

Nicolò Michelusi (Senior Member, IEEE) received the Ph.D. degree in EE from the University of Padua, Italy, in 2013. He is currently an Associate Professor of ECEE with Arizona State University.

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

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

David J. Love (Fellow, IEEE) is currently the Nick Trbovich Professor of Electrical and Computer Engineering at Purdue University.

Mung Chiang (Fellow, IEEE) is president and Roscoe H. George Distinguished Professor of ECE at Purdue University.