UAV-assisted Online Machine Learning over Multi-Tiered Networks: A Hierarchical Nested Personalized Federated Learning Approach

Su Wang, *Student Member, IEEE*, Seyyedali Hosseinalipour, *Member, IEEE*, Maria Gorlatova, *Member, IEEE*, Christopher G. Brinton, *Senior Member, IEEE*, and Mung Chiang, *Fellow, IEEE*

Abstract—We investigate training machine learning (ML) models across a set of geo-distributed, resource-constrained clusters of devices through unmanned aerial vehicles (UAV) swarms. The presence of time-varying data heterogeneity and computational resource inadequacy among device clusters motivate four key parts of our methodology: (i) *stratified UAV swarms* of leader, worker, and coordinator UAVs, (ii) *hierarchical nested personalized federated learning* (HN-PFL), a distributed ML framework for personalized model training across the worker-leader-core network hierarchy, (iii) *cooperative UAV resource pooling* to address computational inadequacy of devices by conducting model training among the UAV swarms, and (iv) *model/concept drift* to model time-varying data distributions. In doing so, we consider both *micro* (i.e., UAV-level) and *macro* (i.e., swarm-level) system design. At the micro-level, we propose network-aware HN-PFL, where we distributively orchestrate UAVs inside swarms to optimize energy consumption and ML model performance with performance guarantees. At the macro-level, we focus on swarm trajectory and learning duration design, which we formulate as a sequential decision making problem tackled via deep reinforcement learning. Our simulations demonstrate the improvements achieved by our methodology in terms of ML performance, network resource savings, and swarm trajectory efficiency.

Index Terms—UAVs, personalized federated learning, distributed model training, network optimization, model drift.

I. INTRODUCTION

Traditionally, machine learning (ML) has been managed centrally [1], [2] with the training performed at one location using all of the data. However, Internet of Things (IoT) use cases, e.g., autonomous driving, smart manufacturing, and object tracking, have highly distributed datasets that are challenging or sometimes impossible to centralize [3]. This has motivated the development of distributed ML, in particular federated learning (FL), techniques [4] to distribute model training across devices themselves. Though IoT devices are large in number, in many situations they are computationally limited (e.g., low cost wireless sensors) [5] leading to situations where data are highly distributed and cannot be locally processed.

S. Wang, C. Brinton, and M. Chiang are with Purdue University, IN, USA e-mail: {wang2506,cgb,chiang}@purdue.edu.

S. Hosseinalipour is with University at Buffalo (SUNY), NY, USA email: alipour@buffalo.edu

M. Gorlatova is with Duke University, Durham, NC, USA e-mail: maria.gorlatova@duke.edu.
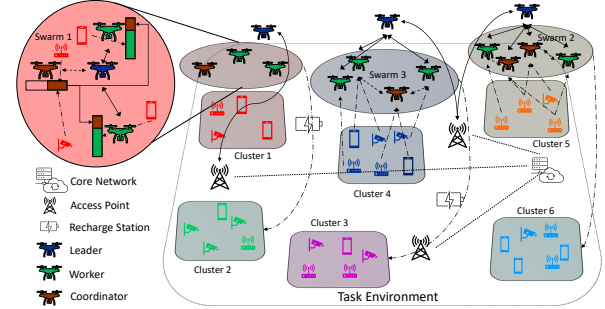
FIGURE 1: Schematic of UAV-enabled HN-PFL. The network consists of multiple IoT device clusters, stratified UAV swarms, access points, and recharging stations. UAV swarm leaders orchestrate intra-swarm ML training by workers, and the data collection/transfer at workers and coordinator UAVs. Periodically, leaders travel to access points for the core network for global model aggregation. As there are more clusters than swarms, the core network also determines the trajectories of the UAV swarms indicated by the dashed curved arrows.

Unmanned aerial vehicles (UAVs) have been recently incorporated into IoT networks to service applications such as surveillance, aerial base stations, smart agriculture, and search and rescue [6]. Existing work [3] have employed commercially-available UAVs with attached edge computing systems to act as a computational layer for receiving data from IoT devices and performing data processing tasks. Current literature is taking initial steps toward integrating such UAVs into FL [7]–[10], which will be particularly useful for geo-distributed IoT settings with sparse cellular connectivity. However, these works have mostly focused on implementing the classic FL architecture using UAVs, where either (i) the data is assumed to already be stored on the UAVs [7]–[9] or (ii) the UAVs simply replace cellular base stations as the model aggregators [10]. These works further have not accounted for the time-evolving nature of data distributions in many IoT settings caused by environmental shifts. To address this, we propose a paradigm, depicted in Fig. 1, that integrates UAVs and UAV swarms into the intelligent IoT ecosystem, where swarms must travel among the device clusters to efficiently collect measurements and conduct model training across time-varying data distributions.

Specifically, a UAV-assisted distributed ML paradigm must account for several factors. Online data variations across geo-distributed devices, as well as heterogeneity across the UAVs and devices in terms of data distributions and computation/communication resources, can each have a large impact on performance. On the infrastructure side, device-UAV-core

network interfacing and the locations of recharging stations become important, especially across a large geographic region. To address these challenges, the methodology we propose consists of four interrelated parts: (i) a new model for UAV swarms, called *stratified swarms*, suitable for distributed ML data collection and model training; (ii) *hierarchical nested personalized federated learning* (HN-PFL) to account for data heterogeneity in model updates; (iii) cooperative data processing across the UAVs via *resource pooling*; and (iv) *model/concept drift* tracking, which is tied to the model performance at devices and UAV movement patterns. We will develop and solve both UAV-level and swarm-level system optimizations: (a) HN-PFL model performance is optimized via efficient orchestration of the UAVs, taking into account the heterogeneity of the network elements, while (b) the swarm trajectories are optimized to account for anticipated model drifts across device clusters that will maximize training performance.

A. Motivations and Applications

The following scenarios will further motivate our system model for UAV-enabled online ML model training, where the UAVs perform both data collection and model training.

Community Service Systems via Amazon Sidewalk: Amazon proposes *Sidewalk* [11] to integrate community networks of IoT devices for household appliance diagnostics (e.g., garage system maintenance, pet finding, smart lighting) in the absence of consistent wi-fi connectivity by leveraging device-to-device (D2D) communications. However, efficient training of ML models on IoT devices and sharing them among distant neighborhoods faces four non-trivial challenges: (i) IoT devices may not be idle, plugged into power, or have direct access to a cellular base station (i.e., the core-network), preventing them from performing computationally intensive ML training, (ii) distant neighborhoods may not be reachable through D2D, (iii) neighborhoods' collected data may be time varying, and (iv) the neighborhoods may have extremely heterogeneous data distributions. We address the first two limitations by introducing UAV resource pooling and transferring model training vertically onto UAV swarms (e.g., the Amazon Prime Air delivery system), the third via model/concept drift at different neighborhoods, and the fourth through model personalization.

Distributed Surveillance in Smart Cities: Our proposed method has natural applications to smart city surveillance systems. For instance, multiple UAV swarms can be spread throughout a city, extracting data from sensors and cameras as well as those devices in rural areas without direct access to the closed-circuit cellular network to train an ML model [12]. Additionally, these swarms can travel to diverse neighborhoods (e.g., industrial parks vs. academic campuses) within a city, allowing for model recalibration in the presence of dynamic environments and improving the quality of personalized models through a global meta-learning approach that captures data commonalities across the city (see Sec. III).

Machine Learning on Wireless Sensor Networks: ML techniques have been adapted for wireless sensor networks (WSN) with respect to a wide variety of tasks, e.g., object targeting and event detection. Our model can be contextualized,

as an example, for integrated-WSN-UAV response systems, wherein wireless sensors collect data, e.g., water level or seismic energy, and offload it to UAVs, which then train ML models [13]. The UAV swarms can travel to distant and disconnected WSNs, e.g., across a beach/coast, to gather heterogeneous and time-varying data, e.g., day vs. night tidal measurements, and train ML models for each WSN cluster, thus integrating the system together.

B. Related Work

This paper contributes to both literature in distributed ML over wireless networks and ML for UAV networks. Below, we provide a summary of related works and highlight the main contributions of our methodology.

Distributed ML over wireless networks: Recent literature concerning ML by wireless networks has shifted towards federated learning [14], and is mostly focused on studying the convergence and behavior of federated learning over wireless networks [15]–[20]. Conventional federated learning assumes training a single ML model for all the engaged devices. However, upon having extreme data heterogeneity across the devices, training a single model may result in poor model performance for a portion of devices. This has motivated a new trend of research in ML that aims to train user-specific ML models, called personalized federated learning [21], in which meta-gradient updates are introduced to enhance the training efficiency. As compared to this literature, we develop and investigate a new distributed ML paradigm over wireless networks called *hierarchical nested personalized federated learning* (HN-PFL) inspired by meta-gradient updates, which is different than hierarchical FL architectures, e.g., [22], which rely on conventional gradient descent updates for local model training. Furthermore, we develop a new framework for network-aware HN-PFL over UAV-assisted wireless networks that considers model training under heterogeneity of resources in wireless networks. Our resulting optimization formulation balances the tradeoffs between ML model performance and network parameters such as data offloading, training batch sizes, and CPU cycles, and is also part of our contributions.

ML for UAV-assisted networks: Deep learning techniques have been utilized to enhance the efficiency of wireless networks [23]. In UAV-assisted networks, especially when the UAVs are deployed as aerial base stations, reinforcement learning (RL) has been utilized to carry out a variety of tasks, such as trajectory and power control for UAVs [24], and UAV server quality [25]. A key observation from the aforementioned existing literature is that UAV-assisted networks are difficult to model using traditional closed form methodologies and, as such, benefit from intelligent and autonomous management via RL. As compared to current literature, we introduce a new system model for UAV swarms and use a RL method to manage only the macro-level (i.e., swarm trajectories and temporal parameters of ML model training) of our methodology.

C. Outline and Summary of Contributions

Our contributions in this work can be summarized as follows:
- We introduce the framework of UAV-enabled online model training for a set of geo-distributed ground device clusters.

TABLE I: Summary of Key Notations for Devices, UAVs, Network Optimization, Machine Learning, and Swarm Trajectory Design

Device Cluster, UAV Swarm, and Network		**Data Processing/Offloading Optimization and Network Energy Consumption**			
$\mathcal{C}$	Set of all device clusters	$\rho_{i,j}(t)$	Data transfer ratio from device i to UAV j	$\varrho_{i,j}(t)$	Data transfer ratio from UAV i to j
$\mathcal{R}$	Set of recharging stations	$g_j(t)$	Adjustable CPU cycle frequency at UAV j	$\zeta_j^{\mathsf{G}}(k)$	Time used by UAV j to gather data
$\mathcal{A}$	Set of access points	$\zeta_j^{\mathsf{P}}(k)$	Time used by UAV j to process data	$E_j^{\mathsf{Ba}}(s)$	Starting battery of UAV j at the s-th sequence
$U(s)$	Number of active swarms at s-th sequence	M	Number of bits per datapoint	$\widetilde{M}$	Number of bits used for model parameters
$\mathcal{U}(s)$	Set of active swarms at s-th sequence	**Machine Learning Notation**			
ℓ_u	Leader UAV of u-th swarm	F_j	Meta loss function at UAV j	f_j	ML loss over all data in $\mathcal{D}_j$
$\mathcal{W}_u$	Worker UAVs in the u-th swarm	$\widehat{f}_j$	ML loss over a datapoint in $\mathcal{D}_j$	$\mathbf{w}_j(t)$	Model parameters at UAV j
$\widehat{u}$	Non-leader UAVs in u-th swarm	$\alpha_{j,1}(t)$	Inner mini-batch ratio for UAV j	$\alpha_{j,2}(t)$	Outer mini-batch ratio for UAV j
$\widehat{\mathcal{W}}_u$	Coordinator UAVs in the u-th swarm	$\alpha_{j,3}(t)$	Hessian mini-batch ratio for UAV j	t_s	Starting time for the s-th sequence
$\widetilde{\mathcal{D}}_i(t)$	Dataset of i-th device	τ_s^{L}	Local aggregation period for s-th sequence	τ_s^{G}	Global aggregation period for s-th sequence
$\mathcal{D}_j(t)$	Dataset of j-th UAV	k	Local aggregation index	k'	Global aggregation index
$\widetilde{\mathcal{D}}_c(t)$	Union of datasets at cluster c	**Swarm Trajectory Optimization**			
$B_j^{\mathsf{D}}(t)$	Max dataset size for UAV j	$\mathcal{X}(s)$	Swarm positions at s-th sequence	$G_c(s)$	Online gradient as a result of model drift
T_s	Duration of the s-th sequence	$\mathcal{Z}(s)$	Network state encoding at s-th sequence	$\mathcal{H}(s)$	DRL agent action at the end of the s-th sequence
Λ_c	Model drift of cluster c	$V(s)$	Reward of the s-th sequence	Q_θ	Q-network for trajectory design

We propose *stratified UAV swarms*, which presume different roles for the UAVs: (i) a *leader* that manages UAVs within the swarm (e.g., adjusting the CPU cycles and mini-batch sizes), and determines data transfer configurations, (ii) *workers* that conduct ML model training through *resource pooling*, and (iii) *coordinators* that enable data relaying between the devices and the worker UAVs.

- We develop hierarchical nested personalized federated learning (HN-PFL), which exploits meta-gradient based learning across disconnected device clusters and yield personalized local models. Through the nesting of intra-swarm updates within inter-swarm aggregations, HN-PFL conducts ML model training across the *worker-leader-core* network hierarchy. We analytically characterize the convergence behavior of HN-PFL, which leads us to new convergence bounds for distributed ML.

- We integrate network characteristics into ML training/performance by formulating a joint energy and ML performance optimization problem, which aims to configure the data offloading among devices-UAVs and UAVs-UAVs, adjust the CPU cycles of the UAVs, and obtain the mini-batch sizes used at the worker UAVs. This formulation is among the first in literature to consider all these design variables together. We demonstrate that the problem belongs to the category of *complementary geometric programming* problems which are highly non-convex and NP-Hard. We then develop a distributed method, with performance guarantee, based on posynomial condensation to solve the problem for all UAV swarms in parallel.

- We formulate the problem of UAV swarm trajectory design, alongside of which we also optimize the learning duration of HN-PFL. In the formulation, we consider *online model training* under temporal data distribution variations, which is quantified via *model/concept drift*. We demonstrate that the problem solution is intractable and then cast the problem as a sequential decision making problem tackled via a deep reinforcement learning-based method.

II. System Model

In this section, we introduce the system components, which include IoT device clusters (Sec. II-A), UAV swarms, recharging stations, and access points (Sec. II-B). We also provide an overview of our entire methodology, including the macro- and micro-level design in Sec. II-C.

A. Device Clusters and Data Distributions

We consider a set of geo-distributed devices collected into into C clusters, denoted by $\mathcal{C} = \{\mathcal{C}_1, \cdots, \mathcal{C}_C\}$, based on geographic proximity. Hereafter, we will refer to an arbitrary cluster via c for brevity. At time instance $t \in \mathcal{T} \triangleq \{1, 2, ...\}$, we denote $\widetilde{\mathcal{D}}_i(t)$ as the set of datapoints device $i \in c$ has collected at that time. Each $x \in \widetilde{\mathcal{D}}_i(t)$ is a data sample containing model features and (possibly) a target variable.

Motivated by the real-world applications in Sec. I-A, wherein data distributions are expected to change temporally (e.g., at environmental sensors), we focus on *online/dynamic* model training and deployment, which is different from current literature that mainly consider static data at the devices (e.g., [15]–[18]). Our goal is to obtain and periodically update a personalized ML model for each cluster c to use for real-time inference. However, due to significant computation/communication constraints the devices may be unable to train high-dimensional ML models (e.g., surveillance cameras as described in Sec. I-A). We thus transpose the ML model training to UAV swarms via data offloading from the devices. We assume that each device $i \in c$ collects data in its buffer $\widetilde{\mathcal{B}}_i$ of finite size, and that each device is capable of transmitting data to nearby UAVs to perform model training. We consider $\widetilde{\mathcal{B}}_i$ as a double-ended queue (i.e., deque) so that, when filled, new data will displace the oldest.[1]

B. Swarms, Recharging Stations, and Access Points

To conduct efficient ML model training at the UAVs, we introduce stratified UAV swarms, where UAVs have different roles for data collection and data processing (Sec. II-B1). Since the battery-limited UAVs perform energy-intensive ML model training, we also integrate a recharging methodology via recharging stations in our model (Sec. II-B2). Also, to synchronize the ML model training and UAV orchestration, we consider a set of access points in the network (Sec. II-B3). In the following, we explain each of these components. A schematic of our model is depicted in Fig. 1.

1) UAV Swarms and Stratification: We consider a set of U UAV swarms $\mathcal{U} = \{\mathcal{U}_1, \cdots, \mathcal{U}_U\}$, and assume that $U < C$. We denote an arbitrary UAV swarm as u for brevity. We assume that each UAV swarm u is composed of UAVs with heterogeneous capabilities, e.g., from micro-drones weighing under 200grams that have data storage capabilities to medium-sized fixed/rotatory wing UAVs that have more advanced

---

[1]**Henceforth, we use calligraphic (e.g., $\widetilde{\mathcal{D}}_i(t)$) to denote a set, and non-calligraphic (e.g., $\widetilde{D}_i(t)$) to denote its cardinality.**
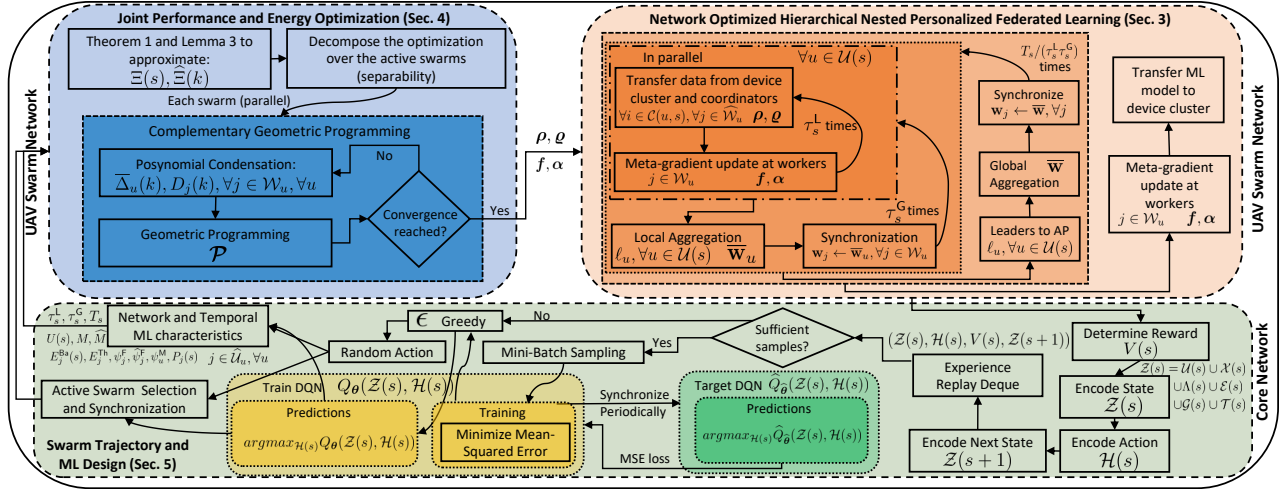
FIGURE 2: Our methodology consists of three components - the HN-PFL architecture (Sec. III), the data transfer and processing optimization (Sec. IV), and the swarm trajectory design (Sec. V). Our three components effectively reduce the problem into two layers: a *micro*-level wherein ML model training is optimized and subsequently performed via HN-PFL and a *macro*-level wherein the optimality of the macro-system (i.e. the network-wide consideration of all separate device clusters) is achieved. We additionally emphasize that a sole reinforcement learning based architecture would struggle greatly from the large and dynamic state-space of energy-efficient swarm trajectory design, HN-PFL performance and energy maximization, coordinator UAV transmissions, online model drifts, and meta-training convergence determination.

computational capabilities [26]. Subsequently, to maximize the performance of each swarm, we propose a new *swarm stratification* model, which compared to current literature on UAV trajectory design [27] is tailored specifically for ML training tasks. In our swarm stratification, there are three types of UAVs: (i) a leader ℓ_u of each swarm u, with the set of leaders across swarms denoted by $\mathcal{L} = \cup_{u \in \mathcal{U}} \ell_u$, (ii) a set of workers $\mathcal{W}_u$ in swarm u, with the set across swarms denoted by $\mathcal{W} = \cup_{u \in \mathcal{U}} \mathcal{W}_u$, and (iii) a set of coordinators $\widehat{\mathcal{W}}_u$ in u, gathered via the set $\widehat{\mathcal{W}} = \cup_{u \in \mathcal{U}} \widehat{\mathcal{W}}_u$. For convenience, we refer to the workers and coordinators together as $\widehat{u} = \mathcal{W}_u \cup \widehat{\mathcal{W}}_u = u \backslash \{\ell_u\}$. UAVs in $\widehat{u}$ collect data from nearby IoT devices, with the workers $\mathcal{W}_u$ conducting model training based on their gathered data, and the coordinators $\widehat{\mathcal{W}}_u$ relaying data to other UAVs in $\widehat{u}$ and building a data profile of the device cluster under visit.

UAV swarms transfer updated models to device clusters at the end of training sequences, and the devices employ these ML models to carry out inference tasks. At the beginning of each training sequence, i.e., when swarms arrive at clusters, each UAV swarm will have been synchronized through the core network with the same global ML model. When swarm u arrives at some device cluster c, the leader ℓ_u scatters the UAVs $\widehat{u}$ throughout predetermined locations in c. The exact positioning of the UAVs is not the focus of this work and can be computed *a priori* at the core network [28]. Since there are fewer UAV swarms $\mathcal{U}$ than device clusters $\mathcal{C}$, we will present a methodology to determine UAV swarm trajectories so that all device clusters can be supported in Sec V. Here, we focus on defining the interactions between a single UAV swarm and device cluster. All UAVs in $\widehat{u}$ gather data from devices $i \in c$ through a "wake up and sleep" paradigm, where the UAV notifies nearby IoT devices and prompts them to upload data. We denote $\mathcal{D}_j(t)$ as the dataset UAV $j \in \widehat{u}$, with buffer size $B_j^{\mathsf{D}}(t)$, obtains at time t. Workers $\mathcal{W}_u$ form a pool of computational resources above the device cluster and use their gathered data for cooperative ML training, in which they

engage in periodic communication with the leader ℓ_u regarding their training results. On the other hand, coordinators $\widehat{\mathcal{W}}_u$ act as *aerial data caches* to relay data to other UAVs in $\widehat{u}$. This arrangement allows more energy-efficient data transfer from IoT devices to worker UAVs via coordinator relaying. We formalize $\mathcal{D}_j(t)$, $\forall j$, through our optimization in Sec. IV-A.

2) Recharging Stations: We consider a set of geo-distributed recharging stations $\mathcal{R}$ in the network. We assume that when any UAV $j \in u$ reaches a minimum battery threshold, the entire swarm u must travel to a recharging station $r \in \mathcal{R}$.

3) Access Points (AP): We consider a set of gateway APs $\mathcal{A}$, which can be interpreted for example as cellular base stations. All APs $a \in \mathcal{A}$ are connected through the core network. The APs are used by the leader UAVs $\mathcal{L}$ to communicate with the core network, which determines swarm trajectories and synchronizes ML training among the swarms.

Deployments of this architecture in practice will require three types of signaling: (i) UAV-UAV communication, i.e., among leader, coordinator, and worker UAVs; (ii) device-UAV communication, i.e., data transfers from devices and requests from coordinator/worker UAVs; and (iii) UAV-AP interactions, i.e., for global model aggregations among swarm leaders. We detail potential approaches to these components in Appendix B.

C. Components of Our Methodology and Roadmap

We break down our methodology for conducting geo-distributed ML using UAVs into three parts: (i) ML model training through the worker-leader-core network hierarchy, (ii) efficient orchestration of the UAVs inside each swarm, and (iii) energy and performance-driven design of the UAV swarm trajectories across device clusters. For (i), we develop hierarchical nested personalized federated learning (HN-PFL) and subsequently derive its performance through theoretical bounds in Sec. III. For (ii), we formulate a network-aware ML performance and swarm-wide energy consumption optimization problem, and then develop a distributed solver in Sec. IV. For (iii), we capture the temporal variation of data across
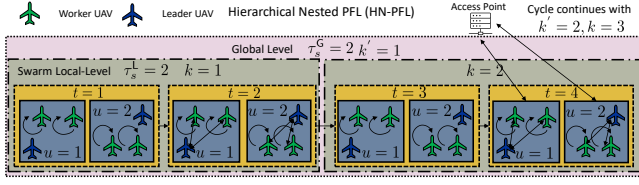
This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3216326

5

FIGURE 3: Behavior of workers, leaders, and APs in `HN-PFL`. Workers perform local meta-updates, and leaders aggregate their workers' model parameters with period τ_s^L. The core network synchronizes leader parameters every τ_s^G local aggregations.

the clusters and obtain the sequence of cluster visits for the UAV swarms (i.e., swarm trajectories) via integrating a deep reinforcement learning architecture in Sec. V. The interactions between these three components and an overview of their functionality is depicted in Fig. 2. From a high-level, the swarm-wide optimization (blue block) determines `HN-PFL`'s meta settings, and the performance of `HN-PFL` (orange block) is used to determine subsequent cluster visits (green block). This process then repeats, cyclically. We explain the details of each block of Fig. 2 in Sec. III, IV, and V, respectively.

D. Metrics of Interest

The evaluation of our methodology requires several primary metrics: classification loss at the UAVs (Thm. 1), the mismatch between the model performance at the UAVs vs. at the device clusters (Prop. 1), and energy consumption (Sec. IV). We will further demonstrate that our learning method provides substantial improvements compared to baselines in terms of convergence rate (Sec. VI-A), network resource savings (Sec. VI-B), and trajectory design efficiency (Sec. VI-C).

III. HIERARCHICAL NESTED PERSONALIZED FEDERATED LEARNING (HN-PFL)

In this section, we develop our UAV-enabled methodology for personalized federated learning (PFL). We begin with the rationale (Sec. III-A), and then present the `HN-PFL` algorithm (Sec. III-B). Finally, we theoretically analyze the convergence of our distributed ML method (Sec. III-C).

A. Overview and Rationale

Conventional federated learning (FL) trains a single ML model suitable for all devices [29]. As devices may exhibit significant heterogeneity in their data distributions, training a global model used for all the devices may lead to poor overall performance. This has motivated personalized federated learning (PFL) [21], which trains device-specific ML models by leveraging the *commonality* across the devices' data. Conventional FL and PFL both assume a "star" learning topology, where workers/devices are connected to and able to communicate directly with a main server [4].

In our setting, the star topology assumed in FL/PFL applies poorly as the IoT device clusters are geo-distributed, and UAVs visiting the clusters may not have direct access to an AP. On the other hand, UAV-to-UAV communications within a swarm is comparatively low in resource consumption, which motivates *local model aggregations* inside the swarms enabled by the leader UAV. The leaders can then occasionally visit their nearest AP for global aggregation of their associated swarm ML

model parameters. By *nesting* intra-swarm (local) aggregations within inter-swarm (global) aggregations, we develop a new methodology to generalize the star topology in conventional FL/PFL to that of a *hierarchical* tree, called `HN-PFL` where the ML model training is segmented into two layers: (i) swarm-level between leaders and their constituents, and (ii) global-level between access points/core-network and swarm leaders. To the best of our knowledge, `HN-PFL` is the first hierarchical personalized federated learning architecture in literature.

B. HN-PFL Algorithm

`HN-PFL` breaks down the model training problem into two layers: (i) *workers-leaders*, in which the worker UAVs carry out the ML model training and the leader UAVs perform swarm-wide (local) aggregations; and (ii) *leaders-APs*, in which the leader UAVs engage in global aggregations.

Since our problem requires the swarms $\mathcal{U}$ to travel between IoT device clusters, `HN-PFL` carries out the model learning through consecutive *training sequences*. Each training sequence starts when all active UAV swarms (i.e., non-recharging swarms) arrive at their designated device clusters, and concludes when the swarms finish the model training and begin travelling to their next destination. We denote the start of the s-th training sequence, $s = 1, 2, \cdots$, by $t_s \in \mathcal{T}$, its scheduled interval as $\mathcal{T}_s = \{t_s, \cdots, t_s + T_s - 1\}$, and the active swarms for the s-th sequence as $\mathcal{U}(s) \subseteq \mathcal{U}$. Defining the active swarms $\mathcal{U}(s)$ with respect to the training sequence encompasses the cases where a portion of UAV swarms recharge their batteries and thus are not engaged in ML model training during $\mathcal{T}_s$.

At each time $t \in \mathcal{T}_s$, each worker UAV $j \in \mathcal{W}_u$, $u \in \mathcal{U}(s)$ conducts a *local model update*. This consists of computing its next ML model parameter vector $\mathbf{w}_j(t+1)$ using a meta-gradient update [30] defined as:

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) - \eta_2 \nabla \widetilde{F}_j(\mathbf{w}_j(t)), \ t+1 \in \mathcal{T}_s, \quad (1)$$

where $\eta_2 > 0$ is the meta-update step-size, and $\nabla \widetilde{F}_j(\mathbf{w}_j(t))$ is the mini-batch approximation of the meta-gradient $\nabla F_j(\mathbf{w}_j(t))$. $\nabla F_j(\mathbf{w}_j(t))$ is the gradient of the meta-function F_j, defined as the loss of the gradient descent procedure:

$$F_j(\mathbf{w}_j(t)) = f_j\Big(\underbrace{\mathbf{w}_j(t) - \eta_1 \nabla f_j(\mathbf{w}_j(t))}_{(a)} \Big), \qquad (2)$$

where $\eta_1 > 0$ is the step size for gradient descent and f_j is the local loss function over the local dataset $\mathcal{D}_j(t)$ at UAV j:

$$f_j(\mathbf{w}_j(t)) = \frac{\sum_{x \in \mathcal{D}_j(t)} \widehat{f}(\mathbf{w}_j(t); x)}{D_j(t)}, \qquad (3)$$

and $\widehat{f}(\mathbf{w}_j(t); x)$ is the loss per datum x. This meta-gradient procedure effectively updates the parameters on the loss of the update rule, which, when connected to other meta-gradient results, yields an ML model that captures the structural data commonality across various data distributions. In particular, it results in a set of global parameters that can be better adapted/personalized to local data distributions with an additional gradient descent step, denoted by term (a) in (2). We approximate the gradient ∇F_j by mini-batch methods:

$$\nabla \widetilde{F}_j(\mathbf{w}_j(t)) = \nabla \widetilde{f}_j\left(\mathbf{w}_j(t) - \eta_1 \nabla \widetilde{f}_j(\mathbf{w}_j(t)|\mathcal{D}_{j,1}(t))\Big|\mathcal{D}_{j,2}(t)\right)$$

$$\cdot \left(\mathbf{I} - \eta_1 \nabla^2 \widetilde{f}_j(\mathbf{w}_j(t)|\mathcal{D}_{j,3}(t))\right), \qquad (4)$$

where $\widetilde{f}_j$ is the mini-batch loss defined similarly to (3) over a specific data batch[2], ∇^2 is the Hessian operator, and $\mathcal{D}_{j,1}(t)$, $\mathcal{D}_{j,2}(t)$, $\mathcal{D}_{j,3}(t)$ are three independent mini-batches sampled with replacement from $\mathcal{D}_j(t)$. We denote the mini-batch sampling ratios as $\alpha_{j,1}(t)$, $\alpha_{j,2}(t)$, and $\alpha_{j,3}(t) \in (0,1)$, i.e., $D_{j,i}(t) = \alpha_{j,i}(t)D_j(t)$, $i = 1, 2, 3$, and the total ratio of data at time t used for processing by worker $j \in \mathcal{W}_u$, $u \in \mathcal{U}(s)$ is denoted by $\alpha_j(t) = \alpha_{j,1}(t) + \alpha_{j,2}(t) + \alpha_{j,3}(t)$.

Remark 1. *The computational burden of the Hessian-based gradient descent in (4) can be alleviated by substituting it with approximate methods such as first order or hessian free model agnostic training [21], [30], which have been shown to attain a similar performance to that of exact Hessian computation.*

HN-PFL performs a series of local and global aggregations during each training sequence $\mathcal{T}_s$. We conduct local aggregations with period of τ_s^{L} (i.e., τ_s^{L} local meta-gradient updates prior to each local aggregation), and global aggregations with period of $\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}$, defined such that $T_s \triangleq \tau_s^{\mathsf{L}}K_s^{\mathsf{L}} \equiv \tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}K_s^{\mathsf{G}}$, where K_s^{L} is the total number of local aggregations, and K_s^{G} is the total number of global aggregations conducted in $\mathcal{T}_s$. Using k for the local aggregation index, $t_s^{\mathsf{L}}(k) = t_s + k\tau_s^{\mathsf{L}}$ is the time of the k-th local aggregation in $\mathcal{T}_s$. Using k' as the global aggregation index, $t_s^{\mathsf{G}}(k') = t_s + k'\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}$ will denote the time of the k'-th global aggregation in $\mathcal{T}_s$.

Within a swarm, workers' datasets will be controlled through a device-to-UAV offloading optimization procedure in Sec. III. Therefore, the swarm averaged ML model should be weighted towards those worker UAVs with the most processed data, as they are likely to have better trained ML models. So, at the k-th *local aggregation* when $t = t_s^{\mathsf{L}}(k)$, HN-PFL performs a weighted average at each leader ℓ_u, $\forall u$:

$$\overline{\mathbf{w}}_u(t) = \frac{\sum_{j \in \mathcal{W}_u} \mathbf{w}_j(t) \sum_{t'=t_s^{\mathsf{L}}(k-1)+1}^{t} \alpha_j(t')D_j(t')}{\sum_{j \in \mathcal{W}_u} \sum_{t'=t_s^{\mathsf{L}}(k-1)+1}^{t} \alpha_j(t')D_j(t')}. \qquad (5)$$

Leader ℓ_u then broadcasts $\overline{\mathbf{w}}_u(t_s^{\mathsf{L}}(k))$ to all worker UAVs $\mathcal{W}_u$. This completes our *swarm-wide aggregation*, and we define the swarm-wide/local meta-function $\overline{F}_u$ for $t = t_s^{\mathsf{L}}(k)$ as follows:

$$\overline{F}_u(\overline{\mathbf{w}}_u(t)) = \frac{\sum_{j \in \mathcal{W}_u} F_j(\overline{\mathbf{w}}_u(t)) \sum_{t'=t_s^{\mathsf{L}}(k-1)+1}^{t} \alpha_j(t')D_j(t')}{\sum_{j \in \mathcal{W}_u} \sum_{t'=t_s^{\mathsf{L}}(k-1)+1}^{t} \alpha_j(t')D_j(t')}. \qquad (6)$$

After τ_s^{L} swarm-wide aggregations at the active swarms, the leaders ℓ_u, $u \in \mathcal{U}(s)$, travel to their nearest AP, and transmit their swarm-wide parameters to the core network. At the k'-th *global aggregation*, when $t = t_s^{\mathsf{G}}(k')$, the core network determines the global model parameters as:

$$\overline{\mathbf{w}}(t) = \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \overline{\mathbf{w}}_u(t), \qquad (7)$$

with global meta-function $\overline{F}$ for $t = t_s^{\mathsf{G}}(k')$ as:

$$\overline{F}(\overline{\mathbf{w}}(t)) = \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \overline{F}_u(\overline{\mathbf{w}}(t)). \qquad (8)$$

[2]The second argument in $\widetilde{f}_j(.|.)$ denotes the data batch used to compute the respective function.

Note that (7)&(8) employ an unweighted aggregation across swarms, as opposed to the weighted aggregation (5)&(6) within swarms. This is due to the fact that the distributions of processed data will be non-i.i.d. across swarms. (7)&(8) allow the determination of a global model that uses the data across all the active clusters without bias towards any specific cluster and data distribution. A global model formed via unbiased meta-function aggregation seeks to capture the structural data commonalities across different UAV swarms, which allows both efficient model personalization and adaptability to online ML environments at device clusters. The access points $\mathcal{A}$ then broadcast $\overline{\mathbf{w}}(t_s^{\mathsf{G}}(k'))$ to the leaders, which return to their swarms and synchronize worker parameters.

At the final global aggregation of training sequence s, i.e., $t = t_s + T_s - 1$, swarms in $\mathcal{U}(s)$ *personalize* the global model to their respective visiting device cluster data distribution by performing a *single* stochastic gradient update (accounting for term (a) in (2)) followed by a swarm-wide aggregation as in (5), and then swarm leaders transfer the resulting ML model to the devices. Next, active swarms from both s and $s + 1$ initialize the next sequence. For each $u \in \mathcal{U}(s) \cup \mathcal{U}(s+1)$, leader ℓ_u will receive (i) final global parameters $\overline{\mathbf{w}}(t_s + T_s - 1)$, and (ii) trajectory decisions from the APs. Swarms then travel to their next cluster (for swarms $u \in \mathcal{U}(s) \setminus \mathcal{U}(s+1)$, this would be a recharging station). Upon arrival, swarm leaders l_u, $u \in \mathcal{U}(s+1)$, will begin training sequence $s + 1$ by synchronizing workers with the latest global parameters. Our developed HN-PFL algorithm is summarized in Fig. 3.

We assume that data distributions at the IoT devices are stationary during training sequences, but vary between training sequences. Distribution changes at clusters may lead to worse ML performance (i.e., a weaker ML model) over time, which incentivizes swarms to re-visit and re-calibrate their ML models. We model online data distributions at clusters by introducing heterogeneous *model/concept drift*, which primarily affects the UAV trajectories (see Sec. V-A). We next obtain the convergence bound of HN-PFL, which will be employed in the optimization formulation in Sec. IV.

C. Convergence Analysis of HN-PFL

In the following, we derive the performance bound of HN-PFL for non-convex loss functions. To this end, in addition to the meta functions defined in (6) and (8), we define the swarm-average loss function $\overline{f}_u(\mathbf{w})$ at $t \in \mathcal{T}_s$ as:

$$\overline{f}_u(\mathbf{w}(t)) = \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k)}{\overline{\Delta}_u(k)} f_j(\mathbf{w}(t)), \qquad (9)$$

where f_j is defined as in (3), k is the most recent local aggregation index, $\Delta_j(k) = \sum_{t'=t_s^{\mathsf{L}}(k-1)+1}^{t} \alpha_j(t')D_j(t')$, and $\overline{\Delta}_u(k) = \sum_{j \in \mathcal{W}_u} \Delta_j(k)$. Also, we define the global average loss function $\overline{f}(\mathbf{w})$ at time $t \in \mathcal{T}_s$ as:

$$\overline{f}(\mathbf{w}(t)) = \frac{1}{U} \sum_{u \in \mathcal{U}} \overline{f}_u(\mathbf{w}(t)). \qquad (10)$$

While (5),(6),(9) are only realized by HN-PFL at a local aggregation, $t = t_s^{\mathsf{L}}(k)$, and (7),(8),(10) are only realized at a

global aggregation, $t = t_s^\mathsf{G}(k')$, defining them for each $t \in \mathcal{T}_s$ will be useful in our theoretical analysis.

Our convergence analysis employs some standard assumptions [31] on non-convex loss functions:

Assumption 1 (Loss function characteristics). *We make the following assumptions on f_j at worker $j \in \mathcal{W}_u$, $\forall u \in \mathcal{U}$:*
1) *f_j is bounded below: $f_j(\mathbf{w}) > -\infty, \forall \mathbf{w}$.*
2) *The gradient of f_j is μ_j^G-Lipschitz and bounded by B_j: (i) $\|\nabla f_j(\mathbf{w}) - \nabla f_j(\mathbf{w}')\| \leq \mu_j^\mathsf{G}\|\mathbf{w} - \mathbf{w}'\|$, $\forall \mathbf{w}, \mathbf{w}'$, (ii) $\|\nabla f_j(\mathbf{w})\| \leq B_j$.*
3) *f_j is twice continuously differentiable, and the Hessian of f_j is μ_j^H-Lipschitz continuous: $\|\nabla^2 f_j(\mathbf{w}) - \nabla^2 f_j(\mathbf{w}')\| \leq \mu_j^\mathsf{H}\|\mathbf{w} - \mathbf{w}'\|$, $\forall \mathbf{w}, \mathbf{w}'$.*
4) *Bounded local data variability: $\mathbb{E}[\|\nabla \widehat{f}(x, \mathbf{w}) - \nabla f_j(\mathbf{w})\|^2] \leq \sigma_j^\mathsf{G}$ and $\mathbb{E}[\|\nabla^2 \widehat{f}(x, \mathbf{w}) - \nabla^2 f_j(\mathbf{w})\|^2] \leq \sigma_j^\mathsf{H}$, $\forall \mathbf{w}, x$.*
5) *The variance of the gradient and Hessian of $f_j(\mathbf{w})$ are bounded: $\sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k)}{\overline{\Delta}_u(k)}\|\nabla f_j(\mathbf{w}) - \nabla \overline{f}_u(\mathbf{w})\|^2 \leq \gamma_u^\mathsf{G}$, $\sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k)}{\overline{\Delta}_u(k)}\|\nabla^2 f_j(\mathbf{w}) - \nabla^2 \overline{f}_u(\mathbf{w})\|^2 \leq \gamma_u^\mathsf{H}, \forall \mathbf{w}, k$.*
6) *The variance of the gradient and Hessian of $\overline{f}_u(\mathbf{w})$ are bounded: $\frac{1}{U}\sum_{u \in \mathcal{U}}\|\nabla \overline{f}_u(\mathbf{w}) - \nabla \overline{f}(\mathbf{w})\|^2 \leq \gamma^\mathsf{G}$, $\frac{1}{U}\sum_{u \in \mathcal{U}}\|\nabla^2 \overline{f}_u(\mathbf{w}) - \nabla^2 \overline{f}(\mathbf{w})\|^2 \leq \gamma^\mathsf{H}, \forall \mathbf{w}$.*

We also require an assumption to characterize the loss function behavior at the device clusters:

Assumption 2 (Device cluster loss function characteristics). *For device cluster c and a model parameter $\mathbf{w}$, let $f_c^\mathsf{D}(\mathbf{w}) = \sum_{x \in \widetilde{\mathcal{D}}_c(t)} \widehat{f}(x; \mathbf{w})/\widetilde{D}_c(t)$, where $\widetilde{\mathcal{D}}_c(t) \triangleq \cup_{i \in c}\widetilde{D}_i(t)$, denote the local loss at time t. We assume that (i) the gradient of the loss function is μ_c^G-Lipschitz continuous, (ii) the Hessian is μ_c^H-Lipschitz continuous, (iii) $\mathbb{E}[\|\nabla \widehat{f}(x, \mathbf{w}) - \nabla f_c^\mathsf{D}(\mathbf{w})\|^2] \leq \sigma_c^\mathsf{G}$ $\forall x, \mathbf{w}$, and (iv) $\mathbb{E}[\|\nabla^2 \widehat{f}(x, \mathbf{w}) - \nabla^2 f_c^\mathsf{D}(\mathbf{w})\|^2] \leq \sigma_c^\mathsf{H}$ $\forall x, \mathbf{w}$.*

The loss functions of many common ML models (e.g., neural networks with continuous activation functions [16]) will satisfy these assumptions. In the following analysis, we let $B = \max_j\{B_j\}$, $\mu^\mathsf{G} = \max_j\{\mu_j^\mathsf{G}\}$, and $\mu^\mathsf{H} = \max_j\{\mu_j^\mathsf{H}\}$.

Our main result in this section will be the convergence behavior of the global meta-function $\overline{F}$ in `HN-PFL` (Theorem 1). To obtain this, we first derive bounds on the expected error of the meta-gradient approximations at worker UAVs (Lemma 1), and on the meta-gradient variability between workers, leaders, and the core network (Lemma 2).

Lemma 1 (Mini-batch versus full batch meta-gradients). *During each training interval $\mathcal{T}_s$, the expected error of the mini-batch approximation $\nabla \widetilde{F}_j(\mathbf{w}(t))$ of the true meta-gradient $\nabla F_j(\mathbf{w}(t))$ at each worker UAV $j \in \mathcal{W}_u$, $\forall u \in \mathcal{U}(s)$ satisfies:*

$$\mathbb{E}[\|\nabla \widetilde{F}_j(\mathbf{w}(t)) - \nabla F_j(\mathbf{w}(t))\|^2] \leq \sigma_j^\mathsf{F}(t), \quad (11)$$

where

$$\sigma_j^\mathsf{F}(t) = \underbrace{\frac{3\eta_1^2\sigma_j^\mathsf{H}}{\alpha_{j,3}(t)D_j(t)}}_{(a)}\underbrace{\left[B^2 + \frac{\sigma_j^\mathsf{G}(\alpha_{j,1}(t) + (\mu^\mathsf{G}\eta_1)^2\alpha_{j,2}(t))}{\alpha_{j,1}(t)\alpha_{j,2}(t)D_j(t)}\right]}_{(b)}$$
$$+ \underbrace{\frac{12\sigma_j^\mathsf{G}(\alpha_{j,1}(t) + (\mu^\mathsf{G}\eta_1)^2\alpha_{j,2}(t))}{\alpha_{j,1}(t)\alpha_{j,2}(t)D_j(t)}}_{(c)}. \quad (12)$$

Proof. See Appendix C. ∎

We make a few observations from Lemma 1 regarding the mini-batch ratios $\alpha_{j,1}(t)$, $\alpha_{j,2}(t)$, and $\alpha_{j,3}(t)$. Intuitively, if any of these are 0, then the upper bound in (11) should diverge, which is what we observe in (12). Next, the groups $(a), (b)$, and (c) in (12) show that each mini-batch ratio has a unique impact on the bound. In particular, $\alpha_{j,3}(t)$ weighs the Hessian data variability σ_j^H in (a), while $\alpha_{j,1}(t)$ and $\alpha_{j,2}(t)$ scale the gradient data variability σ_j^G in (b) and (c). Due to the multiplicative effect of the Hessian and gradient in the meta-gradient computation (see (4)), the effects of the mini-batch ratios are coupled between (a) and (b), through which $\alpha_{j,3}(t)$ also weights the impact of σ_j^G. Hence, when we are faced with a limited budget for data processing in a UAV swarm, the mini-batch ratios must be allocated carefully, which we will address through our optimization problem in Sec. IV-A.

Lemma 2 (Intra- and inter-swarm meta-gradient variability). *During each training interval $\mathcal{T}_s$, the intra-swarm meta-gradient variability at each UAV swarm $u \in \mathcal{U}(s)$ obeys the following upper bound at local aggregation k:*

$$\sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k)}{\overline{\Delta}_u(k)}\left\|\nabla F_j(\mathbf{w}) - \nabla \overline{F}_u(\mathbf{w})\right\|^2 \leq \gamma_u^\mathsf{F}, \quad (13)$$

where $\gamma_u^\mathsf{F} = 3B^2\eta_1^2\gamma_u^\mathsf{H} + 192\gamma_u^\mathsf{G}$. Also, the variability of inter-swarm meta-gradients is upper bounded as follows:

$$\frac{1}{U(s)}\sum_{u \in \mathcal{U}(s)}\left\|\nabla \overline{F}_u(\mathbf{w}) - \nabla F(\mathbf{w})\right\|^2 \leq \gamma^\mathsf{F}, \quad (14)$$

where $\gamma^\mathsf{F} = 3B^2\eta_1^2\gamma^\mathsf{H} + 192\gamma^\mathsf{G}$.

Proof. See Appendix D. ∎

Using Lemmas 1 and 2, we can bound the variance of `HN-PFL` model parameters across worker UAVs after a given global aggregation, which is one of our key theoretical results. For brevity, all proofs are given as sketches, with the key steps emphasized. The full proofs are provided in the appendices.

Proposition 1 (Parameter variability across UAVs). *During interval $\mathcal{T}_s$, the variance in model parameters across the active worker UAVs in the network at the k'-th global aggregation, i.e., $t = t_s^\mathsf{G}(k')$, satisfies the following upper bound:*

$$\mathbb{E}\left[\left\|\frac{1}{U(s)}\sum_{u \in \mathcal{U}(s)}\sum_{j \in \mathcal{W}_u}\frac{\Delta_j(k'\tau_s^\mathsf{G})}{\overline{\Delta}_u(k'\tau_s^\mathsf{G})}\left(\mathbf{w}_j(t) - \overline{\mathbf{w}}(t)\right)\right\|^2\right] \leq \Upsilon(t), \quad (15)$$

where $\Upsilon(t)$ is defined in (16) with $\mu^\mathsf{F} \triangleq 4\mu^\mathsf{G} + \eta_1\mu^\mathsf{H}B$, and $\sigma_u^\mathsf{F}(t) = \sum_{j \in \mathcal{W}_u}\frac{\Delta_j(k'\tau_s^\mathsf{G})}{\overline{\Delta}_u(k'\tau_s^\mathsf{G})}\sigma_j^\mathsf{F}(t)$.

Proof. See Appendix E. ∎

Considering the two terms in (16), term (a) captures the impact of the local aggregation frequency τ_s^L while term (b) captures the *nested* impact of the local-global aggregation frequency $\tau_s^\mathsf{L}\tau_s^\mathsf{G}$. It can be seen that, to reduce the variance of ML models $\mathbf{w}_j$ across workers $j \in \mathcal{W}_u$ $\forall u \in \mathcal{U}(s)$, which is desirable as we will show in Theorem 1, the local or global aggregation periods $\tau_s^\mathsf{L}, \tau_s^\mathsf{G}$ need to be reduced, i.e., more frequent aggregations. Proposition 1 suggests that the improvement is exponential. However, global aggregations

$$\Upsilon(t) = \underbrace{\left(\frac{16\eta_2^2}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \sum_{y=1}^{\tau_s^{\mathsf{L}}} \sigma_j^{\mathsf{F}}(t-y) + \frac{24\eta_2^2}{U(s)} \sum_{u \in \mathcal{U}(s)} \gamma_u^{\mathsf{F}}\right) \frac{1 - (8 + 48(\eta_2\mu^{\mathsf{F}})^2)^{\tau_s^{\mathsf{L}}}}{1 - (8 + 48(\eta_2\mu^{\mathsf{F}})^2)}}_{(a)}$$
$$+ \underbrace{\left(\frac{16\eta_2^2}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{y=1}^{\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}} \sigma_u^{\mathsf{F}}(t-y) + 24\eta_2^2\gamma^{\mathsf{F}}\right) \frac{1 - (8 + 48(\eta_2\mu^{\mathsf{F}})^2)^{\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}}}{1 - (8 + 48(\eta_2\mu^{\mathsf{F}})^2)}}_{(b)} \quad (16)$$

consume more network resources than local aggregations in `HN-PFL` - they require bidirectional communications between worker and leader UAVs, and between leaders and APs during which non-leader UAVs continue to consume energy by remaining idle in the air. We balance this trade-off by obtaining the optimal UAV orchestration for a choice of τ_s^{L} and τ_s^{G} (Sec. IV), and then jointly optimizing τ_s^{L} and τ_s^{G} and UAV trajectories under optimal UAV orchestration (Sec. V).

Additionally, Proposition 1 captures the impact of non-i.i.d. data distributions on parameter variability. In particular, we see that $\Upsilon(t)$ is directly proportional to the intra-swarm meta-gradient variability γ_u^F, the inter-swarm variability γ^F, and the mini-batch approximation error (the $\sigma_j^F(t)$ and $\sigma_u^F(t)$ terms). For the same setting of system control parameters, each of these will increase with the level of data heterogeneity present across devices, both within and between clusters.

Finally, we apply Proposition 1 to obtain our main result, which characterizes the decreasing magnitude of the gradient of (8), i.e., the global meta-function in `HN-PFL`, over training sequences. Since we consider non-convex ML models, the main metric of interest for learning performance is the norm squared of the meta-gradient.

Theorem 1. *[Global meta-gradient over training sequences] For training sequence $\mathcal{T}_s$, if $\eta_2 < \frac{1}{6\mu^F}$, we have the following upper bound on the expected cumulative average magnitude of the global meta-gradient across the active UAV swarms:*

$$\frac{1}{T_s} \sum_{t=t_{s-1}}^{t_s - 1} \mathbb{E}\left[\left\|\nabla\overline{F}(\mathbf{w}(t))\right\|^2\right] \leq \Xi(s), \quad (17)$$

where $\Xi(s)$ is given in (18).

Proof. See Appendix F. ∎

Main takeaways. Theorem 1 yields a general bound on the average gradient for `HN-PFL` with time-varying mini-batch sizes. Smaller values of the bound are desired, as it indicates closeness to a stationary point. This bound quantifies how several parameters (some controllable and others a factor of the environment) affect training performance. Specifically, the bound in (18) is dependent on the mini-batch ratios and processed data sizes at the UAVs (embedded in Δ_j, and Δ_u), the initial performance of the ML model for the s-th training sequence (embedded in $F^{\mathsf{U}}(\mathbf{w}(t_{s-1}))$), the data variability (embedded in σ_j^{F}, σ_u^{F}, and Υ), the gradient/Hessian characteristics (embedded in μ^{F}, γ_u^{F}, and γ^{F}), the local/global aggregation periods (through the nested sums), and the inner and outer step-sizes (η_1, η_2).

Since `HN-PFL` conducts ML model training at the UAVs, the bound in Theorem 1 is based on the meta-functions defined at the UAVs. However, as the ML model is used by the devices and UAVs train on a subset of the cluster's total data, we need

to bound the difference between the ML model performance at the UAVs and the devices. Henceforth, since the data at UAVs is assumed to be constant within a local aggregation period k, we refer to $D_j(t) \; \forall j$ as $D_j(k)$.

Lemma 3 (Meta-gradient mismatch between clusters and swarms). *Let F_c^{D} denote the meta-function defined based on f_c^{D} in Assumption 2, $\mathcal{C}(s)$ denote the set of actively trained device clusters for training sequence s, and $\overline{F}^{\mathsf{D}}(\mathbf{w}(t)) = \frac{1}{|\mathcal{C}(s)|} \sum_{c \in \mathcal{C}(s)} F_c^{\mathsf{D}}(\mathbf{w}(t))$, $t \in \mathcal{T}_s$, be the average meta-function for all actively trained device clusters for sequence s for a given parameter $\mathbf{w}(t)$ at time t. The difference between the meta-gradient computed at the UAVs vs. those of their respective device clusters for local aggregation k is bounded by:*

$$\mathbb{E}\left[\left\|\nabla\overline{F}(\mathbf{w}(t)) - \nabla\overline{F}^{\mathsf{D}}(\mathbf{w}(t))\right\|^2\right] \leq \widehat{\Xi}(k), \; t = \tau_s^{\mathsf{L}}(k), \quad (19)$$

where $\widehat{\Xi}(k)$ is given by:

$$\widehat{\Xi}(k) \triangleq \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k)}{\overline{\Delta}_u(k)} \left[\frac{3\eta_1^2\sigma_{C(u,s)}^{\mathsf{H}}}{D_j(k)}\left[B^2\right.\right.$$
$$\left.+ \frac{\sigma_{C(u,s)}^{\mathsf{G}}(1 + \mu^2\eta_1^2)}{D_j(k)}\right] + \frac{12\sigma_{C(u,s)}^{\mathsf{G}}(1 + \mu^2\eta_1^2)}{D_j(k)}\Bigg], \quad (20)$$

where $C(u,s)$ denotes the cluster which UAV swarm u trains during s and $\sigma_{\cdot}^{\mathsf{H}}$ and $\sigma_{\cdot}^{\mathsf{G}}$ are defined in Assumption 2.

Proof. See Appendix G. ∎

The bound in Lemma 3 captures the effect of total data at the worker UAVs on the mismatch between the cluster and UAV swarm meta-gradients. In particular, as more data from the devices are transferred to the UAVs (increasing $D_j(k)$), the error in (20) due to sampling decreases.

IV. Data Processing Optimization for `HN-PFL`

Given our `HN-PFL` algorithm and its convergence behavior in Sec. III-B, we next aim to develop efficient ML model training for our system. To this end, we need to obtain the optimal orchestration of UAVs once they engage in model training (micro-level design) and the swarm trajectories (macro-level design). We break these down into two main components: (i) data transfer and processing configurations at UAVs during the training sequences (the blue block in Fig. 2) and (ii) learning sequence duration and UAV swarm trajectory/movement patterns in-between training sequences (the green block in Fig. 2). These two parts are intertwined, i.e., the model training performance at clusters affects the UAV trajectory design, and vice versa.

In this section, we address the first component, by formulating the UAV data processing and transfer optimization problem, and subsequently develop the second component

$$\Xi(s) \triangleq \frac{1}{\frac{\eta_2}{2} - 6\eta_2^2 \frac{\mu^{\mathsf{F}}}{2}} \left[\frac{F^{\mathsf{U}}(\mathbf{w}(t_{s-1})) - (F^{\mathsf{U}})^\star}{T_s} + \frac{1}{T_s} \sum_{k'=1}^{K_s^{\mathsf{G}}} \sum_{k=(k'-1)\tau_s^{\mathsf{G}}+1}^{\tau_s^{\mathsf{G}} k'} \sum_{t=(k-1)\tau_s^{\mathsf{L}}}^{\tau_s^{\mathsf{L}} k - 1} \left[\left(3\eta_2^2 \frac{\mu^{\mathsf{F}}}{2} + \eta_2 \right) \right. \right.$$

$$\left. \left. \times \left(\frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\Delta_u(k'\tau_s^{\mathsf{G}})} \sigma_j^{\mathsf{F}}(t) + (\mu^{\mathsf{F}})^2 \Upsilon(t_s^{\mathsf{G}}(k')) \right) + 3\eta_2^2 \mu^{\mathsf{F}} \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \gamma_u^{\mathsf{F}} \right] \right] \tag{18}$$

in Sec. V. Our formulation will account for the interplay between the overall energy cost of the system and the developed `HN-PFL` performance metrics from Sec. III via a control optimization of key network parameters (Sec. IV-A). Then, we show that the resulting optimization can be characterized as a complementary geometric program, and develop an iterative *distributed optimization* method for solving it (Sec. IV-B).

A. Data Processing/Offloading Configuration

1) Offloading and processing models: At each training sequence s, each active UAV swarm $u \in \mathcal{U}(s)$ is located above a device cluster denoted by $\mathcal{C}(u,s) \in \mathcal{C}$. Let $A(u,s) \in \mathcal{A}$ denote the nearest AP to UAV swarm u, which leader ℓ_u will periodically visit during the model training to conduct global model aggregation. Also, let ψ_j^{F} denote the energy consumed per unit time by worker/coordinator UAV $j \in \mathcal{W}_u$ for flying/hovering, and $\hat{\psi}_u^{\mathsf{F}}$ that of leader UAV ℓ_u. We denote the geographical distance between leader ℓ_u and its nearest AP as $d(\ell_u, A)$, and denote the movement energy consumption of the leader UAV per unit distance as $\psi_{\ell_u}^{\mathsf{M}}$.

Device-UAV and UAV-UAV data transfers are carried out at the beginning of local aggregation rounds, i.e., when $t = t_s^{\mathsf{L}}(k)$. The data received by the worker UAVs is used throughout the local aggregation period. Upon engaging in data transmission, each device $i \in \mathcal{C}(u,s)$ samples data points uniformly at random from its local buffer and transmits them to the UAVs. We let $\rho_{i,j}(k) \in [0,1]$ denote the fraction of datapoints in the local dataset $\widetilde{D}_i(k)$ of the device that is transmitted to worker/coordinator UAV $j \in \widehat{u}$. Coordinator UAVs act as data caches that facilitate multi-hop data relaying between the devices to worker UAVs used for ML model training. We let $\varrho_{h,j}(k) \in [0,1]$ denote the fraction of the local dataset at the coordinator UAV $h \in \widehat{\mathcal{W}}_u$ that is forwarded to another UAV $j \in \mathcal{W}_u$. Each worker UAV $j \in \mathcal{W}_u$ processes data with CPU frequency $g_j(k) \in [g_j^{\mathsf{min}}, g_j^{\mathsf{max}}]$.

For sequence s, we denote the transmit powers of device i, UAV j, and leader ℓ_u by $P_i(s)$, $P_j(s)$, and $P_{\ell_u}(s)$, respectively. Through transmissions, either data or model parameters are transferred. We denote the number of bits used to represent one data point as $\widetilde{M}$, and the number of bits used to represent the one model parameter vector as M.

UAV-to-UAV data transmissions are carried out through air-to-air (A2A) channels, which are considered to be line-of-sight (LoS). Device-to-UAV and leader UAV-to-AP transmissions are performed through ground-to-air (G2A) and air-to-ground (A2G) channels, respectively, which are a mixture of LoS and non-line-of-sight (NLoS) links. Denoting $d(a,b)$ as the geographical distance between two nodes a and b $(a, b \in \mathcal{U} \cup \mathcal{C})$, the path-loss model for the LoS link between two nodes a and b is then given by [32]:

$$L_{a,b}^{\mathsf{A2A}} = \eta^{\mathsf{LoS}} (\mu^{\mathsf{Tx}} d(a,b))^{\alpha^{\mathsf{PL}}}, \tag{21}$$

where $\eta^{\mathsf{LoS}} > 1$ denotes the excessive path loss factor for the LoS link, α^{PL} is the path-loss exponent, and $\mu^{\mathsf{Tx}} = 4\pi f^{\mathsf{Tx}} / c^{\mathsf{light}}$ with c^{light} denoting the speed of light and f^{Tx} denoting the carrier frequency. For the G2A/A2G channel, the probability of having an LoS link between two nodes a and b is given by [32], [33]: $P_{a,b}^{\mathsf{LoS}} = \left(1 + \psi^{\mathsf{Tx}} \exp(-\beta^{\mathsf{Tx}}[\theta_{ab} - \psi^{\mathsf{Tx}}])\right)^{-1}$, where ψ^{Tx} and β^{Tx} are constants depending on the carrier frequency and the conditions of the environment, and $\theta_{a,b}$ is the elevation angle between the respective nodes defined as: $\theta_{a,b} = \frac{180}{\pi} \times \sin^{-1}\left(\frac{\Delta h_{a,b}}{d(a,b)}\right)$, with $\Delta h_{a,b}$ denoting the difference in altitude between nodes a and b. The probability of NLoS link is given by $P_{a,b}^{\mathsf{NLoS}} = 1 - P_{a,b}^{\mathsf{LoS}}$. With this, the path-loss of an A2G/G2A link from node a to node b can be obtained as:

$$L_{a,b}^{\mathsf{A2G}} = (\mu^{\mathsf{Tx}} d(a,b))^{\alpha^{\mathsf{PL}}} [P_{a,b}^{\mathsf{LoS}} \times \eta^{\mathsf{LoS}} + P_{a,b}^{\mathsf{NLoS}} \times \eta^{\mathsf{NLoS}}], \tag{22}$$

where $\eta^{\mathsf{NLoS}} > \eta^{\mathsf{LoS}}$ denotes the excessive path loss factor for the NLoS link. Finally, the data-rate between two nodes a and b is given by:

$$R_{a,b} = \bar{B}_{a,b} \log_2(1 + \frac{P_a/L_{a,b}}{\sigma^2}), \; a,b \in \mathcal{U} \cup \mathcal{C}, \tag{23}$$

where $\bar{B}_{a,b}$ denotes the bandwidth, $\sigma^2 = N_0 \bar{B}_{a,b}$ denotes the noise power with N_0 as the noise spectral density, P_a is the transmit power of node a, and $L_{a,b}$ is the path-loss obtained through either (21) or (22). As the channel conditions may change over time, we incorporate the training sequence index into the data rate notation denoted by $R_{a,b}(s)$ to represent the data rate between two nodes a and b during the training sequence s. To focus our optimization formulation on the interplay between ML convergence and energy consumption, we simplify the contributions of specific modulation and signaling schemes by ignoring interference from simultaneous transmissions to the UAVs as they are stationary and can use orthogonal frequency bands [34] for communications.

We denote the time used for data gathering at UAV j as $\zeta_j^{\mathsf{G}}(k)$, and the time used for data processing as $\zeta_j^{\mathsf{P}}(k)$. We also define ζ^{Local} as the maximum allowable time for data gathering and local computation before each UAV transmits its parameters to the leader UAV for aggregation.

2) Joint energy and performance optimization: With the aforementioned models in hand, we formulate the following optimization problem for determining data offloading/processing configuration at training sequence s:

$$(\mathcal{P}) : \min_{\boldsymbol{\rho}, \boldsymbol{\varrho}, \boldsymbol{\alpha}, \boldsymbol{g}} (1-\theta) \underbrace{\left(\theta_1 \Xi(s) + \theta_2 \sum_{k=1}^{K_s^{\mathsf{L}}} \widehat{\Xi}(k) \right)}_{(a)}$$

$$+ \theta \underbrace{\sum_{k=1}^{K_s^{\mathsf{L}}} \sum_{u \in \mathcal{U}(s)} \left[\sum_{j \in \widehat{\mathcal{W}}_u} E_j^{\mathsf{Tx,U}}(k) + \sum_{j \in \mathcal{W}_u} E_j^{\mathsf{P}}(k) + \sum_{i \in \mathcal{C}(u,s)} E_i^{\mathsf{Tx,C}}(k) \right]}_{(b)} \tag{24}$$

s.t.

$$E_j^{\mathsf{P}}(k) = \tau_s^{\mathsf{L}} \frac{a_j c_j}{2} (\alpha_j(k)) D_j(k) g_j^2(k), j \in \mathcal{W}_u, \forall u \quad (25)$$

$$E_i^{\mathsf{Tx,C}}(k) = \sum_{j \in \widehat{u}} \frac{\rho_{i,j}(k)\widetilde{D}_i(k)\widetilde{M}}{R_{i,j}(s)} P_i(s), i \in \mathcal{C}(u,s), \forall u \quad (26)$$

$$E_j^{\mathsf{Tx,U}}(k) = \sum_{h \in \mathcal{W}_u} \frac{\varrho_{j,h}(k)D_j(k)\widetilde{M}}{R_{j,h}(s)} P_j(s), j \in \widehat{\mathcal{W}}_u, \forall u \quad (27)$$

$$E_j^{\mathsf{Tx,W}} = K_s^{\mathsf{L}} P_j(s) M / R_{j,\ell_u}(s), j \in \mathcal{W}_u, \forall u \quad (28)$$

$$E_{\ell_u}^{\mathsf{Tx,L}} = K_s^{\mathsf{L}} P_{\ell_u}(s) \max_{j \in \mathcal{W}_u} \left\{ M / R_{\ell_u,j}(s) \right\}, \forall u \quad (29)$$

$$E_j^{\mathsf{F,U}} = T_s \psi_j^{\mathsf{F}}, j \in \widehat{u}, \forall \widehat{u} \quad (30)$$

$$E_{\ell_u}^{\mathsf{F,L}} = T_s \hat{\psi}_u^{\mathsf{F}} + 2K_s^{\mathsf{G}} \psi_u^{\mathsf{M}}(d(\ell_u, A)), \forall u \quad (31)$$

$$\sum_{k=1}^{K_s^{\mathsf{L}}} E_j^{\mathsf{P}}(k) + E_j^{\mathsf{Tx,W}} + E_j^{\mathsf{F,U}} \le E_j^{\mathsf{Ba}}(s) - E_j^{\mathsf{Th}}, j \in \mathcal{W}_u, \forall u \quad (32)$$

$$\sum_{k=1}^{K_s^{\mathsf{L}}} E_j^{\mathsf{Tx,U}}(k) + E_j^{\mathsf{F,U}} \le E_j^{\mathsf{Ba}}(s) - E_j^{\mathsf{Th}}, j \in \widehat{\mathcal{W}}_u, \forall u \quad (33)$$

$$E_{\ell_u}^{\mathsf{Tx,L}} + E_{\ell_u}^{\mathsf{F,L}} \le E_{\ell_u}^{\mathsf{Ba}}(s) - E_{\ell_u}^{\mathsf{Th}}, \forall u \quad (34)$$

$$\sum_{j \in \widehat{u}} \rho_{i,j}(k) \le 1, i \in \mathcal{C}(u,s), \forall u \quad (35)$$

$$\sum_{h \in \mathcal{W}_u} \varrho_{j,h}(k) \le 1, j \in \widehat{\mathcal{W}}_u, \forall u \quad (36)$$

$$D_j(k) = \sum_{i \in \mathcal{C}(u,s)} \rho_{i,j}(k)\widetilde{D}_i(k)$$
$$+ \sum_{h \in \widehat{\mathcal{W}}_u} \varrho_{h,j}(k) D_h(k) \le B_j^{\mathsf{D}}(k), j \in \mathcal{W}_u, \forall u \quad (37)$$

$$D_j(k) = \sum_{i \in \mathcal{C}(u,s)} \rho_{i,j}(k)\widetilde{D}_i(k) \le B_j^{\mathsf{D}}(k), j \in \widehat{\mathcal{W}}_u, \forall u \quad (38)$$

$$\zeta_j^{\mathsf{G}}(k) + \zeta_j^{\mathsf{P}}(k) \le \zeta^{\mathsf{Local}}, j \in u, \forall u \quad (39)$$

$$\zeta_j^{\mathsf{G}}(k) = \sum_{i \in \mathcal{C}(u,s)} \rho_{i,j}(k)\widetilde{D}_i(k)\widetilde{M}/R_{i,j}(s)$$
$$+ \sum_{h \in \widehat{\mathcal{W}}_u} \varrho_{h,j}(k) D_h(k)\widetilde{M}/R_{h,j}(s), j \in \widehat{u}, \forall \widehat{u} \quad (40)$$

$$\zeta_j^{\mathsf{P}}(k) = \tau_s^{\mathsf{L}} c_j \alpha_j(k) D_j(k)/g_j(k), j \in \mathcal{W}_u, \forall u \quad (41)$$

$$g_j^{\mathsf{min}} \le g_j(k) \le g_j^{\mathsf{max}}, j \in \mathcal{W}_u, \forall u \quad (42)$$

$$0 \le \rho_{i,j}(k) \le 1, i \in \mathcal{C}(u,s), j \in \widehat{u}, \forall u \quad (43)$$

$$0 \le \varrho_{h,j}(k) \le 1, h \in \widehat{\mathcal{W}}_u, j \in \mathcal{W}_u \quad (44)$$

$$\alpha_j(k) = \alpha_{j,1}(k) + \alpha_{j,2}(k) + \alpha_{j,3}(k), \; j \in \mathcal{W}_u, \forall u \quad (45)$$

$$0 < \alpha_{j,1}(k), \alpha_{j,2}(k), \alpha_{j,3}(k) \le 1, j \in \mathcal{W}_u, \forall u \quad (46)$$

Objective and variables. The objective function in (24) captures the tradeoff between the expected ML model performance (term (a)) and the energy consumption during the training sequence (term (b)). Term (a) encompasses both the ML performance at the active UAV swarms (through Ξ) and the mismatch to the ML performance at the device cluster (through $\widehat{\Xi}$), which we quantified in Theorem 1 and Lemma 3. We weigh the importance of the two terms in (a) with normalized positive coefficients θ_1 and θ_2. The coefficient $\theta \in [0,1]$ weighs the importance of the terms in the objective function. The

variables in the problem are the device-to-UAV data transfer configurations $\boldsymbol{\rho} = \{\rho_{i,j}\}_{i \in \mathcal{C}(u,s), j \in \widehat{u}, u \in \mathcal{U}(s)}$, the UAV-to-UAV data transfer configurations $\boldsymbol{\varrho} = \{\varrho_{h,j}\}_{h \in \widehat{\mathcal{W}}_u, j \in \mathcal{W}_u, u \in \mathcal{U}(s)}$, the mini-batch ratios $\boldsymbol{\alpha} = \{\alpha_{j,1}, \alpha_{j,2}, \alpha_{j,3}\}_{j \in \mathcal{W}_u, u \in \mathcal{U}(s)}$, and the UAV CPU frequency cycles $\boldsymbol{g} = \{g_j\}_{j \in \mathcal{W}_u, u \in \mathcal{U}(s)}$.

For the energy objective terms, $E_j^{\mathsf{P}}(k)$ in (25) denotes the energy consumption used for processing data during each meta-gradient update at worker UAV $j \in \mathcal{W}_u$, where a_j is the effective capacitance coefficient of UAV j's chipset [35], and c_j is the number of CPU cycles to process each datapoint. In (26), $E_i^{\mathsf{Tx,C}}(k)$ is the energy used for data transmission (Tx) from device i in cluster $\mathcal{C}(u,s)$ to the UAVs. In (27), we define $E_j^{\mathsf{Tx,U}}(k)$ as the energy used for data transmissions from coordinator UAV $j \in \widehat{\mathcal{W}}_u$ to other UAVs. In (28), $E_j^{\mathsf{Tx,W}}$ denotes the total energy consumed for ML model parameter transmission from worker UAV $j \in \mathcal{W}_u$ to its associated leader, which occurs K_s^{L} times during the training sequence. In (29), $E_{\ell_u}^{\mathsf{Tx,L}}$ captures the energy consumption used for parameter transmission through broadcasting from leader ℓ_u to worker UAVs $j \in \mathcal{W}_u$. In (30), $E_j^{\mathsf{F,U}}$ represents the total energy used for flying/hovering by UAV $j \in \widehat{u}$ for the s-th training sequence, which can be obtained using the result of [36] for fixed-wing UAVs, and the result of [28] for rotary wing UAVs. Similarly, in (31), $E_{\ell_u}^{\mathsf{F,L}}$ captures the energy consumption for flying by leaders ℓ_u, accounting for levitation and round-trips to the nearest APs at the global aggregation instances.

Constraints. Constraint (32) ensures that the total energy consumed by each worker UAV j for data processing, parameter transmission and flying is less than $E_j^{\mathsf{Ba}}(s) - E_j^{\mathsf{Th}}$, where $E_j^{\mathsf{Ba}}(s)$ represents the battery energy at j at the start of the s-th training sequence and E_j^{Th} encompasses both (i) surplus idle energy needed for extra hovering time caused by potential asynchronocity due to the heterogeneity of leader UAV to AP travel times, and (ii) the minimum energy threshold for j to reach the nearest recharging station after the conclusion of the training sequence. Constraint (33) imposed on the coordinator UAVs is similar to (32), except that coordinator UAVs only conduct data transmission while flying. Constraint (34) imposed on the leader UAVs guarantees that there is enough energy remaining after parameter broadcasting and flying to reach the nearest recharging station. Constraints (35) and (36) ensure that the total amount of data offloaded from each device $i \in \mathcal{C}(u,s)$ and each coordinator UAV $h \in \widehat{\mathcal{W}}_u$ is less than the size of the available data set. As a result of this offloading, (37) and (38) capture the total number of datapoints at the UAVs. Constraint (39) ensures that the accumulated time used for data gathering $\zeta_j^{\mathsf{G}}(k)$ and local data processing $\zeta_j^{\mathsf{P}}(k)$ at each worker UAV j during one local aggregation is lower than the maximum allowable time for parameter transfer to the corresponding leader UAV ζ^{Local}. These two quantities are in turn defined in (40) and (41). Finally, constraints (42)-(46) ensure the feasibility of the optimization variables.

Complementary geometric program. $\mathcal{P}$ is a non-convex problem, since products of the optimization variables are present in the objective. For instance, in the definition $E_j^{\mathsf{P}}(n)$ in (25), there are multiplications between all four variable types: α, g, ρ, and ϱ (the last two are encompassed in term D_j

according to (37)). In fact, this problem can be categorized as a *complementary geometric program* [37], a particular class of non-convex and NP-hard optimization problems. Based on this classification, we will develop a distributed, iterative approach to solve $\mathcal{P}$ where iterative approximation of the problem in the format of a geometric program (GP) is considered.

B. Distributed Algorithm for Data Processing/Offloading

To solve $\mathcal{P}$, we first make two key observations:

Observation 1: Complementary Geometric Programming. GP is a method for converting a non-convex optimization problem into convex form when the objective and constraints are composed of *monomials* and *posynomials*. We provide an overview of this for the interested reader in Appendix H. Although the objective in $\mathcal{P}$ is composed of multiplications between the variables, it does not follow the format of GP: the expressions of Ξ and $\widehat{\Xi}$ consist of terms that are in the format of ratio of two posynomials, which are not posynomials (e.g., $\frac{\Delta_j}{\Delta_u}$, Υ, and σ_j^{F}, each of which contain arithmetic relationships of ρ and ϱ in the denominators). Rather, $\mathcal{P}$ is a complementary GP, which cannot readily be translated to convex form [37]. We exploit a method based on *posynomial condensation* to approximate the ratios of two posynomials appearing in the expressions of Ξ and $\widehat{\Xi}$ (via approximating $\overline{\Delta}_u$ and D_j where needed) as the ratio between a posynomial and a monomial. Since the ratio of a posynomial and a monomial is a posynomial, we then can convert the approximations into geometric programs.

Observation 2: Potential of Distributed Implementation. In $\mathcal{P}$, all of the constraints are separable with respect to the UAV swarm index u, and the objective function (including the terms Ξ and $\hat{\Xi}$) can be written as a sum of separable functions with respect to the UAV swarm index. In practice, however, γ^{F} (defined in Lemma 2) and μ^{F} (defined in Proposition 1) cannot be locally and independently computed by UAV swarms $u \in \mathcal{U}(s)$, as it depends on the maximum over all worker UAVs in the network. We can approximate γ^{F} and μ^{F} by estimating it at the core network at the instance of global aggregations, and subsequently broadcast it to the leader UAVs though the APs. With the knowledge of γ^{F} and μ^{F} at the swarm leaders, the problem can then be decomposed among the local aggregation instances $k \in \{1, \cdots, K_s^{\mathsf{L}}\}$ and solved distributively by each swarm $u \in \mathcal{U}(s)$ at its leader.

Developing the solver: First, we must convert the ratio of posynomials in the objective of $\mathcal{P}$ into that of a posynomial and a monomial. To do so, we iteratively approximate the posynomial denominators, which are Δ_u and D_j in our case, using the arithmetic-geometric mean inequality (see Lemma 5 in Appendix H). The resulting approximations for iteration $m+1$ are given in (48) and (51), where m is the iteration index. Here, the solution to the problem at the m-th approximation iteration is denoted $\mathbf{x}[m] = \{\varrho[m], \rho[m], \alpha[m], g[m]\}$. $\overline{\Delta}_u(k; m)$, $\delta_{j,h,i,n}(k; m)$, and $\widetilde{\delta}_{i,j,n}(k; m)$, $n \in \{1,2,3\}$ in (48) and $D_j(k; m)$, $\lambda_{h,i,j}(k; m)$, and $\widetilde{\lambda}_{i,j}(k; m)$ in (51) are scalar values obtained by substituting the solution at iteration m (i.e., $\mathbf{x}[m]$) in the corresponding expressions in (47), (49), (50), and (52), respectively. It can be verified that the approximations in (48) and (51) are in fact the best local monomial approximations

---

Algorithm 1: Distributed network-aware optimization of `HN-PFL` under partial global knowledge

input : Convergence criterion between two consecutive iterations. Estimated Lipschitz continuity factors (i.e., μ^{G}, μ^{H}, and μ^{F}), estimated data variability (i.e., σ^{H} and σ^{G}), and estimated data quantities (i.e., $D_j(k)$) at devices, all of which are obtained based on historical data observations.

1   // *Procedure at the UAV swarms*
2   **for** $k' \in \{1, \ldots, K_s^{\mathsf{G}}\}$ **do**
3   // *Procedure at the core network*
4   **for** $\ell_u \in \mathcal{L}(s)$ **do**
5   // *Procedure conducted across the UAV swarms in parallel using their leader*
6   **for** $k \in \{(k'-1)\tau_s^{\mathsf{G}} + 1, \cdots, k'\tau_s^{\mathsf{G}}\}$ **do**
7   Initialize the iteration count $m = 0$.
8   Choose an initial feasible point $\mathbf{x}[0] = [\rho[0], \varrho[0], \alpha[0], g[0]]$ for $\widetilde{\mathcal{P}}_0$ at k.
9   **while** *convergence criterion is not met OR* $m = 0$ **do**
10   Obtain the scalar values for $\overline{\Delta}_u(k, m)$, $\widetilde{\delta}_{j,i,n}(k; m)$, $\delta_{j,h,i,n}(k; m)$, $D_j(k, m)$, and $\lambda_{i,j}(k; m)$, $\widetilde{\lambda}_{h,i,j}(k; m)$ by substituting $\mathbf{x}[m]$ in (47), (49), (50), and (52).
11   Use the above scalars to obtain the monomial approximations given in (48) and (51), and replace them in $\widetilde{\mathcal{P}}_m$.
12   Apply the logarithmic change of variables and take the log from the constraints of $\widetilde{\mathcal{P}}_m$ and convert it to a convex optimization problem (as in (93) in Appendix H).
13   Solve the resulting convex optimization problem to obtain $\mathbf{x}[m+1]$ (e.g., using CVX [38]), and set $m = m + 1$.
14   Choose the obtained point as $\mathbf{x}^\star = \mathbf{x}[m] = [\rho^\star, \varrho^\star, \alpha^\star, g^\star]$.
15   The leader broadcasts the solution $\mathbf{x}^\star$ among its constituent workers/coordinators/devices to start their respective data transfer procedures, and specifically for workers to tune their CPU cycles and mini-batch sizes.

---

to their respective posynomials near the fixed point $\mathbf{x}[m]$ in the sense of the first-order Taylor approximation.

We solve $\mathcal{P}$ in a distributed manner at each UAV swarm through sequentially applying the above approximations to obtain problem $\widetilde{\mathcal{P}}_m$ at iteration m. In $\widetilde{\mathcal{P}}_m$, each constraint is an inequality on posynomials, and the objective function is a sum of posynomials, admitting the GP format:

$$(\widetilde{\mathcal{P}}_m): \min_{\rho, \varrho, \alpha, g} (1-\theta)\widetilde{(a)}_m + \theta(b) \quad \text{s.t.} \quad (25) - (46),$$

where term $\widetilde{(a)}_m$ is obtained from term (a) in problem $\mathcal{P}$ by: (i) decomposing (a) into a sum of separable functions with respect to UAV swarm indexes; and then (ii) using the expressions in (48) and (51) for iteration m in (a). Note that term (b) defined in (24) is indexed by UAV swarm u, and the constraints of $\widetilde{\mathcal{P}}$ are with respect to UAV swarm.

The pseudocode of our resulting sequential approximation method is given in Algorithm 1. The problem is solved at the beginning of each global aggregation interval k' by the leader UAVs in each active swarm. The following proposition shows that the algorithm has the most desirable convergence properties for a non-convex solver:

Proposition 2. *Algorithm 1 generates a sequence of improved feasible solutions for problem $\widetilde{\mathcal{P}}_m$ that converge to a point $\mathbf{x}^\star$ satisfying the Karush-Kuhn-Tucker (KKT) conditions of $\mathcal{P}$.*

$$\overline{\Delta}_u(k) \triangleq \sum_{j \in \mathcal{W}_u} \left(\alpha_{j,1}(k) + \alpha_{j,2}(k) + \alpha_{j,3}(k) \right) \left(\sum_{i \in \mathcal{C}(u,s)} \rho_{i,j}(k)\widetilde{D}_i(k) + \sum_{h \in \widehat{\mathcal{W}}_u} \sum_{i \in \mathcal{C}(u,s)} \varrho_{h,j}(k)\rho_{i,h}(k)\widetilde{D}_i(k) \right) \tag{47}$$

$$\overline{\Delta}_u(k) \geq \widehat{\overline{\Delta}}_u(k) \triangleq \prod_{j \in \mathcal{W}_u} \left[\prod_{i \in \mathcal{C}(u,s)} \left[\left(\frac{\widetilde{\delta}_{i,j,1}(k) \left[\overline{\Delta}_u(k;m) \right]}{\widetilde{\delta}_{i,j,1}(k;m)} \right)^{\frac{\widetilde{\delta}_{i,j,1}(k;m)}{\overline{\Delta}_u(k;m)}} \left(\frac{\widetilde{\delta}_{i,j,2}(k) \left[\overline{\Delta}_u(k;m) \right]}{\widetilde{\delta}_{i,j,2}(k;m)} \right)^{\frac{\widetilde{\delta}_{i,j,2}(k;m)}{\overline{\Delta}_u(k;m)}} \left(\frac{\widetilde{\delta}_{i,j,3}(k) \left[\overline{\Delta}_u(k;m) \right]}{\widetilde{\delta}_{i,j,3}(k;m)} \right)^{\frac{\widetilde{\delta}_{i,j,3}(k;m)}{\overline{\Delta}_u(k;m)}} \right]$$

$$\prod_{h \in \widehat{\mathcal{W}}_u} \prod_{i \in \mathcal{C}(u,s)} \left[\left(\frac{\delta_{j,h,i,1}(k) \left[\overline{\Delta}_u(k;m) \right]}{\delta_{j,h,i,1}(k;m)} \right)^{\frac{\delta_{j,h,i,1}(k;m)}{\overline{\Delta}_u(k;m)}} \left(\frac{\delta_{j,h,i,2}(k) \left[\overline{\Delta}_u(k;m) \right]}{\delta_{j,h,i,2}(k;m)} \right)^{\frac{\delta_{j,h,i,2}(k;m)}{\overline{\Delta}_u(k;m)}} \left(\frac{\delta_{j,h,i,3}(k) \left[\overline{\Delta}_u(k;m) \right]}{\delta_{j,h,i,3}(k;m)} \right)^{\frac{\delta_{j,h,i,3}(k;m)}{\overline{\Delta}_u(k;m)}} \right] \right], \tag{48}$$

where

$$\widetilde{\delta}_{j,i,n}(k) \triangleq \alpha_{j,n}(k)\rho_{i,j}(k)\widetilde{D}_i(k), \quad \delta_{j,h,i,n}(k) \triangleq \alpha_{j,n}(k)\varrho_{h,j}(k)\rho_{i,h}(k)\widetilde{D}_i(k), \quad n \in \{1,2,3\}. \tag{49}$$

$$D_j(k) \triangleq \sum_{i \in \mathcal{C}(u,s)} \rho_{i,j}(k)\widetilde{D}_i(k) + \sum_{h \in \widehat{\mathcal{W}}_u} \sum_{i \in \mathcal{C}(u,s)} \varrho_{h,j}(k)\rho_{i,h}(k)\widetilde{D}_i(k), \tag{50}$$

$$D_j(k) \geq \widehat{D}_j(k) \triangleq \prod_{i \in \mathcal{C}(u,s)} \left(\frac{\widetilde{\lambda}_{i,j}(k)D_j(k;m)}{\widetilde{\lambda}_{i,j}(k;m)} \right)^{\frac{\widetilde{\lambda}_{i,j}(k;m)}{D_j(k;m)}} \prod_{h \in \widehat{\mathcal{W}}_u} \prod_{i \in \mathcal{C}(u,s)} \left(\frac{\lambda_{h,i,j}(k)D_j(k;m)}{\lambda_{h,i,j}(k;m)} \right)^{\frac{\lambda_{h,i,j}(k;m)}{D_j(k;m)}}, \tag{51}$$

where

$$\lambda_{i,j}(k) \triangleq \rho_{i,j}(k)\widetilde{D}_i(k), \quad \widetilde{\lambda}_{h,i,j}(k) \triangleq \varrho_{h,j}(k)\rho_{i,h}(k)\widetilde{D}_i(k). \tag{52}$$

Proof. See Appendix I. ∎

V. SWARM TRAJECTORY AND TEMPORAL DESIGN

Our developed `HN-PFL` methodology (Sec. I) and ML performance and energy optimization (Sec. II) are the fundamental components of our UAV-assisted ML methodology. In fact, in cases where the swarms are deployed and *stationary* above the clusters (e.g., [8]), these two components are enough to achieve optimal ML performance and energy efficiency.

Nevertheless, we go further by generalizing our methodology to a more realistic scenario where there are more clusters than UAV swarms, which requires UAV swarms to travel among the clusters. It is this generalization that motivates swarm trajectory and temporal (e.g., length of ML training sequence) design components (the green block of Fig. 2). Together our three components, depicted in Fig. 2 and developed in Sec. III, IV, and V, complete our methodology.

Upon completion of a training sequence using our `HN-PFL` and performance optimization components (Sec. III and IV), the core network instructs the UAV swarms to travel to their next device cluster, which will result in an update to both the global and cluster specific ML models, or to a recharging station. The core network also instructs the UAV swarms to discard their gathered data, as the local training and personalization conducted by each swarm should be specific to the unique underlying data distribution of its next visited swarm. We next design the swarm trajectories, sequence duration and aggregation period to maximize the ML performance.

A. Online Model Training under Model/Concept Drift

Since we consider *online model training*, where the data distributions at the devices are time varying, the performance of the local model changes over time. To capture this effect, we introduce the notion of *model/concept drift*.

Definition 1 (Model/Concept Drift). *For device cluster c with local meta-loss function F_c, we denote the online model drift at time t by $\Lambda_c(t) \in \mathbb{R}^+$, which captures the maximum variation of the gradient for any given model parameter across two consecutive time instances; mathematically:*

$$\left\| \nabla F_c \left(\mathbf{w} \Big| \tilde{\mathcal{D}}(t) \right) - \nabla F_c \left(\mathbf{w} \Big| \tilde{\mathcal{D}}(t-1) \right) \right\|^2 \leq \Lambda_c(t), \forall \mathbf{w}. \tag{53}$$

Device clusters with higher values of model drift are likely to require more frequent recalibration of their local models (achieved by revisiting with UAV swarms), as their local models become obsolete faster. In contrast, clusters with smaller model drift, i.e., small fluctuations in local gradient, may not be worth revisiting, due to marginal rewards (in terms of model performance gains) per energy consumed. Also, if clusters experience large and consistent model/concept drift, then the learning duration T_s should be smaller as the network will require swarms to re-calibrate local models more frequently. Model drift can be estimated every time a UAV swarm returns to a previously visited cluster by comparing historical meta-gradient vs. current meta-gradient computed by upon arrival.

Next, we demonstrate how model drift can be utilized to estimate local model performance given the current data distribution at a device cluster:

Lemma 4 (Estimating online gradient via model drift). *Let $t_c^{\vee}$ denote a time instance when device cluster c was visited by a UAV swarm, $\mathbf{w}_c(t_c^{\vee})$ denote the corresponding local model, and $\nabla F_c(\mathbf{w}_c(t_c^{\vee})|\widetilde{\mathcal{D}}_c(t_c^{\vee}))$ denote the local gradient. Consider time instance $t \geq t_c^{\vee} + 1$, where during the time interval $[t_c^{\vee} + 1, t]$ cluster c remains unvisited by UAV swarms. Given the updated data at device cluster c, i.e., $\widehat{\mathcal{D}}_c(t)$, the local meta gradient at the device cluster for the outdated local model is upper bounded as:*

$$\left\| \nabla F_c \left(\mathbf{w}_c(t_c^{\vee}) \Big| \widetilde{\mathcal{D}}_c(t) \right) \right\|^2 \leq \underbrace{(t - t_c^{\vee} + 1) \left\| \nabla F_c \left(\mathbf{w}_c(t_c^{\vee}) \Big| \widetilde{\mathcal{D}}_c(t_c^{\vee}) \right) \right\|^2}_{(a)}$$

$$+ \underbrace{(t - t_c^{\vee} + 1) \sum_{t'=t_c^{\vee}+1}^{t} \Lambda_c(t')}_{(b)}, \tag{54}$$

where $\Lambda_c(\cdot)$ is defined in (53). Subsequently, assuming an upper bound on the model drift at the device cluster $\Lambda_c(t') \leq \Lambda_c$, $t' \in [t_c^{\vee} + 1, t]$, we have: $\left\| \nabla F_c \left(\mathbf{w}_c(t_c^{\vee}) \Big| \widetilde{\mathcal{D}}_c(t) \right) \right\|^2 \leq (t - t_c^{\vee} + 1) \left\| \nabla F_c \left(\mathbf{w}_c(t_c^{\vee}) \Big| \widetilde{\mathcal{D}}_c(t_c^{\vee}) \right) \right\|^2 + (t - t_c^{\vee} + 1)(t - t_c^{\vee})\Lambda_c.$

Proof. See Appendix J. ∎

The bound in (54) demonstrates that the real-time performance of the outdated model, measured through the value of the meta gradient, becomes obsolete linearly with time (term (a)) and cumulative value of the model drift at the device cluster (term (b)). We will incorporate this result in the design of UAV swarm trajectories.

B. Problem Formulation and DRL Characteristics

1) Background: In the following, we first formulate the problem of UAV swarm trajectory design and learning sequence duration. We then cast the problem as a sequential decision making problem. Subsequently, we develop a deep reinforcement learning (DRL) methodology by encoding the real-time characteristics of our network into *states* for the DRL, defining a calculation methodology for the *actions* available for the DRL, and linking the actions to DRL *reward* calculations by embedding the optimization methodology from Section IV.

For training sequence s, let $X_u(s)$ denote the location of UAV swarm u, $\Lambda_c(s)$ denote the latest estimation of the model drift of device cluster c, $E_u(s)$ denote the minimum remaining battery among the UAVs belong to swarm u, and $G_c(s)$ denote the latest value of the gradient at device cluster c. Since the duration of ML model training T_s, $\forall s$, is usually far smaller than the delay between consecutive visits to the same device cluster, we assume that the effect of model drift during model training is negligible, as we do so in Sec. III, and instead integrate the effects of model drift in our DRL design caused by delay between consecutive visits of the device cluster. For device cluster c, if it has been visited during sequence s, i.e., $c \in \mathcal{C}(s)$, $G_c(s)$ captures its recent gradient: $G_c(s) = \left\| \nabla F_c \left(\mathbf{w}_c(t_s + T_s) | \widetilde{\mathcal{D}}_c(t_s + T_s) \right) \right\|^2$ measured at the end of the training sequence; however, if it is not visited, i.e., $c \in \overline{\mathcal{C}}(s)$, then $G_c(s) = \left\| \nabla F_c \left(\mathbf{w}_c(t_c^{\mathsf{V}}) | \widetilde{\mathcal{D}}_c(t) \right) \right\|^2$, which is computed via the bound in (54). Also, we let $E^{\mathsf{M}}(s)$ denote the sum of energy of movement of the UAV swarms to travel to their current location in training sequence s from their previous locations in sequence $s-1$, and let $O(s)$ denote the value of the final objective function of $\mathcal{P}$ solved in Section IV encompassing both the energy used for model training and the gradient of those devices under model training.

2) Problem Formulation: Given a total of S ML model training sequences, we propose the following formulation to determine the optimal trajectory design and temporal ML characteristics:

$$(\widehat{\mathcal{P}}) : \min_{\substack{\{\mathcal{X}(s)\}_{s=1}^S \\ \{\mathcal{T}(s)\}_{s=1}^S}} \frac{1}{S} \sum_{s=1}^S \left[c_1 E^{\mathsf{M}}(s) + c_2 O(s) + c_3 \sum_{c \in \overline{C}(s)} G_c(s) \right] \quad (55)$$

s.t.

$$E_u(s) \geq E^{\mathsf{T}}, u \in \mathcal{U}(s), s \in \{1, \cdots, S\} \quad (56)$$

$$\mathcal{T}(s) \in \mathcal{T}^{\mathsf{F}}, s \in \{1, \cdots, S\}, \quad (57)$$

where $\mathcal{X}(s) \triangleq \{X_1(s), \cdots, X_U(s)\}$, and $\mathcal{T}(s) \triangleq \{T_s \cup \tau_s^{\mathsf{L}} \cup \tau_s^{\mathsf{G}}\}$. The objective function in (55) consists of three parts, the energy required for swarms to move to their next destination ($E^{\mathsf{M}}(s)$), the data offloading/processing objective function result ($O(s)$

derived from $\mathcal{P}$), and the estimated online gradient as a result of model drift at clusters without UAV training ($G_c(s), c \in \overline{C}(s)$). Constraint (56) ensures that swarms have sufficient energy E^{T}, at all times, to travel to a recharging station before they can no longer sustain their flight, and constraint (57) guarantees that the temporal ML characteristics $\mathcal{T}(s) \triangleq \{T_s \cup \tau_s^{\mathsf{L}} \cup \tau_s^{\mathsf{G}}\}$, $\forall s$, are always within some feasible set $\mathcal{T}^{\mathsf{F}}$. Finally, c_1, c_2, and c_3 in (55) scale the objective components.

Solving $\widehat{\mathcal{P}}$ faces the following challenges: (i) the effects of training at a previous device cluster carries over to all future training sequences, at any device cluster, (ii) $\widehat{\mathcal{P}}$ is a combinatorial problem, which suffers from the *curse of dimensionality*, since at each sequence the core network must assign the swarms to either device clusters or recharging stations, and (iii) $\widehat{\mathcal{P}}$ is defined over an unknown environment, i.e., neither the data distributions at the device clusters nor the model/concept drift are known apriori. These facts motivate us to cast the problem as a *sequential decision making problem*, which is solved at the core network for each training sequence by encoding the network aspects as reinforcement learning objectives. Classical reinforcement learning methods rely on a pre-built Q-table to determine future actions and associated network states [39], but, due to the aforementioned challenges of our problem, in particular the curse of dimensionality, building a Q-table is infeasible. Motivated by the success that recurrent neural networks (RNN) exhibited as the deep Q-network for sequential decision making [40], we adapt an RNN based DRL featuring LSTM layers.

3) State of the DRL: We encode the locations of the UAV swarms, the model drift and gradients at device clusters, and the temporal ML characteristics (i.e., total time, local-global aggregation periods) as the state of the DRL. Formally, we define the state at the end of training sequence s, $\mathcal{Z}(s)$, as:

$$\mathcal{Z}(s) = \mathcal{U}(s) \cup \mathcal{X}(s) \cup \Lambda(s) \cup \mathcal{E}(s) \cup \mathcal{G}(s) \cup \mathcal{T}(s), \quad (58)$$

where $\mathcal{U}(s)$ denotes the set of active UAV swarms, $\mathcal{X}(s)$ denotes the locations of all the swarms, $\Lambda(s) = \{\Lambda_1(s), \cdots, \Lambda_C(s)\}$ are the latest model drift estimates observed at the device clusters, $\mathcal{E}(s) = \{E_1(s), \cdots, E_U(s)\}$, $\mathcal{G}(s) = \{G_1(s), \cdots, G_C(s)\}$, and $\mathcal{T}(s) = \{T_s \cup \tau_s^{\mathsf{L}} \cup \tau_s^{\mathsf{G}}\}$.

4) Action of the DRL: The core network, as the DRL agent, determines the active UAV swarms $\mathcal{U}(s+1)$, their locations $\mathcal{X}(s+1)$, and the temporal behavior $\mathcal{T}(s+1)$ of the next training sequence. In particular, we define the DRL action, $\mathcal{H}(s)$, as:

$$\mathcal{H}(s) = \mathcal{X}(s+1) \cup \mathcal{U}(s+1) \cup \mathcal{T}(s+1). \quad (59)$$

5) Reward of the DRL: The DRL agent aims to maximize the reward of each action $\mathcal{H}(s)$ with respect to the current state $\mathcal{Z}(s)$ via the objective value of $\widehat{\mathcal{P}}$. Formally, we define the reward of the agent as:

$$V(s) = C \underbrace{\left(c_1 O(s) + c_2 \sum_{c \in \overline{C}(s)} G_c(s) + c_3 E^{\mathsf{M}}(s) \right)^{-1}}_{(a)} - \underbrace{P \sum_{u \in \mathcal{U}} \mathbb{1}_{\{E_u \leq E^{\mathsf{T}}\}}}_{(b)}, (60)$$

where (a) captures the (inverse) value of the objective function of $\widehat{\mathcal{P}}$ for a particular training sequence s, and (b) is a penalty function with $P \gg 1$, capturing the case where UAVs' battery level drops below E^{T}, upon which the indicator $\mathbb{1}_{\{E_u \leq E^{\mathsf{T}}\}}$ takes the value of one for UAV swarm u.
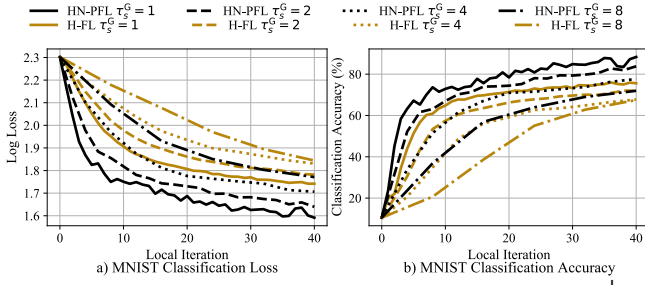
FIGURE 4: ML architecture comparisons for MNIST with fixed $\tau_s^L = 1$. HN-PFL demonstrates faster convergence and better final classification accuracy than H-FL for various τ_s^G.

FIGURE 5: ML architecture comparisons for CIFAR-10 with fixed $\tau_s^L = 1$. HN-PFL demonstrates faster convergence and better final classification accuracy than H-FL for various τ_s^G.

Main Takeaways. Our reward function in (60) captures multiple possibly competing objectives: (i) it motivates visiting device clusters that benefit from network-aware ML model training, i.e., those that require less energy to achieve a better model, (ii) it avoids leaving clusters with larger model drifts (via $G_c(s)$) unvisited for long periods of time, (iii) it promotes visiting clusters with high performance gain (via $O(s)$) as compared to ML energy consumption, and (iv) it avoids situations where swarm battery levels would drop below a threshold (via term (b)).

6) DRL Learning Architecture: We exploit an RNN with LSTM layers to approximate the optimal action-value function based on the Bellman equation:

$$Q^\star(\mathcal{Z}(s), \mathcal{H}(s)) = \mathbb{E}\Big[V(s)$$
$$+ \gamma \max_{\mathcal{H}(s+1)} Q^\star(\mathcal{Z}(s+1), \mathcal{H}(s+1)) \big| \mathcal{Z}(s), \mathcal{H}(s)\Big], \quad (61)$$

where γ is the future reward discount. We use one RNN to approximate $Q^\star$ called the *train Q-network* $Q_\theta(\mathcal{Z}(s), \mathcal{H}(s))$, which we train by adjusting its parameters θ to reduce the mean-squared error (MSE). Classical deep Q-network techniques determine the MSE of the train Q-network with respect to a target reward $y(s)$ that also depends on the train Q-network's model parameters θ. This self-coupling produces over-estimation, so we use an additional RNN called *target Q-network* $\widehat{Q}_{\widehat{\theta}}$ with parameters $\widehat{\theta}$ to produce accurate loss measurements and periodically synchronize the target Q-network to the train Q-network. Furthermore, to prevent correlations in the environment observation sequence from influencing the parameters θ in the *train Q-network*, we use *experience replay* to sample a randomized mini-batch of $\mathcal{M}^Q$ experience tuples, each of which is of the form $(\mathcal{Z}(s), \mathcal{H}(s), V(s), \mathcal{Z}(s+1))$, to calculate MSE. In particular, we compute the mean-squared error as:

$$L(\theta) = \frac{1}{|\mathcal{M}^Q|} \sum_{s \in \mathcal{M}^Q} \big(Q_\theta(\mathcal{Z}(s), \mathcal{H}(s)) - y(s)\big)^2, \quad (62)$$

where $y(s) = V(s) + \gamma \max_{\mathcal{H}(s+1)} \widehat{Q}_{\widehat{\theta}}(\mathcal{Z}(s+1), \mathcal{H}(s+1))$ is the target reward. Using the MSE, we then perform SGD to adjust the parameters θ. Since this procedure requires at least M^Q previous experience tuples (saved in a deque-style buffer of size B^Q), and each training sequence s only generates a single tuple, the RNN training requires at least M^Q training sequences before it begins. In order to generate representative experience tuples from the environment before the RNN is properly trained and to ensure that the DRL process is better
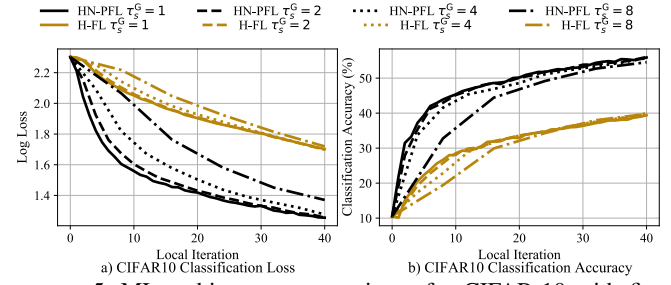
able to find the global maximum (rather than getting stuck in a local maxima), we utilize an ϵ-*greedy policy* [41], wherein with probability ϵ the agent will select an action randomly and with probability $(1 - \epsilon)$ the agent will determine the action based on the train Q-network Q_θ. As the train Q-network improves over time, ϵ decreases with limit $\epsilon^{\min} \leq \epsilon$.

C. System Integration: A Solution for UAV-enabled Online ML Over Heterogeneous Networks

In the past few sections, we developed our methodology for network-aware UAV-enabled online model training, which comprises HN-PFL from Sec. III, the data processing/offloading optimization from Sec. IV, and the swarm trajectory and ML design from Sec. V. The interconnection between the different components is visualized in Fig. 2, which we provided a high level discussion on in Sec. II-C. Given our introduced variables and parameters in Sec. III, IV, and V, we can now further comment on the interdependence.

Specifically, the HN-PFL methodology (orange block) is implemented by the UAV swarms, and operates based on the reception of transfer/processing parameters (i.e., $\rho, \varrho, \alpha, \mathbf{f}$) from the data processing optimization at the swarm leaders (blue block). The ML training results and UAV battery statuses are embedded into network states and form the bases of our swarm trajectory (i.e., $\mathcal{X}(s)$) and temporal training decisions (i.e., $\mathcal{T}(s), \tau_s^L, \tau_s^G$) for the training sequences (green block), which is then shared from the core network to the swarm leaders. With these core network decisions, the next sequence of ML training will begin, and the cycles continues.

VI. NUMERICAL EVALUATION AND DISCUSSION

In this section, we conduct numerical evaluations of our proposed methodology. When literature contains existing techniques, such as hierarchical federated learning (H-FL) [22], we compare our methodology against it; otherwise, we develop heuristic algorithms as baselines. For brevity, we present results here based on two datasets, MNIST (digits) and CIFAR-10 (color objects). Results for two other datasets, FMNIST (grayscale objects) and RADIOML (wireless signals) are deferred to Appendix A. Descriptions of each dataset and UAV network parameters can also be found in Appendix A. The key findings for each dataset are qualitatively consistent.

A. HN-PFL Proof-of-Concept

We start by investigating the HN-PFL procedure from Sec. III, by comparing its performance (measured via classification accuracy, loss, and energy consumption) to that
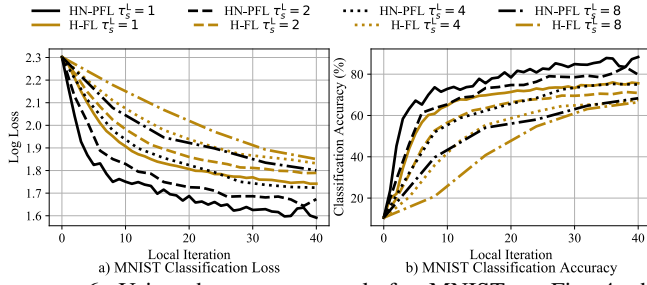
FIGURE 6: Using the same network for MNIST as Fig. 4, this experiment fixes $\tau_s^G = 1$ to vary τ_s^L instead. Our methodology `HN-PFL` continues to outperform `H-FL` consistently.

FIGURE 7: Using the same network for CIFAR10 as Fig. 5, this experiment fixes $\tau_s^G = 1$ to vary τ_s^L instead. Our methodology `HN-PFL` continues to outperform `H-FL` consistently.

TABLE II: Energy consumption for MNIST to reach 65% accuracy

Ratio		MNIST (kJ)			Ratio		MNIST (kJ)		
τ_s^L	τ_s^G	HFL	HNPFL	Savings	τ_s^L	τ_s^G	HFL	HNPFL	Savings
1	1	4.64	2.32	50.0%	1	1	4.64	2.32	50.0%
1	2	8.13	4.06	50.1%	2	1	9.29	4.64	50.1%
1	4	14.51	8.13	44.0%	4	1	15.10	8.13	46.2%
1	8	19.73	17.99	8.8 %	8	1	18.57	14.51	21.9%

of hierarchical federated learning (`H-FL`) proposed in [22]. For this simulation, we consider a network composed of 4 UAV swarms with 2-3 workers, where each swarm has data from only 3 labels (thus, non-iid data distributions) and data quantity determined randomly from a Gaussian distribution: $\mathcal{N}(2500, 250)$ for MNIST and $\mathcal{N}(3500, 350)$ for CIFAR-10. As a proof-of-concept, we strictly isolate the performance of the distributed ML methodologies to the UAV layers, i.e., we assume the data has already been transferred to the worker UAVs. For a fair comparison between `HN-PFL` and `H-FL`, we ensure that both methodologies train over the same amount of data for each of their iterations by defining the batch size for `H-FL` as α^{HFL} and setting each batch ratio for `HN-PFL` as $\alpha_{j,i} \triangleq \lfloor \frac{1}{3}\alpha^{HFL} \rfloor \ \forall i \in \{1, 2, 3\}$. Finally, we use the following settings for ML model training: $\eta = \eta_1 = 10^{-3}$ and $\eta_2 = 10^{-2}$, where η is the learning rate for gradient descent in `H-FL`, and to avoid lengthy Hessian computations, we use the Hessian first-order approximation (see Remark 1).

We investigate the impact of local and global aggregations on the performance and efficiency of the ML training separately. We first investigate the effects of varying global aggregation period τ_s^G and fix local aggregation period (i.e., $\tau_s^L = 1$) in Fig. 4 for MNIST. We repeat this experiment with fixed $\tau_s^G = 1$ and vary τ_s^L in Fig. 6, also for MNIST. The corresponding CIFAR-10 experiments are in Fig. 5 and 7. Due to the non-iid and non-convex natures of our problem, the noisy convergence seen in Figs. 4-7 is expected.

On both MNIST and CIFAR-10, we see that `HN-PFL` attains better final model accuracy and classification loss over its `H-FL` counterpart. For example, when $\tau_s^L = \tau_s^G = 1$, `HN-PFL` outperforms `H-FL` in trained classification accuracy by at least 10% on both MNIST and CIFAR-10. When we increase the aggregation period, the model performances obtained at any given local iteration are lower because the total iterations are fixed and thus longer aggregation periods result in fewer aggregations within the same timeframe. Fewer aggregations leads to worse performance as a result. Nonetheless, `HN-PFL` is able to maintain its advantage as τ_s^G or τ_s^L increases for both
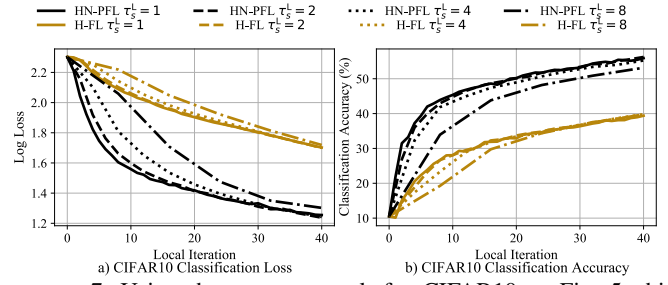
MNIST and CIFAR-10. Furthermore, the ability of `HN-PFL` to attain higher accuracy values with fewer training iterations than `H-FL` results in less network energy consumption in the form of flight, processing, and communication energy among UAVs in order to reach desired accuracy thresholds. We demonstrate the corresponding energy savings of `HN-PFL` in Table II for MNIST and Table III for CIFAR-10 to reach a specific training accuracy. For each table, we selected model accuracies that are reachable for all combinations of τ_s^L and τ_s^G. We chose 65% for MNIST and 25% for CIFAR-10. On average, our `HN-PFL` method saves 38.7% and 40.6% of the energy used for `H-FL` for MNIST and CIFAR-10, respectively.

B. Data Offloading/Processing Optimization

Next, we focus on the optimization (i.e., blue) block from Fig. 2, which relates to the network control aspect of our problem. Note that our optimization formulation from Sec. IV adjusts the performance and energy consumption of `HN-PFL` and is separable with respect to each swarm. We investigate the performance of our optimization solver on a single swarm composed of two workers and two coordinators training for a device cluster composed of ten devices in Figs. 8-9, averaging over 10 experiments. In practice, a network operator can vary $1 - \theta$, which scales the importance of the ML model performance vs. energy consumption in the objective of $\mathcal{P}$. In Fig. 8, our optimization responds by adjusting `HN-PFL`'s parameters: device-to-UAV data offloading ($\boldsymbol{\rho}$), coordinator-to-worker data offloading ($\boldsymbol{\varrho}$), aggregate data processing (i.e., $D_j \times \alpha_j$), and worker CPU frequency determination ($\boldsymbol{g}$). As the network operator places greater importance on ML model performance (i.e., increasing $1 - \theta$), our solver increases data offloading, i.e., larger average $\boldsymbol{\rho}$ in Fig. 8(a), and swarm leaders also instruct their workers to increase their CPU frequencies in Fig. 8(c). In this manner, more data is offloaded and subsequently processed within the same time frame, which increases the ML model performance. Fig. 8(b) also demonstrates that the coordinator UAVs never retain data for themselves as the average $\boldsymbol{\varrho}$ is 0.5, which implies that all the data is getting offloaded to the 2 workers. This is the case since the coordinator UAVs are only used for data relaying. The joint effect of larger average $\boldsymbol{\rho}$ and $\boldsymbol{g}$ is more average total data processed, seen in Fig. 8(d).

Next, in Fig. 9, we verify that our optimization formulation optimizes the performance of `HN-PFL`. As no alternative methodology for our optimization problem exists, we develop

FIGURE 8: The behavior of the optimization variables: $\boldsymbol{\rho}, \boldsymbol{\varrho}, \boldsymbol{\alpha}, \boldsymbol{g}$ depends on the ML learning importance factor $1 - \theta$ in the optimization formulation. As $1 - \theta$ increases, i.e., the importance of ML model performance increases, the network responds by offloading more data on average, increasing the CPU frequencies of workers, and processing more data on average as a result.
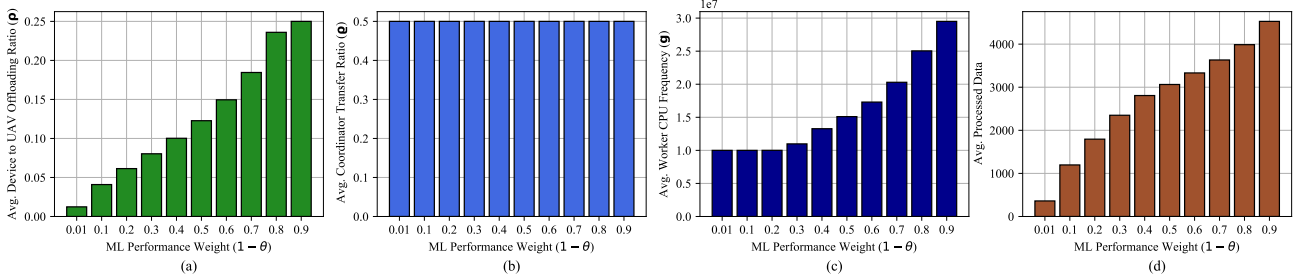
TABLE III: Energy consumption for CIFAR-10 to reach 25% accuracy.

Ratio		CIFAR-10 (kJ)			Ratio		CIFAR-10 (kJ)		
τ_s^{L}	τ_s^{G}	HFL	HNPFL	Savings	τ_s^{L}	τ_s^{G}	HFL	HNPFL	Savings
1	1	3.48	1.74	50.0%	1	1	3.48	1.74	50.0%
1	2	3.48	2.32	33.3%	2	1	3.48	2.32	33.3%
1	4	5.22	2.32	55.6%	4	1	5.22	2.32	55.6%
1	8	8.13	2.90	64.3%	8	1	8.13	2.90	64.3%

two methods, called greedy offloading (G.O.) and maximum processed (M.P.), and use them as baselines to compare the effectiveness of our data offloading/processing optimization in Fig. 9. In G.O., the devices always offload their entire datasets to the UAVs, and, in M.P., the UAVs overclock their CPU frequencies (reaching 2.3GHz for all UAVs) and maximize mini-batch ratios. Both G.O. and M.P. are determined in a fashion that adheres to the constraints in (25)-(46), and *we use our solver to determine the rest of the optimization variables in each baseline*. We show the percentage savings in objective function of $\mathcal{P}$ of our method over the two baselines in Fig. 9. When the network operator places greater emphasis on energy efficiency (i.e., small $1 - \theta$), our joint optimization will decrease data offloading and processing to conserve energy. As a result, our method achieves over 80% decrease in objective function value and energy consumption compared to either baseline when $1 - \theta = 0.01$. Even when $1 - \theta = 0.9$, i.e., the network aims to process more data in order to improve the ML component of the objective function, our method retains a roughly 6% improvement for the objective function and a 25% improvement for the energy consumption against both baselines.

C. Trajectory Optimization with Model/Concept Drift

Next, we turn to the trajectory optimization component of Sec. V (green block in Fig. 2). We consider 2 recharging stations and 8 device clusters separated by distances in $[500, 2000]$m, where each device cluster has $[8, 10]$ devices. At the UAV level, we consider 3 swarms, each of which has $[3, 5]$ workers and $[1, 2]$ coordinators. We refer to recharging stations as R:1 and R:2, and clusters as C:1, $\cdots$, C:8. First, we evaluate the performance of the DRL methodology by calculating the moving average reward from (60), average UAV battery levels, and learning objective (sum of the objective function $O(s)$ and the estimated online gradient $G_c(s)$) for our RNN-based DRL method with $\epsilon = \gamma = 0.7$ and three baselines in Fig. 10. Since existing baselines for our problem do not exist, we developed three baseline algorithms: (i) sequential heuristic (S.H.), (ii) greedy minimum distance (G.M.D.), and
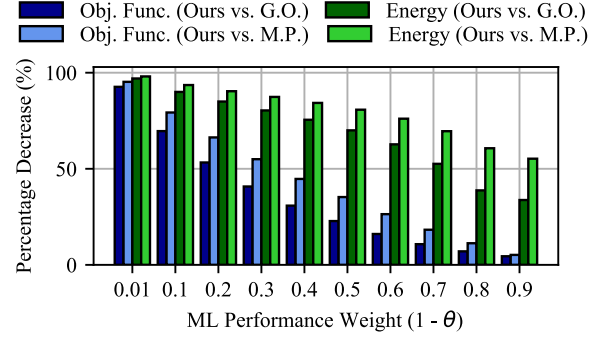
FIGURE 9: Our data offloading/processing optimization minimizes the network objective better than the two baselines, greedy offloading (G.O.) and maximum processed (M.P.). For comparison, we show the percentage decrease of energy consumption and objective function that our optimization achieves when compared to the baselines.

(iii) threshold minimum distance (T.M.D.), and calculated their rewards using the reward function in (60). We explain these baseline algorithms in Appendix A. To analyze the impact of UAV downtime due to recharging, we consider our methodology under three values of E^{T}, i.e., battery recharging threshold (RT): low (16.88 kJ), medium (25.32 kJ), and high (33.76 kJ). A large RT requires UAV swarms to recharge more frequently, inducing downtime in place of active model training, while a small RT permits UAV swarms to participate in more training sequences per recharge, which should lead to higher reward and learning objective for the overall network. We present the behavior of the reward, average battery level, and learning objective of these three recharging thresholds in Fig. 10. We quantify their respective recharging downtime by measuring the average recharging station visits per epoch: 0.125 for RT Low, 0.163 for RT Medium, and 0.228 for RT High, where a higher value indicates more downtime. We see that less recharging downtime (i.e., RT Low) indeed leads to the best overall reward and learning objective among test cases. We additionally note that our methodology is able to outperform each of the baselines by at least 7% in reward and 30% in learning objective across these values of RT. We further evaluate the sensitivity of our RNN-based DRL method to varying ϵ and γ in Appendix A.

As the other goal for our DRL-based trajectory optimization is adaptability to changing model drifts, we use the same network as that in Fig. 10 with time-varying model drifts for each device cluster, and measure the cluster visit rate per 1k epochs in Fig. 11. Each cluster has a unique affine

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3216326

17

FIGURE 10: Comparison of our method for swarm trajectory and temporal ML design against several baselines. In the legend, the first row denotes baseline algorithms and the second row denotes our method under three separate recharging thresholds (RT Low, RT Medium, and RT High). For all cases, our method learns optimal patterns for efficient swarm trajectories and outperforms the baselines.

FIGURE 11: Swarm visit statistics per thousand epochs at various training stages. Our method learns to favor visiting device clusters with the largest model drift (MD), even when the model drift is time-varying. Over time, cluster C:3 has the largest model drift of 21, and our method increases its visit rate from 0.21 to 0.45. Conversely, the final model drift at cluster C:4 is only 11, and our method decreases its visit rate from 0.22 to 0.17.

function to model its model drift growth. Initially, the 8 clusters have scaled model drifts of $[2, 4, 6, 8, 10, 12, 14, 16]$, but end at $[15, 15, 21, 11, 12, 17, 19, 17]$. So, initially, the cluster visit rate favors C:8 and C:7, which have the highest model drifts. However, as the epochs increase, the model drift begins skewing towards C:3, and our methodology responds by increasing its visit rate from 0.21 to $0.45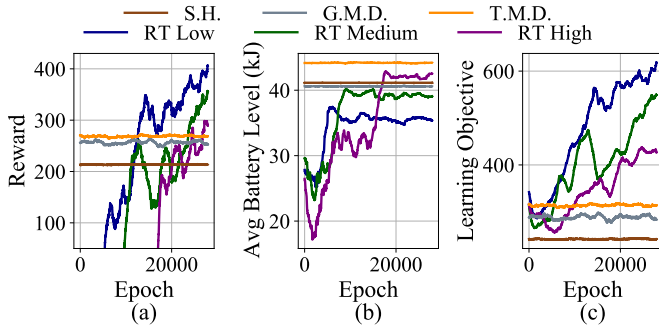$. Inversely, C:4 has the smallest final model drift and our method is able to adjust its visit rate from 0.22 to 0.17, showing our methodology's ability to provide more network services and UAV swarms to those device clusters that need it the most.

VII. CONCLUSION AND FUTURE WORK

We developed a holistic framework for integrating UAV swarm networks for online distributed machine learning. This involved a number of unique modelling decisions and analysis. We proposed a swarm stratification architecture tailored for our distributed machine learning framework. Our introduced distributed machine learning architecture, hierarchical nested personalized federated learning `HN-PFL`, nests meta-function based gradient descent into local and global aggregations through the worker-leader-core network hierarchy, for which we characterized the performance bound. Finally, we proposed and developed a holistic framework for network-aware UAV-enabled model training, consisting of two intertwined parts: (i) data offloading and processing optimization, for which we developed a distributed algorithm with performance guarantee, and (ii) learning duration and trajectory design, for which we developed a solution based on deep reinforcement learning.

One important direction for future work is the problem of swarm dimensioning, i.e., optimizing the number and type of UAVs comprising each swarm. UAVs could be intelligently exchanged in-between training periods to further optimize the tradeoff between model performance and energy consumption.

REFERENCES

[1] S. AbdulRahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5476–5497, 2020.

[2] O. A. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1342–1397, 2021.

[3] Q.-V. Pham, M. Le, T. Huynh-The, Z. Han, and W.-J. Hwang, "Energy-efficient federated learning over uav-enabled wireless powered communications," *IEEE Trans. Veh. Technol.*, 2022.

[4] 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, 2020.

[5] Y. Jiang, S. Wang, V. Valls, B. J. Ko, W.-H. Lee, K. K. Leung, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," *IEEE Trans. Neural Netw. Learn. Syst*, 2022.

[6] O. M. Rosabal, O. A. López, D. E. Pérez, M. Shehab, H. Hilleshein, and H. Alves, "Minimization of the worst-case average energy consumption in uav-assisted iot networks," *IEEE Internet Things J.*, 2022.

[7] X. Tu, K. Zhu, N. C. Luong, D. Niyato, Y. Zhang, and J. Li, "Incentive mechanisms for federated learning: From economic and game theoretic perspective," *IEEE Trans. Cogn. Commun. Netw.*, 2022.

[8] T. Zeng, O. Semiari, M. Mozaffari, M. Chen, W. Saad, and M. Bennis, "Federated learning in the sky: Joint power allocation and scheduling with UAV swarms," *arXiv:2002.08196*, 2020.

[9] H. Zhang and L. Hanzo, "Federated learning assisted multi-UAV networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 14 104–14 109, 2020.

[10] H. Yang, J. Zhao, Z. Xiong, K.-Y. Lam, S. Sun, and L. Xiao, "Privacy-preserving federated learning for UAV-enabled networks: Learning-based joint scheduling and resource management," *arXiv preprint arXiv:2011.14197*, 2020.

[11] R. Ciovacco, "Amazon Sidewalk Privacy and Security Whitepaper," Tech. Rep., 09 2020.

[12] E. L. Piza, B. C. Welsh, D. P. Farrington, and A. L. Thomas, "Cctv surveillance for crime prevention: A 40-year systematic review with meta-analysis," *Criminology & Public Policy*, vol. 18, no. 1, pp. 135–159, 2019.

[13] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz, "Help from the sky: Leveraging UAVs for disaster management," *IEEE Pervasive Comput.*, vol. 16, no. 1, pp. 24–32, 2017.

[14] 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, 2020.

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

[16] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, 2019.

[17] 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, 2021.

[18] S. Hosseinalipour, S. Wang, N. Michelusi, V. Aggarwal, C. G. Brinton, D. J. Love, and M. Chiang, "Parallel successive learning for dynamic distributed model training over heterogeneous wireless networks," 2022.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3216326

18

[19] S. Hosseinalipour, S. S. Azam, C. G. Brinton, N. Michelusi, V. Aggarwal, D. J. Love, and H. Dai, "Multi-stage hybrid federated learning over large-scale D2D-enabled fog networks," *IEEE/ACM Transactions on Networking*, 2022.

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

[21] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning: A meta-learning approach," *arXiv preprint arXiv:2002.07948*, 2020.

[22] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *IEEE Int. Conf. Commun. (ICC)*. IEEE, 2020, pp. 1–6.

[23] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues," *IEEE Commun. Surveys & Tuts.*, vol. 21, no. 4, pp. 3072–3108, 2019.

[24] N. Zhao, Z. Liu, and Y. Cheng, "Multi-agent deep reinforcement learning for trajectory design and power allocation in multi-UAV networks," *IEEE Access*, vol. 8, pp. 139 670–139 679, 2020.

[25] J. Cui, Y. Liu, and A. Nallanathan, "Multi-agent reinforcement learning-based resource allocation for UAV networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 2, pp. 729–743, 2019.

[26] K. Lu, J. Xie, Y. Wan, and S. Fu, "Toward UAV-based airborne computing," *IEEE Wireless Commun.*, vol. 26, no. 6, pp. 172–179, 2019.

[27] X. Liu, Y. Liu, Y. Chen, and L. Hanzo, "Trajectory design and power control for multi-UAV assisted wireless networks: A machine learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7957–7969, 2019.

[28] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2329–2345, 2019.

[29] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, 2017.

[30] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv:1703.03400*, 2017.

[31] A. Fallah, A. Mokhtari, and A. Ozdaglar, "On the convergence theory of gradient-based model-agnostic meta-learning algorithms," in *Int. Conf. Artificial Intell. Stat.* PMLR, 2020, pp. 1082–1092.

[32] A. Al-Hourani, S. Kandeepan, and A. Jamalipour, "Modeling air-to-ground path loss for low altitude platforms in urban environments," in *Proc. of IEEE Global Commun. Conf. (GLOBECOM)*, pp. 2898–2904, Austin, TX, USA, Dec. 2014.

[33] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Mobile unmanned aerial vehicles (UAVs) for energy-efficient internet of things communications," *IEEE Trans. Wireless Commun.*, vol. 16, no. 11, pp. 7574–7589, 2017.

[34] V. Vahidi and E. Saberinia, "Orthogonal frequency division multiplexing and channel models for payload communications of unmanned aerial systems," in *Int. Conf. Unmanned Aircraft Syst. (ICUAS)*, 2016, pp. 1156–1161.

[35] C. T. Dinh, N. H. Tran, M. N. H. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Trans. Netw.*, pp. 1–12, 2020.

[36] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3747–3760, June 2017.

[37] M. Chiang, C. W. Tan, D. P. Palomar, D. O'neill, and D. Julian, "Power control by geometric programming," *IEEE Trans. Wireless Commun.*, vol. 6, no. 7, pp. 2640–2651, 2007.

[38] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *J. Machine Learn. Research*, vol. 17, no. 83, pp. 1–5, 2016.

[39] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[40] A. Doshi, S. Yerramalli, L. Ferrari, T. Yoo, and J. G. Andrews, "A deep reinforcement learning framework for contention-based spectrum sharing," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2526–2540, 2021.

[41] B. C. Stadie, S. Levine, and P. Abbeel, "Incentivizing exploration in reinforcement learning with deep predictive models," *arXiv preprint arXiv:1507.00814*, 2015.

[42] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3747–3760, 2017.

[43] T. J. O'shea and N. West, "Radio machine learning dataset generation with gnu radio," in *Proceedings of the GNU Radio Conference*, vol. 1, no. 1, 2016.

[44] R. Sahay, C. G. Brinton, and D. J. Love, "A deep ensemble-based wireless receiver architecture for mitigating adversarial attacks in automatic modulation classification," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 1, pp. 71–85, 2021.

[45] T. L. Paine, C. Paduraru, A. Michi, C. Gulcehre, K. Zolna, A. Novikov, Z. Wang, and N. de Freitas, "Hyperparameter selection for offline reinforcement learning," *arXiv preprint arXiv:2007.09055*, 2020.

[46] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A tutorial on uavs for wireless networks: Applications, challenges, and open problems," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2334–2360, 2019.

[47] M. Ezuma, F. Erden, C. K. Anjinappa, O. Ozdemir, and I. Guvenc, "Detection and classification of uavs using rf fingerprints in the presence of wi-fi and bluetooth interference," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 60–76, 2019.

[48] S. Yang, Y. Luo, W. Miao, C. Ge, W. Sun, and C. Luo, "Rf signal-based uav detection and mode classification: a joint feature engineering generator and multi-channel deep neural network approach," *Entropy*, vol. 23, no. 12, p. 1678, 2021.

[49] Z. Sheng, H. D. Tuan, A. A. Nasir, T. Q. Duong, and H. V. Poor, "Secure uav-enabled communication using han–kobayashi signaling," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 2905–2919, 2020.

[50] T. V. Nguyen, T. V. Pham, N. T. Dang, and A. T. Pham, "Uav-based fso systems using sc-qam signaling over fading channels with misalignment," in *2020 IEEE 92nd Veh. Technol. Conf. (VTC2020-Fall)*. IEEE, 2020, pp. 1–5.

[51] Y. Shen, Y. Qu, C. Dong, F. Zhou, and Q. Wu, "Joint training and resource allocation optimization for federated learning in uav swarm," *IEEE Internet Things J.*, 2022.

[52] 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, 1978.
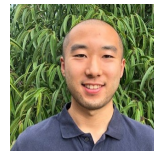
Su Wang is a PhD student in ECE at Purdue University. He received his BS in Electrical Engineering from Purdue in 2018.

Seyyedali Hosseinalipour received his Ph.D. in EE from NCSU in 2020. He has won the 2020 ECE Doctoral Scholar of the Year Award and 2021 ECE Distinguished Dissertation Award at NCSU. He was a postdoctoral researcher at Purdue University from 2020 to 2022. He is currently an assistant professor of EE at University at Buffalo (SUNY).

Maria Gorlatova is a Nortel Networks Assistant Professor of ECE at Duke University. She received her PhD from Columbia University in 2013. She won the NSF CAREER Award, the ACM/IEEE IPSN Best Research Artifact Award, and the IEEE Communications Society Young Author Best Paper Award.

Christopher G. Brinton is an assistant professor of ECE at Purdue University. He received his Ph.D. in EE from Princeton University in 2016. He is the recipient of the NSF CAREER Award, the ONR Young Investigator Program Award, the DARPA Young Faculty Award, and the Intel Rising Star Faculty Award.

Mung Chiang is the John A. Edwardson Dean of the College of Engineering and Executive Vice President at Purdue University. Previously, he was the Arthur LeGrand Doty Professor of EE at Princeton University. He received his Ph.D. from Stanford University in 2003. He won the 2013 Alan T. Waterman Award, the highest honor to U.S. young scientists and engineers.

APPENDIX A
SIMULATION SETTINGS AND ADDITIONAL SIMULATIONS

Network Characteristics and Parameters. To calculate data rates among pairs of networked devices/UAVs, i.e., (21), (22), we use the following set of values [33], [42]: $N_0 = -174$dBm/Hz, $\alpha^{\mathsf{PL}} = 2$, $\eta^{\mathsf{LoS}} = 3$dB, $\eta^{\mathsf{NLoS}} = 23$dB, $f = 2$GHz, $\psi = 11.95$, $\beta = 0.14$, $\overline{B} = 2$MHz. We set the transmit power of the devices in the range $[23, 25]$dBm and the transmit power of UAVs to 20dBm. The UAVs' altitude are also selected from $[25, 30]$m.

To determine the coefficients presented in Assumption 1 (i.e. B, γ_u^{H}, γ_u^{G}, γ^{H}, γ^{G}, σ_j^{G}, σ_j^{H}), we performed ML model training on our datasets, which yielded empirical estimates of these values. Then to ensure that they are upper bounds for more general test cases, we increase them by an order of magnitude for our simulations. To summarize, we used $B = 500$, $\gamma_u^{\mathsf{H}} = 0.1$, $\gamma_u^{\mathsf{G}} = 0.05$, $\gamma^{\mathsf{H}} = 0.1$, $\gamma^{\mathsf{G}} = 0.05$, $\sigma_j^{\mathsf{G}} = 50$, and $\sigma_j^{\mathsf{H}} = 50$.

Datasets. We consider the MNIST (http://yann.lecun.com/exdb/mnist/), Fashion-MNIST (FMNIST) (https://github.com/zalandoresearch/fashion-mnist), CIFAR-10 (http://www.cs.toronto.edu/~kriz/cifar.html), and RADIOML2016.10b [43] datasets. MNIST and FMNIST datasets contain 70K images (60K for training, 10K for testing), where each image belongs to one of 10 labels of hand-written digits and fashion products, respectively. FMNIST can be considered as a harder classification task as compared to MNIST due to its more complex images. While MNIST and FMNIST have grayscale images of 28x28 pixels, CIFAR-10 images are of 32x32 pixels and feature three channels (R,G,B), with the full dataset containing 60K images (50K for training, 10K for testing). On the other hand, RADIOML2016.10b (RADIOML) consists of waveform data, not images, and is commonly employed to assess ML algorithms for automatic modulation classification, i.e., classifying the modulation format of a particular signal. Similarly to existing works on RADIOML (e.g., [44]), we pre-process the data to obtain only the waveforms relevant to the CPFSK, GFSK, PAM4, and QPSK modulation schemes. Our resulting dataset size for RADIOML is 40K waveforms (32K for training, 8K for testing). For clarity, we reproduce a few constituent images from the datasets that we use for our simulations in Fig. 12.

Neural Network Architectures. For the MNIST and FMNIST datasets, we use a CNN with two convolutional layers followed by two linear layers. Both convolutional layers have kernel size 2, with the first layer outputting 16 maps and the second layer outputting 32 maps. The linear layers are correspondingly adjusted to fit the output dimensions of the second convolutional layer. For CIFAR-10 and RADIOML, we use a CNN with the same number of layers as MNIST and FMNIST, but with modified dimensions. Specifically, our CNN for CIFAR-10 uses kernel size 5 for the two convolutional layers, with 20 maps for the first convolutional layer and 50 maps for the second convolutional layer. Our CNN for RADIOML uses non-uniform kernel sizes of (2,5) and 16 maps for the first layer, and kernel sizes of (1,4) and 32 maps for the second layer. The linear layers of our CNNs for CIFAR-10 and RADIOML are correspondingly adjusted to fit the output dimensions of the second convolutional layer.

Reinforcement Learning Integration and Training. To ensure that our reinforcement learning training is based on the results of the optimization of `HN-PFL`, we simulate and then store the results of various configurations of our optimization formulation $\mathcal{P}$. For each unique configuration (e.g., $T_s = 100, \tau_s^{\mathsf{L}} = \tau_s^{\mathsf{G}} = 2$), we save the ML model performance result as the learning reward $O(s)$, and utilize the energy consumed to complete training to update the minimum UAV battery levels $E_u(s)$ for a swarm. In this way, when we start training the RNN-based deep reinforcement learning agent, the agent determines the optimal swarm trajectories and temporal ML characteristics based on the results of `HN-PFL` and our optimization formulation $\mathcal{P}$.

To train the DRL agent, we use train-target DQNs with an update period of 20 epochs and an experience replay deque of size 20. As the experience replay storage is initially empty, we initialize the training by running 20 random instances to fill the experience replay deque with the swarm trajectories and temporal decisions for `HN-PFL` and their associated reward. Thereafter, we train/update our DQN parameters after every epoch by calculating the mean square error (i.e., the loss) between the train and target network outputs, and then use gradient descent with learning rate 0.001 to update the train DQN's parameters. After every 20 epochs of training, we re-synchronize the train and target DQNs' model parameters.

At this point, our methodology will select subsequent swarm trajectory and temporal decisions for `HN-PFL` based on an epsilon-greedy policy, with $\epsilon = 0.7$. An epsilon-greedy policy means that, with probability $1 - \epsilon$, the DQN is used to determine the next swarm trajectory and temporal decisions. As our ϵ is linearly decaying with the number of training epochs, reaching a minimum of 0.005, our DQN is initially trained by biasing against random results, and, eventually when the DQN is well trained, our methodology will primarily adjust the DQN based on its own decisions.

Our swarm/UAV flight and communication characteristics used for the reinforcement learning training are the same as those used in Fig. 4- 14. For the additional reward coefficients found in (60), we use: $C = 10^5$, $c_1 = 0.2$, $c_2 = 0.25$, and $c_3 = 0.005$.
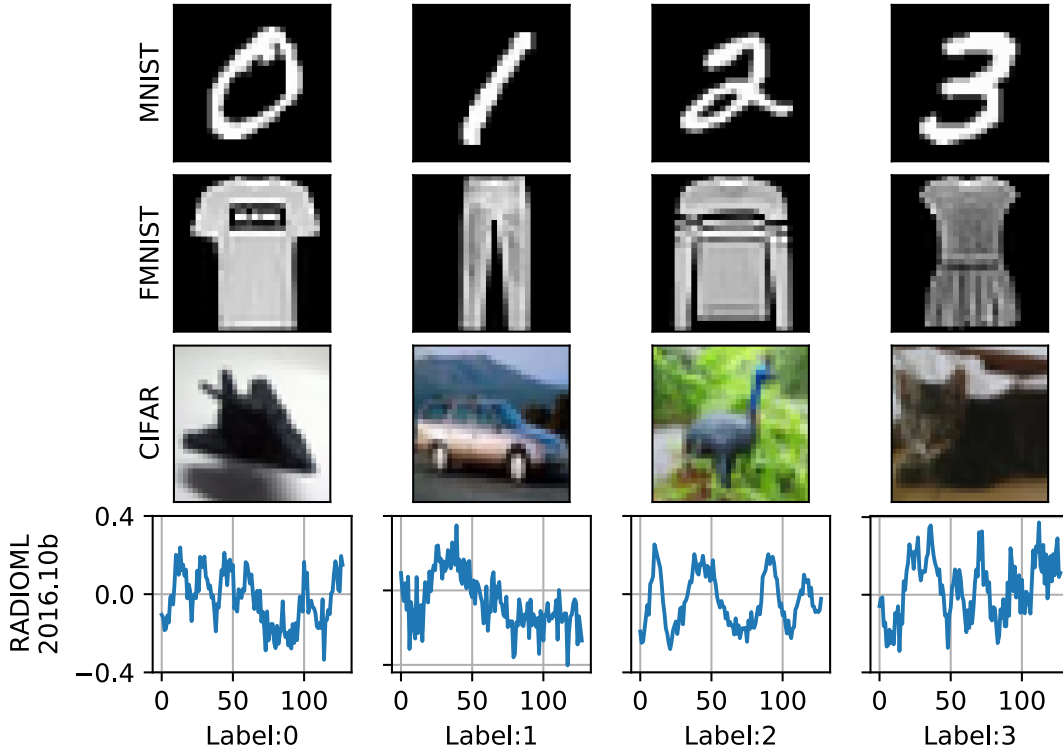
FIGURE 12: The datasets used for our simulations. We chose to use a classical digit recognition dataset for ML, MNIST, two object detection datasets, FMNIST (Fashion-MNIST) and CIFAR-10, an object detection task with RGB images, and a network management dataset RADIOML2016.10b (RADIOML).

Baseline Comparison Algorithms for Reinforcement Learning Method. In sec. VI, we compare our RNN-based DRL methodology against three baseline algorithms: (i) sequential heuristic (S.H.), (ii) greedy minimum distance (G.M.D.), and (iii) threshold minimum distance (T.M.D.). Due to space constraints, we explain them here. S.H. cycles through the clusters sequentially from C:1 to R:2, in the order presented in the x-axis of Fig. 11. On the other hand, our minimum distance methods determine the next destination for a swarm based on closest proximity, with G.M.D. always selecting the minimum distance and T.M.D. choosing between a random option or the minimum distance based on a probability threshold (set at 10% probability to select the random option). Both minimum distance based methods will reroute swarms to the nearest recharging station when needed.

Additional Simulations. In the following subsection, we will first present additional simulations on Fashion-MNIST (FMNIST) and RADIOML2016.10b (RADIOML) to verify the superiority of our `HN-PFL` methodology. Then, we will further demonstrate the veracity of our RNN-based DRL methodology via additional simulations, which separately vary ϵ and γ.

`HN-PFL` *Proof of Concept:* We further validate the improvement in accuracy and energy obtained by the `HN-PFL` methodology on FMNIST and RADIOML, and provide these simulation results here. Similar to the experiments for MNIST and CIFAR-10, we consider a network composed of 4 UAV swarms with 2-3 workers, where each swarm has data from only 3 labels (thus, non-iid data distributions) and data quantity determined randomly from a Gaussian distribution: N(3500, 350) for FMNIST and N(4500, 450) for RADIOML. The comparisons of classification accuracy for FMNIST are presented in Fig. 13 and 14, and the comparison of energy consumption is presented in Table IV. We see that our `HN-PFL` attains at least 5% better final classification accuracy relative to the baseline `H-FL` for variety of τ_s^{G} and τ_s^{L} combinations. Table IV also shows that `HN-PFL` attains more than 35% energy savings over `H-FL` for a variety of τ_s^{G} and τ_s^{L} combinations to reach 45% classification accuracy.

`HN-PFL` performs similarly well on RADIOML, the results of which are shown in Fig. 15 and 16 and Table V. We note that, on RADIOML, the training for both `HN-PFL` and `H-FL` started from a starter model, which was trained centrally via meta-gradient descent for 10 iterations. Fig. 15 and 16 show that `HN-PFL` attains at least 7% improvement over the baseline `H-FL` for a variety of τ_s^{G} and τ_s^{L} combinations. Furthermore, due to its faster convergence speed, `HN-PFL` saves over 48% of the energy used by `H-FL` to reach 80% classification accuracy.

RNN-based DRL Method: We further consider the sensitivity of our RNN-based DRL method for swarm trajectory and temporal
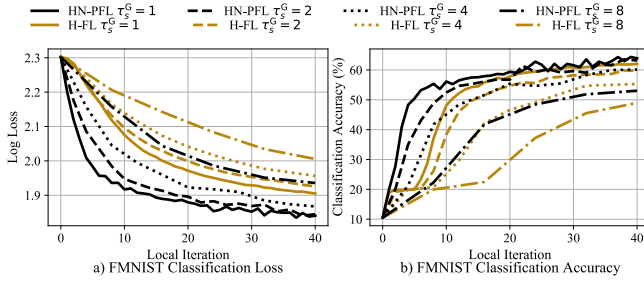
FIGURE 13: ML architecture comparisons for FMNIST with fixed $\tau_s^{\mathsf{L}} = 1$. HN-PFL demonstrates faster convergence than H-FL for various τ_s^{G}.
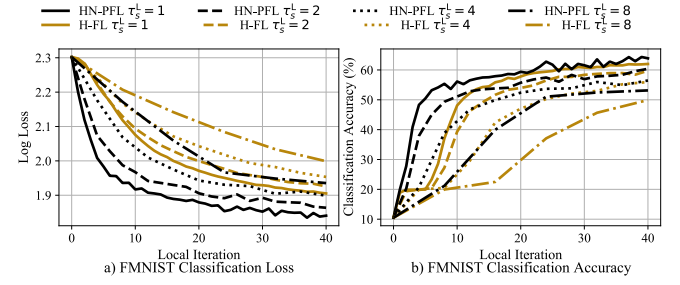
FIGURE 14: Using the same network for FMNIST as Fig. 13, this experiment fixes $\tau_s^{\mathsf{G}} = 1$ to vary τ_s^{L} instead. Our methodology HN-PFL converges more rapidly than H-FL for all test cases.

TABLE IV: Energy Consumption for FMNIST to Reach 45% Classification Accuracy

Ratio		FMNIST (kJ)			Ratio		FMNIST (kJ)		
τ_s^{L}	τ_s^{G}	HFL	HNPFL	Savings	τ_s^{L}	τ_s^{G}	HFL	HNPFL	Savings
1	1	5.22	2.32	55.6%	1	1	5.22	2.32	55.6%
1	2	6.96	3.48	50.0%	2	1	6.96	4.06	41.7%
1	4	11.03	6.38	42.2%	4	1	11.61	7.55	35.0%
1	8	17.99	11.61	35.5%	8	1	18.57	11.61	37.5%
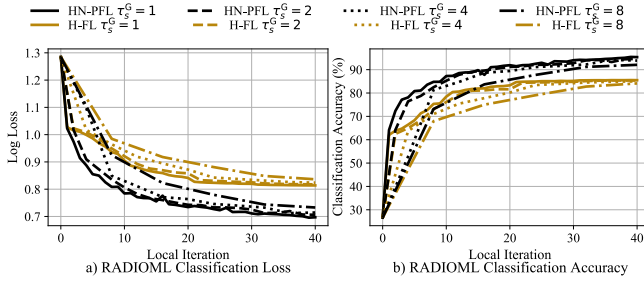
FIGURE 15: ML architecture comparisons for RADIOML with fixed $\tau_s^{\mathsf{L}} = 1$. HN-PFL demonstrates faster convergence than H-FL for various τ_s^{G}.
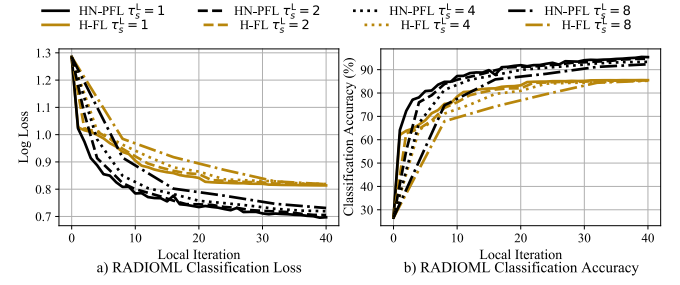
FIGURE 16: Using the same network for RADIOML as Fig. 15, this experiment fixes $\tau_s^{\mathsf{G}} = 1$ to vary τ_s^{L} instead. Our methodology HN-PFL converges more rapidly than H-FL for all test cases.

TABLE V: Energy Consumption for RADIOML to Reach 80% Classification Accuracy

Ratio		RADIOML (kJ)			Ratio		RADIOML (kJ)		
τ_s^{L}	τ_s^{G}	HFL	HNPFL	Savings	τ_s^{L}	τ_s^{G}	HFL	HNPFL	Savings
1	1	5.80	1.74	70.0%	1	1	5.80	1.74	70.0%
1	2	6.96	2.90	58.3%	2	1	7.45	2.90	61.1%
1	4	8.71	3.48	60.0%	4	1	9.29	4.06	56.3%
1	8	14.51	6.96	52.0%	8	1	14.51	7.45	48.7%

ML design to the values of ϵ and γ from Sec. V. We first evaluate varying ϵ from among $\{0.65, 0.7, 0.75\}$ in Fig. 17 with constant $\gamma = 0.7$ and then varying γ from among $\{0.65, 0.7, 0.75\}$ in Fig. 18 with constant $\epsilon = 0.7$. Foundational work, e.g., [45] has previously established that deep reinforcement learning techniques are sensitive to hyperparameters, such as our ϵ and γ terms, as we observe here.

The choice of ϵ influences the randomness with which actions are selected in the DRL method due to exploration, with a larger ϵ denoting more randomness and a smaller ϵ denoting less randomness. Additionally, the nominal value of ϵ gradually decreases during the training process, so a smaller ϵ means that the DRL method relies on the DRL agent more frequently and earlier on in the training process for decisions. A larger ϵ correspondingly means that the DRL method relies less on the DRL agent early on in the training process. For the choices ϵ shown in Fig. 17, we can see that our RNN-based DRL methodology is able to adapt and outperform the baselines over time.

The choice of γ determines the value of future rewards on the current reward computation. A larger γ means that the DRL agent will place greater emphasis on the anticipated future rewards during DRL agent training, while a smaller γ results in DRL agent training that relies more on instantaneous rewards, i.e., the specific rewards as a result of the current action. For the choices γ

shown in Fig. 18, we can see that our RNN-based DRL methodology is able to outperform the baselines for $\gamma \in \{0.7, 0.75\}$ and match the baselines for $\gamma = 0.65$ by the end of the training process. These results lead us to choose $\epsilon = 0.7, \gamma = 0.7$ for our other experiments.
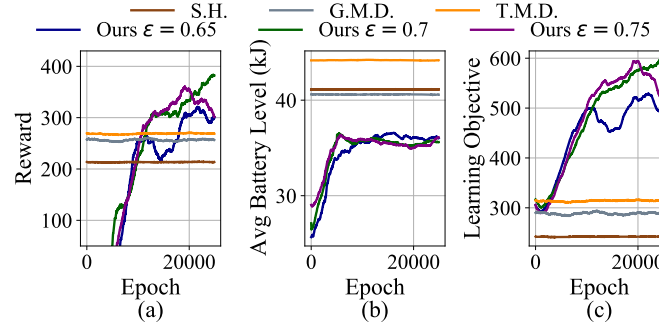
FIGURE 17: The impact of ϵ selection on our RNN-based DRL methodology for swarm trajectory and temporal ML design. Our method outperforms the baselines over time for all three ϵ values.
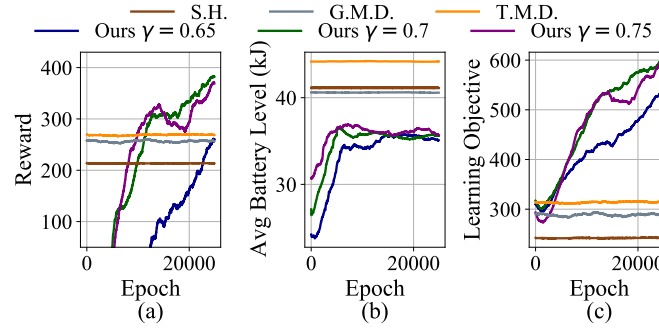
FIGURE 18: The impact of γ selection on our RNN-based DRL methodology for swarm trajectory and temporal ML design. We see that our method either outperforms ($\gamma \in \{0.7, 0.75\}$) or matches ($\gamma = 0.65$) the baselines in terms of reward by the end of the training process for all three values of γ.

Appendix B
A Discussion on UAV Signaling

UAV signaling, according to the highly cited paper [46], is typically used to coordinate data/information transfers, though it can also be used to detect/classify the presence of a UAV [47]. The specific design of a signaling layer for UAVs and devices is an important research problem, with works such as [48] devoted to using RF signaling techniques to optimize UAV classification and [49], [50] that develop coding or modulation techniques to better determine UAV signaling. Below, we outline the communication sequence of `HN-PFL` among devices, UAVs, and access points. For simplicity in the optimization formulation of $(\mathcal{P})$, we ignore interference from simultaneous transmissions to the UAVs and devices as they are stationary and can use of orthogonal frequency bands for all communications. We note that existing work, e.g., [51], in UAV networks for machine learning has yet to consider signaling details in-depth, which motivates a new research study with a comprehensive treatment of this topic.

Communications among different types of UAVs (leader, worker, and coordinator): The communication between the different types of UAVs happens sequentially. When a UAV swarm arrives above a device cluster, the leader signals to the worker and coordinator UAVs to travel to fixed positions above the device cluster, and sends coordinator UAVs the data offloading ratios for their transmissions to worker UAVs. Upon arrival at their given destinations, the coordinator and worker UAVs signal to the leader UAV that they have arrived, and, once the leader UAV receives this signal from all coordinator and worker UAVs, the leader UAV signals the commencement of the training sequence and the swarm-wide aggregation frequency to the worker UAVs. Upon reception, the coordinator UAVs signal to relevant worker UAVs that they will initiate data transfers, and begin transferring data received from devices to the worker UAVs. At swarm-wide aggregation stages, the workers signal to their leader that they will begin transmitting ML model parameters, and then transfer their local ML model parameters to their leader. Once the leader receives all local parameter vectors, it performs a swarm-wide aggregation, synchronizes the ML model parameters across workers, and then signals initiation of the next training round.

UAV requests for data from devices: Coordinator UAVs are assigned specific devices, based on our optimization formulation in Sec. IV, from which they collect data. After signaling to the leader UAV of their arrival at their fixed positions, coordinator UAVs send wake-up signals to their assigned devices to initiate data offloading. Devices then transmit their data to relevant coordinator UAVs.

Leader and access point interactions: At each global aggregation stage, the leader UAVs will travel to their nearest access point, and signal their arrival. Once the core network (via the APs) has received such a signal from all active leader UAVs, it will prompt leader UAVs to upload their swarm-wide ML model parameters. Once the core network has received all of these parameters, it will perform a global aggregation and synchronize the ML model parameters at leader UAVs with the result of the latest global aggregation. The leader UAVs will return to their respective swarms, upon synchronization completion, to commence the next training round.

Appendix C
Sketch Proof of Lemma 1

Sketch of Proof: The proof uses similar techniques to those found in Lemma 4.3 in [21]. We use our defined data processing ratios $\alpha_{j,1}, \alpha_{j,2}, \alpha_{j,3}$ in place for batch sizes within the data variability component of Assumption 1. ∎

Appendix D
Sketch Proof of Lemma 2

Sketch of Proof: The proof uses similar techniques to those found in Lemma 4.4 in [21]. The difference is that we use the weighted aggregation definition for the meta function in (6) before applying Assumption 1 and Jensen's inequality. ∎

APPENDIX E
PROOF OF PROPOSITION 1

Proof. To bound

$$\mathbb{E}\left[\left\|\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\Big(\mathbf{w}_j(t)-\overline{\mathbf{w}}(t)\Big)\right\|^2\right],\tag{63}$$

where $t = t_s^{\mathsf{G}}(k') = t_s + k'\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}$, we introduce $\overline{\mathbf{w}}_u(t)$ and omit the t_s aspect of $t_s^{\mathsf{G}}(k')$, i.e., ignoring the starting time of the sequence t_s in all the derivations since it is just a constant, to obtain:

$$\mathbb{E}\left[\left\|\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\Big(\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')-\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')+\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')-\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')\Big)\right\|^2\right]$$

$$\leq 2\underbrace{\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\mathbb{E}[\|\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')-\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')\|^2]}_{(i)}+2\underbrace{\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\mathbb{E}[\|\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')-\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')\|^2]}_{(ii)},\tag{64}$$

where the inequality is due to Jensen's inequality and $\|a+b\|^2 \leq 2(\|a\|^2+\|b\|^2)$. We first upper bound (64)(i), omitting the summations for simplicity:

$$2\mathbb{E}[\|\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')-\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')\|^2] \overset{(a)}{=} 2\mathbb{E}\left[\left\|\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')-\sum_{\hat{j}\in\mathcal{W}_u}\frac{\Delta_{\hat{j}}(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\mathbf{w}_{\hat{j}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')\right\|^2\right]$$

$$\overset{(b)}{=} 2\mathbb{E}\left[\left\|\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)-\eta_2\nabla\widetilde{F}_j(\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))-\sum_{\hat{j}\in\mathcal{W}_u}\frac{\Delta_{\hat{j}}(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\Big(\mathbf{w}_{\hat{j}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)-\eta_2\nabla\widetilde{F}_{\hat{j}}(\mathbf{w}_{\hat{j}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\Big)\right\|^2\right]$$

$$\overset{(c)}{\leq} 8\underbrace{\mathbb{E}[\|\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)-\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)\|^2]}_{(i)}+8\eta_2^2\underbrace{\mathbb{E}[\|\nabla F_j(\mathbf{w}_i(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))-\nabla\widetilde{F}_j(\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\|^2]}_{(ii)}$$

$$+8\eta_2^2\underbrace{\mathbb{E}\left[\left\|\sum_{\hat{j}\in\mathcal{W}_u}\frac{\Delta_{\hat{j}}(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\Big(\nabla\widetilde{F}_{\hat{j}}(\mathbf{w}_{\hat{j}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))-\nabla F_{\hat{j}}(\mathbf{w}_{\hat{j}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\Big)\right\|^2\right]}_{(iii)}$$

$$+8\eta_2^2\underbrace{\mathbb{E}\left[\left\|\nabla F_j(\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))-\sum_{\hat{j}\in\mathcal{W}_u}\frac{\Delta_{\hat{j}}(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\nabla F_{\hat{j}}(\mathbf{w}_{\hat{j}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\right\|^2\right]}_{(iv)},$$

$$\tag{65}$$

where (a) is the aggregation rule of $\overline{\mathbf{w}}_u$, (b) uses the gradient update from (1), and (c) introduces $\eta_2\nabla F_j$ and $\eta_2\nabla\overline{F}_u$ terms and then applies Cauchy-Schwarz, i.e., $(\sum_i^n a_i)^2 \leq n(\sum_i^n a_i^2)$. The bounds for (65)(ii) and (65)(iii) follow immediately from Lemma 1, and we analyze (65)(iv), while recalling the summations from (64) as follows:

$$8\eta_2^2\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\mathbb{E}\left[\left\|\nabla F_j(\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))-\sum_{\hat{j}\in\mathcal{W}_u}\frac{\Delta_{\hat{j}}(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\nabla F_{\hat{j}}(\mathbf{w}_{\hat{j}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\right\|^2\right]\tag{66}$$

$$\overset{(a)}{=} 8\eta_2^2\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\mathbb{E}\left[\left\|\nabla F_j(\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))-\nabla F_j(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))+\nabla F_j(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\right.\right.$$

$$\left.\left.-\sum_{\hat{j}\in\mathcal{W}_u}\frac{\Delta_{\hat{j}}(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\Big(\nabla F_{\hat{j}}(\mathbf{w}_{\hat{j}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))+\nabla F_{\hat{j}}(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))-\nabla F_{\hat{j}}(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\Big)\right\|^2\right]$$

$$\overset{(b)}{\leq} 24\eta_2^2\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\mathbb{E}[\|\nabla F_j(\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))-\nabla F_j(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\|^2]$$

$$+24\eta_2^2\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\mathbb{E}\left[\left\|\sum_{\hat{j}\in\mathcal{W}_u}\frac{\Delta_{\hat{j}}(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\Big(\nabla F_{\hat{j}}(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))-\nabla F_{\hat{j}}(\mathbf{w}_{\hat{j}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\Big)\right\|^2\right]$$

$$+24\eta_2^2\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\mathbb{E}\left[\left\|\nabla F_j(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))-\sum_{\hat{j}\in\mathcal{W}_u}\frac{\Delta_{\hat{j}}(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\nabla F_{\hat{j}}(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\right\|^2\right]$$

$$\overset{(c)}{\leq} 24\eta_2^2 \frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{j\in\mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \mathbb{E}[\|\nabla F_j(\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)) - \nabla F_j(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\|^2]$$

$$+ 24\eta_2^2 \frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{j\in\mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \sum_{\hat{j}\in\mathcal{W}_u} \frac{\Delta_{\hat{j}}(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \mathbb{E}\left[\left\|\nabla F_{\hat{j}}(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)) - \nabla F_{\hat{j}}(\mathbf{w}_{\hat{j}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\right\|^2\right]$$

$$+ 24\eta_2^2 \frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{j\in\mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \mathbb{E}\left[\left\|\nabla F_j(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)) - \nabla\overline{F}_u(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\right\|^2\right]$$

$$\overset{(d)}{\leq} 24\eta_2^2 \frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{j\in\mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \mu_F^2 \mathbb{E}[\|\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1) - \overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)\|^2]$$

$$+ 24\eta_2^2 \frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{\hat{j}\in\mathcal{W}_u} \frac{\Delta_{\hat{j}}(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \mu_F^2 \mathbb{E}[\|\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1) - \mathbf{w}_{\hat{j}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)\|^2]$$

$$+ 24\eta_2^2 \frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{j\in\mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \mathbb{E}\left[\left\|\nabla F_j(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)) - \nabla\overline{F}_u(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\right\|^2\right]$$

$$\overset{(e)}{\leq} 48\mu_F^2\eta_2^2 \frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{j\in\mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \mathbb{E}[\|\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1) - \overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)\|^2] + 24\eta_2^2 \frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} (\gamma_F^u)^2,$$

where (a) introduces $\nabla F_j(\overline{\mathbf{w}}_u)$ and $\nabla F_{\hat{j}}(\overline{\mathbf{w}}_u)$, (b) applies Jensen's inequality, (c) recalls the aggregation rule of $\nabla\overline{F}_u$ and applies Jensen's inequality, (d) uses μ_F-smoothness (i.e., $\|\nabla F_j(\mathbf{w}) - \nabla F_j(\mathbf{w}')\| \leq \mu_F\|\mathbf{w} - \mathbf{w}'\| \ \forall i \in \cup_{u\in\mathcal{U}(s)}\mathcal{W}_u$), and (e) leverages Lemma 2. Combining (65) and (66) yields:

$$2\frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{j\in\mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \mathbb{E}[\|\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k') - \overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')\|^2] \tag{67}$$

$$\overset{(a)}{\leq} 8\frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{j\in\mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \mathbb{E}[\|\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1) - \overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)\|^2]$$

$$+ 8\eta_2^2 \frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{j\in\mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \mathbb{E}[\|\nabla F_j(\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)) - \nabla\widetilde{F}_j(\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\|^2]$$

$$+ 8\eta_2^2 \frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{j\in\mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \mathbb{E}\left[\left\|\sum_{\hat{j}\in\mathcal{W}_u} \frac{\Delta_{\hat{j}}(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \left(\nabla\widetilde{F}_{\hat{j}}(\mathbf{w}_{\hat{j}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)) - \nabla F_{\hat{j}}(\mathbf{w}_{\hat{j}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\right)\right\|^2\right]$$

$$+ 8\eta_2^2 \frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{j\in\mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \mathbb{E}\left[\left\|\nabla F_j(\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)) - \sum_{\hat{j}\in\mathcal{W}_u} \frac{\Delta_{\hat{j}}(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \nabla F_{\hat{j}}(\mathbf{w}_{\hat{j}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\right\|^2\right]$$

$$\overset{(b)}{\leq} 8\frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{j\in\mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \mathbb{E}[\|\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1) - \overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)\|^2] + 16\eta_2^2 \frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{j\in\mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} (\sigma_j^{\mathsf{F}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))^2$$

$$+ 8\eta_2^2 \frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{j\in\mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \mathbb{E}\left[\left\|\nabla F_j(\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)) - \sum_{\hat{j}\in\mathcal{W}_u} \frac{\Delta_{\hat{j}}(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \nabla F_{\hat{j}}(\mathbf{w}_{\hat{j}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\right\|^2\right]$$

$$\overset{(c)}{\leq} (8 + 48\eta_2^2\mu_F^2)\frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{j\in\mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \mathbb{E}[\|\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1) - \overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)\|^2]$$

$$+ 16\eta_2^2 \frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{j\in\mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} (\sigma_j^{\mathsf{F}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))^2 + 24\eta_2^2 \frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} (\gamma_F^u)^2,$$

where (a) restates (65) with the previously omitted summations, (b) applies Jensen's inequality and Lemma 1, and (c) uses the result in (66). Solving (67) recursively yields:

$$2\frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{j\in\mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \mathbb{E}[\|\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k') - \overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')\|^2] \tag{68}$$

$$\overset{(a)}{\leq} 16\eta_2^2 \frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} \sum_{j\in\mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \left(\sum_{y=0}^{\tau_s^{\mathsf{L}}-1} (\sigma_j^{\mathsf{F}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1-y))^2 (8 + 48\eta_2^2\mu_F^2)^y\right) + 24\eta_2^2 \frac{1}{U(s)} \sum_{u\in\mathcal{U}(s)} (\gamma_F^u)^2 \frac{1-(8+48\eta_2^2\mu_F^2)^{\tau_s^{\mathsf{L}}}}{1-(8+48\eta_2^2\mu_F^2)} \tag{69}$$

$$\overset{(b)}{\leq} \left(16\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \sum_{y=0}^{\tau_s^{\mathsf{L}}-1} (\sigma_j^{\mathsf{F}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1 - y))^2 + 24\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} (\gamma_F^u)^2 \right) \frac{1 - (8 + 48\eta_2^2\mu_F^2)^{\tau_s^{\mathsf{L}}}}{1 - (8 + 48\eta_2^2\mu_F^2)}, \tag{70}$$

where (a) is the result of recursion and the finite sum of geometric series, and (b) uses $\sum_i a_i b_i \leq \sum_i a_i \sum_i b_i \ \forall a_i, b_i \geq 0$. Next, we bound (64)(ii) as follows:

$$2\frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}[\|\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k') - \overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')\|^2] \overset{(a)}{=} 2\frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}\left[\left\| \overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k') - \frac{1}{U(s)} \sum_{\hat{u} \in \mathcal{U}(s)} \overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k') \right\|^2\right]$$

$$\overset{(b)}{=} 2\frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}\left[\left\| \overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1) - \eta_2 \nabla \widetilde{\overline{F}}_u(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) - \frac{1}{U(s)} \sum_{\hat{u} \in \mathcal{U}(s)} \left(\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1) - \eta_2 \nabla \widetilde{\overline{F}}_{\hat{u}}(\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) \right) \right\|^2\right]$$

$$\overset{(c)}{=} 2\frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}\Bigg[\Bigg\| \overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1) - \eta_2 \nabla \widetilde{\overline{F}}_u(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) + \eta_2 \nabla \overline{F}_u(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) - \eta_2 \nabla \overline{F}_u(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1))$$

$$- \frac{1}{U(s)} \sum_{\hat{u} \in \mathcal{U}(s)} \left(\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1) - \eta_2 \nabla \widetilde{\overline{F}}_{\hat{u}}(\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) + \eta_2 \nabla \overline{F}_{\hat{u}}(\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) - \eta_2 \nabla \overline{F}_{\hat{u}}(\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) \right) \Bigg\|^2\Bigg]$$

$$\overset{(d)}{\leq} 8\frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \underbrace{\mathbb{E}\left[\left\| \overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1) - \frac{1}{U(s)} \sum_{\hat{u} \in \mathcal{U}(s)} \overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1) \right\|^2\right]}_{(i)}$$

$$+ 8\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \underbrace{\mathbb{E}[\|\nabla \overline{F}_u(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) - \nabla \widetilde{\overline{F}}_u(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1))\|^2]}_{(ii)}$$

$$+ 8\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \underbrace{\mathbb{E}\left[\left\| \frac{1}{U(s)} \sum_{\hat{u} \in \mathcal{U}(s)} \left(\nabla \widetilde{\overline{F}}_{\hat{u}}(\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) - \nabla \overline{F}_{\hat{u}}(\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) \right) \right\|^2\right]}_{(iii)}$$

$$+ 8\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \underbrace{\mathbb{E}\left[\left\| \frac{1}{U(s)} \sum_{\hat{u} \in \mathcal{U}(s)} \nabla \overline{F}_{\hat{u}}(\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) - \nabla \overline{F}_u(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) \right\|^2\right]}_{(iv)},$$

$$\tag{71}$$

where (a) uses the definition of $\overline{\mathbf{w}}$ from (7), (b) is the global parameter update rule with $\widetilde{\overline{F}}_u$ using an approximation $\widetilde{F}$ of F in (6) (the same applies for $\overline{F}$), (c) introduces $\eta_2 \nabla \overline{F}_u(\overline{\mathbf{w}}_u)$ and $\eta_2 \nabla \overline{F}_{\hat{u}}(\overline{\mathbf{w}}_{\hat{u}})$, and (d) follows from Cauchy-Schwarz inequality. Upper bounds for (71)(ii) and (71)(iii) follow immediately from Lemma 1, and we bound (71)(iv) as follows:

$$8\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}\left[\left\| \frac{1}{U(s)} \sum_{\hat{u} \in \mathcal{U}(s)} \nabla \overline{F}_{\hat{u}}(\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) - \nabla \overline{F}_u(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) \right\|^2\right]$$

$$\overset{(a)}{=} 8\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}\Bigg[\Bigg\| \frac{1}{U(s)} \sum_{\hat{u} \in \mathcal{U}(s)} \left(\nabla \overline{F}_{\hat{u}}(\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) - \nabla \overline{F}_{\hat{u}}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) + \nabla \overline{F}_{\hat{u}}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) \right)$$

$$- \nabla \overline{F}_u(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) + \nabla \overline{F}_u(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) - \nabla \overline{F}_u(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) \Bigg\|^2\Bigg]$$

$$\overset{(b)}{\leq} 24\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}\left[\left\| \frac{1}{U(s)} \sum_{\hat{u} \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_{\hat{u}}} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\Delta_{\hat{u}}(k'\tau_s^{\mathsf{G}})} \left(\nabla F_j(\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) - \nabla F_j(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) \right) \right\|^2\right]$$

$$+ 24\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}\left[\left\| \sum_{j \in \mathcal{W}_{\hat{u}}} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\Delta_{\hat{u}}(k'\tau_s^{\mathsf{G}})} \left(\nabla F_j(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) - \nabla F_j(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) \right) \right\|^2\right]$$

$$+ 24\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}[\|\nabla \overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) - \nabla \overline{F}_u(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1))\|^2]$$

$$\overset{(c)}{\leq} 24\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \frac{1}{U(s)} \sum_{\hat{u} \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_{\hat{u}}} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\Delta_{\hat{u}}(k'\tau_s^{\mathsf{G}})} \mathbb{E}\left[\left\| \nabla F_j(\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) - \nabla F_j(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k' - 1)) \right\|^2\right]$$

$$+ 24\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_{\hat{u}}} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\Delta_{\hat{u}}(k'\tau_s^{\mathsf{G}})} \mathbb{E}\left[\left\|\nabla F_j(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)) - \nabla F_j(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\right\|^2\right]$$

$$+ 24\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}[\|\nabla \overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)) - \nabla \overline{F}_u(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\|^2]$$

$$\overset{(d)}{\leq} 24\eta_2^2 \mu_F^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \frac{1}{U(s)} \sum_{\hat{u} \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_{\hat{u}}} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\Delta_{\hat{u}}(k'\tau_s^{\mathsf{G}})} \mathbb{E}[\|\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1) - \overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)\|^2]$$

$$+ 24\eta_2^2 \mu_F^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_{\hat{u}}} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\Delta_{\hat{u}}(k'\tau_s^{\mathsf{G}})} \mathbb{E}[\|\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1) - \overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)\|^2]$$

$$+ 24\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}[\|\nabla \overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)) - \nabla \overline{F}_u(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\|^2]$$

$$\overset{(e)}{\leq} 48\eta_2^2 \mu_F^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}[\|\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1) - \overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)\|^2] + 24\eta_2^2 \gamma_F^2, \tag{72}$$

where (a) introduces $\nabla \overline{F}_{\hat{u}}(\mathbf{w})$ and $\nabla \overline{F}_u(\mathbf{w})$ terms, (b) uses the Cauchy-Schwarz followed by the swarm-wide meta-gradient definition from (6), (c) applies Jensen's inequality to the expectations, (d) recalls the μ_F-smoothness property of ∇F_j, and (e) uses the result of Lemma 2. Combining (71) and (72) yields:

$$2\frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}[\|\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k') - \overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')\|^2]$$

$$\overset{(a)}{\leq} 8\frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}\left[\left\|\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1) - \frac{1}{U(s)} \sum_{\hat{u} \in \mathcal{U}(s)} \overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)\right\|^2\right]$$

$$+ 8\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}[\|\nabla \overline{F}_u(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)) - \nabla \widetilde{\overline{F}}_u(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\|^2]$$

$$+ 8\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}\left[\left\|\frac{1}{U(s)} \sum_{\hat{u} \in \mathcal{U}(s)} \left(\nabla \widetilde{\overline{F}}_{\hat{u}}(\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)) - \nabla \overline{F}_{\hat{u}}(\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\right)\right\|^2\right]$$

$$+ 48\eta_2^2 \mu_F^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}[\|\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1) - \overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)\|^2] + 24\eta_2^2 \gamma_F^2$$

$$\overset{(b)}{\leq} (8 + 48\eta_2^2 \mu_F^2)\frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}\left[\left\|\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1) - \frac{1}{U(s)} \sum_{\hat{u} \in \mathcal{U}(s)} \overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)\right\|^2\right] + 24\eta_2^2 \gamma_F^2$$

$$+ 8\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}\left[\left\|\sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\left(\nabla F_j(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)) - \nabla \widetilde{F}_j(\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\right)\right\|^2\right]$$

$$+ 8\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}\left[\left\|\frac{1}{U(s)} \sum_{\hat{u} \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\left(\nabla \widetilde{F}_j(\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)) - \nabla F_j(\overline{\mathbf{w}}_{\hat{u}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\right)\right\|^2\right]$$

$$\overset{(c)}{\leq} (8 + 48\eta_2^2 \mu_F^2)\frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}[\|\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1) - \overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)\|^2] + 24\eta_2^2 \gamma_F^2$$

$$+ 16\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}(\sigma_F^j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))^2, \tag{73}$$

where (a) combines (71) and (72), (b) applies definition of $\nabla \overline{F}_u$ from (6), (c) uses Jensen's inequality followed by Lemma 1 and the definition of $\overline{\mathbf{w}}$ from (7). Solving (73) recursively yields:

$$2\frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \mathbb{E}[\|\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k') - \overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')\|^2] \tag{74}$$

$$\overset{(a)}{\leq} 16\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\left(\sum_{y=0}^{\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}-1}(\sigma_j^{\mathsf{F}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1-y))^2(8 + 48\eta_2^2 \mu_F^2)^y\right) + 24\eta_2^2 \gamma_F^2 \frac{1 - (8 + 48\eta_2^2 \mu_F^2)^{\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}}}{1 - (8 + 48\eta_2^2 \mu_F^2)} \tag{75}$$

$$\overset{(b)}{\leq} \left(16\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \sum_{y=0}^{\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}-1} (\sigma_j^{\mathsf{F}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1-y))^2 + 24\eta_2^2\gamma_F^2\right) \frac{1 - (8 + 48\eta_2^2\mu_F^2)^{\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}}}{1 - (8 + 48\eta_2^2\mu_F^2)}, \tag{76}$$

where (a) is the result of recursion and the finite sum of geometric series, and (b) uses $\sum_i a_i b_i \leq \sum_i a_i \sum_i b_i \ \forall a_i, b_i \geq 0$. Combining the results of (68) and (74) in (64) yields:

$$2\frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \left(\mathbb{E}[\|\mathbf{w}_j(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k') - \overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')\|^2] + \mathbb{E}[\|\overline{\mathbf{w}}_u(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k') - \overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k')\|^2]\right)$$

$$\leq \left(16\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \sum_{y=1}^{\tau_s^{\mathsf{L}}} (\sigma_j^{\mathsf{F}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-y))^2 + 24\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} (\gamma_F^u)^2\right) \frac{1 - (8 + 48\eta_2^2\mu_F^2)^{\tau_s^{\mathsf{L}}}}{1 - (8 + 48\eta_2^2\mu_F^2)} \tag{77}$$

$$+ \left(16\eta_2^2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})} \sum_{y=1}^{\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}} (\sigma_j^{\mathsf{F}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-y))^2 + 24\eta_2^2\gamma_F^2\right) \frac{1 - (8 + 48\eta_2^2\mu_F^2)^{\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}}}{1 - (8 + 48\eta_2^2\mu_F^2)},$$

with the summation shifted from $y = 0$ to $y = 1$. ∎

APPENDIX F
PROOF OF THEOREM 1

Proof. First, from Assumption 1, we see that $F_j(t)$, $F_u(t)$, and $F(t)$ are all μ_F-smooth where $\mu_F = 4\mu^{\mathsf{G}} + \eta_1 \mu^{\mathsf{H}} B$ (result follows immediately from [21] upon using triangle inequality). In order to bound the first-order stationary point, $\mathbb{E}[\|\nabla \overline{F}(\overline{\mathbf{w}})\|^2]$, we start with

$$
\overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k')) \stackrel{(a)}{\leq} \overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)) + \nabla \overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1))^T (\overline{\mathbf{w}}(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k') - \overline{\mathbf{w}}(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)) + \frac{\mu_F}{2} \|\overline{\mathbf{w}}(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k') - \overline{\mathbf{w}}(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)\|^2
$$

$$
\stackrel{(b)}{=} \overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)) - \eta_2 \nabla \overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1))^T \left(\frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k' \tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k' \tau_s^{\mathsf{G}})} \nabla \widetilde{F}_j(\mathbf{w}_j(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)) \right)
$$

$$
+ \frac{\mu_F}{2} \left\| -\eta_2 \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k' \tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k' \tau_s^{\mathsf{G}})} \nabla \widetilde{F}_j(\mathbf{w}_j(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)) \right\|^2
$$

$$
\stackrel{(c)}{=} \overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)) - \eta_2 \nabla \overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1))^T \left(\frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k' \tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k' \tau_s^{\mathsf{G}})} \nabla \widetilde{F}_j(\mathbf{w}_j(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)) \right)
$$

$$
+ \eta_2^2 \frac{\mu_F}{2} \left\| \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k' \tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k' \tau_s^{\mathsf{G}})} \nabla \widetilde{F}_j(\mathbf{w}_j(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)) \right\|^2,
$$

(78)

where (a) follows from the μ_F-smoothness property of F, F_u, and F_j, (b) uses the meta-gradient update procedure from (1) and the definition of $\overline{\mathbf{w}}$ from (7), and (c) simplifies the algebra. Taking the expectation of both sides of (78) yields:

$$
\mathbb{E}[\overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k'))] - \mathbb{E}[\overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1))] \leq
$$

$$
\underbrace{-\eta_2 \mathbb{E}\left[\nabla \overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1))^T \left(\frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k' \tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k' \tau_s^{\mathsf{G}})} \nabla \widetilde{F}_j(\mathbf{w}_j(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)) \right) \right]}_{(i)}
$$

$$
+ \underbrace{\eta_2^2 \frac{\mu_F}{2} \mathbb{E}\left[\left\| \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k' \tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k' \tau_s^{\mathsf{G}})} \nabla \widetilde{F}_j(\mathbf{w}_j(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)) \right\|^2 \right]}_{(ii)}.
$$

(79)

Before we bound (79)(i) and (79)(ii), we first analyze

$$
\frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k' \tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k' \tau_s^{\mathsf{G}})} \nabla \widetilde{F}_j(\mathbf{w}_j(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1))
$$

$$
= X + Y + \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k' \tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k' \tau_s^{\mathsf{G}})} \nabla F_j(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)),
$$

(80)

where:

$$
X \triangleq \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k' \tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k' \tau_s^{\mathsf{G}})} \left[\nabla \widetilde{F}_j(\mathbf{w}_j(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)) - \nabla F_j(\mathbf{w}_j(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)) \right]
$$

$$
Y \triangleq \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k' \tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k' \tau_s^{\mathsf{G}})} \left[\nabla F_j(\mathbf{w}_j(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)) - \nabla F_j(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)) \right].
$$

(81)

We now bound $\|X\|^2$ as follows:

$$
\mathbb{E}[\|X\|^2] = \mathbb{E}\left[\left\| \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k' \tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k' \tau_s^{\mathsf{G}})} \left(\nabla \widetilde{F}_j(\mathbf{w}_j(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)) - \nabla F_j(\mathbf{w}_j(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)) \right) \right\|^2 \right]
$$

$$
\stackrel{(a)}{\leq} \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k' \tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k' \tau_s^{\mathsf{G}})} \mathbb{E}[\|\nabla \widetilde{F}_j(\mathbf{w}_j(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1)) - \nabla F_j(\mathbf{w}_j(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1))\|^2]
$$

$$
\stackrel{(b)}{\leq} \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k' \tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k' \tau_s^{\mathsf{G}})} (\sigma_j^{\mathsf{F}}(\tau_s^{\mathsf{L}} \tau_s^{\mathsf{G}} k' - 1))^2,
$$

(82)

using (a) Jensen's inequality, and (b) Lemma 1. Next, we bound $\|Y\|^2$ as follows:

$$
\begin{aligned}
\mathbb{E}[\|Y\|^2] &= \mathbb{E}\left[\left\|\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^\mathsf{G})}{\overline{\Delta}_u(k'\tau_s^\mathsf{G})}\Big(\nabla F_j(\mathbf{w}_j(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1)) - \nabla F_j(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))\Big)\right\|^2\right] \\
&\overset{(a)}{\leq} \mu_F^2\,\mathbb{E}\left[\left\|\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^\mathsf{G})}{\overline{\Delta}_u(k'\tau_s^\mathsf{G})}\Big(\mathbf{w}_j(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1) - \overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1)\Big)\right\|^2\right] \\
&\overset{(b)}{\leq} \mu_F^2\,\Upsilon(k'\tau_s^\mathsf{L}\tau_s^\mathsf{G}),
\end{aligned}
\tag{83}
$$

where (a) uses μ_F-Lipschitz gradient property of F_j, and (b) follows from Proposition 1, note that the recursion step which bounds $\mathbf{w}_j(t) - \overline{\mathbf{w}}(t)$ in Proposition 1 also bounds $\mathbf{w}_j(t-1) - \overline{\mathbf{w}}(t-1)$. Using these results, we first analyze (79)(i) as follows:

$$
\begin{aligned}
&-\eta_2\mathbb{E}\left[\nabla\overline{F}(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))^T\left(\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^\mathsf{G})}{\overline{\Delta}_u(k'\tau_s^\mathsf{G})}\nabla\widetilde{F}_j(\mathbf{w}_j(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))\right)\right] \\
&\overset{(a)}{=} -\eta_2\mathbb{E}\left[\nabla\overline{F}(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))^T\left(X+Y+\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^\mathsf{G})}{\overline{\Delta}_u(k'\tau_s^\mathsf{G})}\nabla F_j(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))\right)\right] \\
&\overset{(b)}{=} -\eta_2\mathbb{E}\left[\nabla\overline{F}(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))^T\left(X+Y+\nabla\overline{F}(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))\right)\right] \\
&\overset{(c)}{=} -\eta_2\left(\mathbb{E}[\nabla\overline{F}(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))^T(X+Y)] + \mathbb{E}[\|\nabla\overline{F}(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))\|^2]\right) \\
&\overset{(d)}{\leq} \frac{-\eta_2}{2}\mathbb{E}[\|\nabla\overline{F}(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))\|^2] + \eta_2\left(\mathbb{E}[\|X\|^2] + \mathbb{E}[\|Y\|^2]\right) \\
&\overset{(e)}{\leq} \frac{-\eta_2}{2}\mathbb{E}[\|\nabla\overline{F}(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))\|^2] + \eta_2\left(\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^\mathsf{G})}{\overline{\Delta}_u(k'\tau_s^\mathsf{G})}(\sigma_j^\mathsf{F}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))^2 + \mu_F^2\Upsilon(k',\tau_s^\mathsf{L},\tau_s^\mathsf{G})\right),
\end{aligned}
\tag{84}
$$

where (a) comes from introducing $\nabla F_j(\mathbf{w}_j)$ and $\nabla F_j(\mathbf{w})$ terms, (b) is the definition of $\nabla\overline{F}$, (c) follows from linearity of expectation, (d) is due to

$$
\begin{aligned}
-\eta_2\mathbb{E}[\nabla\overline{F}(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))^T(X+Y)] &\leq \frac{1}{2}\eta_2\mathbb{E}[\|\nabla\overline{F}(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))\|^2] + \frac{1}{2}\mathbb{E}[\|X+Y\|^2] \\
&\leq \frac{1}{2}\eta_2\mathbb{E}[\|\nabla\overline{F}(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))\|^2] + \eta_2\mathbb{E}[\|X\|^2] + \eta_2\mathbb{E}[\|Y\|^2]
\end{aligned}
\tag{85}
$$

(i.e., $(A+B)^2 \geq 0$ and Cauchy-Schwarz), and (e) applies the results of (82) and (83). Next, we analyze (79)(ii) as follows:

$$
\mathbb{E}\left[\left\|\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^\mathsf{G})}{\overline{\Delta}_u(k'\tau_s^\mathsf{G})}\nabla\widetilde{F}_j(\mathbf{w}_j(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))\right\|^2\right]
\tag{86}
$$

$$
\begin{aligned}
&\overset{(a)}{\leq} \mathbb{E}\left[\left\|X+Y+\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^\mathsf{G})}{\overline{\Delta}_u(k'\tau_s^\mathsf{G})}\nabla F_j(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))\right\|^2\right] \\
&\overset{(b)}{\leq} 3\mathbb{E}[\|X\|^2] + 3\mathbb{E}[\|Y\|^2] + 3\mathbb{E}\left[\left\|\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^\mathsf{G})}{\overline{\Delta}_u(k'\tau_s^\mathsf{G})}\nabla F_j(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))\right\|^2\right] \\
&\overset{(c)}{\leq} 3\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^\mathsf{G})}{\overline{\Delta}_u(k'\tau_s^\mathsf{G})}(\sigma_j^\mathsf{F}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))^2 + 3\mu_F^2\Upsilon(k'\tau_s^\mathsf{L}\tau_s^\mathsf{G}) \\
&\quad + 3\mathbb{E}\left[\left\|\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^\mathsf{G})}{\overline{\Delta}_u(k'\tau_s^\mathsf{G})}\nabla F_j(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))\right\|^2\right] \\
&\overset{(d)}{\leq} 3\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^\mathsf{G})}{\overline{\Delta}_u(k'\tau_s^\mathsf{G})}(\sigma_j^\mathsf{F}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))^2 + 3\mu_F^2\Upsilon(k'\tau_s^\mathsf{L}\tau_s^\mathsf{G}) \\
&\quad + 3\mathbb{E}\left[\left\|\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^\mathsf{G})}{\overline{\Delta}_u(k'\tau_s^\mathsf{G})}\Big(\nabla F_j(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1)) - \nabla\overline{F}(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1)) + \nabla\overline{F}(\overline{\mathbf{w}}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))\Big)\right\|^2\right] \\
&\overset{(e)}{\leq} 3\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^\mathsf{G})}{\overline{\Delta}_u(k'\tau_s^\mathsf{G})}(\sigma_j^\mathsf{F}(\tau_s^\mathsf{L}\tau_s^\mathsf{G}k'-1))^2 + 3\mu_F^2\Upsilon(k'\tau_s^\mathsf{L}\tau_s^\mathsf{G})
\end{aligned}
$$

$$+ 6\mathbb{E}\left[\left\|\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\left(\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}\nabla F_j(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1)) - \nabla F_u(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\right)\right\|^2\right] + 6\mathbb{E}\left[\left\|\nabla\overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\right\|^2\right]$$

$$\overset{(f)}{\leq} 6\mathbb{E}[\|\nabla\overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\|^2] + 3\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}(\sigma_j^{\mathsf{F}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))^2 + 3\mu_F^2\Upsilon(k'\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}) + 6\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}(\gamma_F^u)^2,$$

where, we (a) substitute the result from (80), (b) apply $(\sum_i^n a_i)^2 \leq n(\sum_i a_i^2)$, (c) combine the results of (82) and (83), (d) introduce the global gradient $\nabla\overline{F}$, (e) recall that $\nabla\overline{F} = \frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\nabla\overline{F}_u$ and use $(\sum_i^n a_i)^2 \leq n(\sum_i a_i^2)$, and (f) apply Jensen's inequality and Lemma 2. Combining the results of (84) and (86) in (79) yields:

$$\mathbb{E}[\overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'))] - \mathbb{E}[\overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))]$$

$$\leq -\frac{\eta_2}{2}\mathbb{E}[\|\nabla\overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\|^2] + \eta_2\left(\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}(\sigma_j^{\mathsf{F}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))^2 + \mu_F^2\Upsilon(k'\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}})\right)$$

$$+ \eta_2^2\frac{\mu_F}{2}\left(6\mathbb{E}[\|\nabla\overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\|^2] + 3\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}(\sigma_j^{\mathsf{F}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))^2 + 3\mu_F^2\Upsilon(k'\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}) + 6\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}(\gamma_F^u)^2\right).$$
(87)

With some algebra, we obtain:

$$\mathbb{E}[\overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'))] - \mathbb{E}[\overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))] \leq (6\eta_2^2\frac{\mu_F}{2} - \frac{\eta_2}{2})\mathbb{E}[\|\nabla\overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))\|^2]$$

$$+ (3\eta_2^2\frac{\mu_F}{2} + \eta_2)\left(\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}(\sigma_j^{\mathsf{F}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}k'-1))^2 + \mu_F^2\Upsilon(k'\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}})\right) + 3\eta_2^2\mu_F\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}(\gamma_F^u)^2.$$
(88)

Since our goal from the beginning was to find an upper bound for the first-order stationary point, $\mathbb{E}[\|\nabla\overline{F}(\overline{\mathbf{w}}(t))\|^2]$, we now take the average of (88) over all time t, which yields:

$$\frac{1}{\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}K_s^{\mathsf{G}}}\sum_{k'=1}^{K_s^{\mathsf{G}}}\sum_{k=(k'-1)\tau_s^{\mathsf{G}}+1}^{\tau_s^{\mathsf{G}}k'}\sum_{t=(k-1)\tau_s^{\mathsf{L}}+1}^{\tau_s^{\mathsf{L}}k}\mathbb{E}[\overline{F}(\overline{\mathbf{w}}(t))] - \mathbb{E}[\overline{F}(\overline{\mathbf{w}}(t-1))] = \frac{1}{\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}K_s^{\mathsf{G}}}(\mathbb{E}[\overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}K_s^{\mathsf{G}}))] - \mathbb{E}[\overline{F}(\overline{\mathbf{w}}(0))])$$

$$\leq \frac{1}{\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}K_s^{\mathsf{G}}}\sum_{k'=1}^{K_s^{\mathsf{G}}}\sum_{k=(k'-1)\tau_s^{\mathsf{G}}+1}^{\tau_s^{\mathsf{G}}k'}\sum_{t=(k-1)\tau_s^{\mathsf{L}}+1}^{\tau_s^{\mathsf{L}}k}\left[(6\eta_2^2\frac{\mu_F}{2} - \frac{\eta_2}{2})\mathbb{E}[\|\nabla\overline{F}(\overline{\mathbf{w}}(t-1))\|^2]\right.$$
(89)

$$+ (3\eta_2^2\frac{\mu_F}{2} + \eta_2)\left(\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}(\sigma_j^{\mathsf{F}}(t-1))^2 + \mu_F^2\Upsilon(k'\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}})\right) + 3\eta_2^2\mu_F\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}(\gamma_F^u)^2\bigg].$$

We set $\eta_2 < \frac{1}{6\mu_F}$ and obtain:

$$-\frac{6\eta_2^2\frac{\mu_F}{2} - \frac{\eta_2}{2}}{\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}K_s^{\mathsf{G}}}\sum_{k'=1}^{K_s^{\mathsf{G}}}\sum_{k=\tau_s^{\mathsf{G}}(k'-1)+1}^{\tau_s^{\mathsf{G}}k'}\sum_{t=\tau_s^{\mathsf{L}}(k-1)+1}^{\tau_s^{\mathsf{L}}k}\mathbb{E}[\|\nabla\overline{F}(\overline{\mathbf{w}}(t-1))\|^2]$$
(90)

$$\leq \frac{1}{\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}K_s^{\mathsf{G}}}\left(\mathbb{E}[\overline{F}(\overline{\mathbf{w}}(0))] - \mathbb{E}[\overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}K_s^{\mathsf{G}}))] + \sum_{k'=1}^{K_s^{\mathsf{G}}}\sum_{k=\tau_s^{\mathsf{G}}(k'-1)+1}^{\tau_s^{\mathsf{G}}k'}\sum_{t=\tau_s^{\mathsf{L}}(k-1)+1}^{\tau_s^{\mathsf{L}}k}\right.$$

$$\left[(3\eta_2^2\frac{\mu_F}{2} + \eta_2)\left(\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}(\sigma_j^{\mathsf{F}}(t-1))^2 + \mu_F^2\Upsilon(k',\tau_s^{\mathsf{L}},\tau_s^{\mathsf{G}})\right) + 3\eta_2^2\mu_F\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}(\gamma_F^u)^2\right]\bigg).$$

Finally, noting that $\mathbb{E}[\overline{F}(\overline{\mathbf{w}}(0))] = \overline{F}(\overline{\mathbf{w}}(0))$ and that $-\mathbb{E}[\overline{F}(\overline{\mathbf{w}}(\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}K_s^{\mathsf{G}}))] \leq -F^*$, we have the result:

$$\frac{1}{\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}K_s^{\mathsf{G}}}\sum_{k'=1}^{K_s^{\mathsf{G}}}\sum_{k=(k'-1)\tau_s^{\mathsf{G}}+1}^{\tau_s^{\mathsf{G}}k'}\sum_{t=(k-1)\tau_s^{\mathsf{L}}+1}^{\tau_s^{\mathsf{L}}k}\mathbb{E}[\|\nabla\overline{F}(\overline{\mathbf{w}}(t-1))\|^2] \leq \frac{1}{\frac{\eta_2}{2} - 6\eta_2^2\frac{\mu_F}{2}}\frac{1}{\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}}K_s^{\mathsf{G}}}\left[\overline{F}(\overline{\mathbf{w}}(0)) - F^* + \sum_{k'=1}^{K_s^{\mathsf{G}}}\sum_{k=(k'-1)\tau_s^{\mathsf{G}}+1}^{\tau_s^{\mathsf{G}}k'}\right.$$

$$\sum_{t=(k-1)\tau_s^{\mathsf{L}}+1}^{\tau_s^{\mathsf{L}}k}\left[(3\eta_2^2\frac{\mu_F}{2} + \eta_2)\left(\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k'\tau_s^{\mathsf{G}})}{\overline{\Delta}_u(k'\tau_s^{\mathsf{G}})}(\sigma_j^{\mathsf{F}}(t-1))^2 + \mu_F^2\Upsilon(k'\tau_s^{\mathsf{L}}\tau_s^{\mathsf{G}})\right) + 3\eta_2^2\mu_F\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}(\gamma_F^u)^2\right]\bigg].$$
(91)

$\blacksquare$

APPENDIX G
PROOF OF LEMMA 3

Expanding the left hand side of (19) using the definitions of $\nabla\overline{F}$ and $\nabla\overline{F}^{\mathsf{D}}$, and upper bounding it using the Jensen's inequality yields $\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\mathbb{E}\big[\|\nabla\overline{F}_u(\mathbf{w}(t)) - \nabla F^{\mathsf{D}}_{C(u,s)}(\mathbf{w}(t))\|^2\big]$, where we substitute $C(u,s)$ for c as the coupling of active UAV swarm to actively trained device cluster allows us to apply (6) onto $\nabla\overline{F}_u$ and obtain: $\frac{1}{U(s)}\sum_{u\in\mathcal{U}(s)}\sum_{j\in\mathcal{W}_u}\frac{\Delta_j(k)}{\Delta_u(k)}\mathbb{E}\big[\|\nabla F_j(\mathbf{w}(t)) - \nabla F^{\mathsf{D}}_{C(u,s)}(\mathbf{w}(t))\|^2\big]$ by applying of Jensen's inequality. Similar techniques used in Lemma 1 then yield the result.

APPENDIX H
AN OVERVIEW OF GEOMETRIC PROGRAMMING

A prerequisite to geometric programming (GP) is the notion of *monomials* and *posynomials*, which we provide below.

Definition 2. *A monomial is a function* $f : \mathbb{R}^n_{++} \to \mathbb{R}$: $f(\mathbf{y}) = dy_1^{a_1}y_2^{a_2}\cdots y_n^{a_n}$, *with* $d \geq 0$, $\mathbf{y} = [y_1, \cdots, y_n]$, *and* $a_j \in \mathbb{R}$, $\forall j$, *where* $\mathbb{R}^n_{++}$ *denotes the strictly positive quadrant of n-dimensional Euclidean space. Also, a posynomial g is a sum of monomials:* $g(\mathbf{y}) = \sum_{m=1}^M d_m y_1^{\alpha_m^{(1)}}\cdots y_n^{\alpha_m^{(n)}}$.

A standard GP is a non-convex problem formulated as minimizing a posynomial under posynomial inequality constraints and monomial equality constraints [37]:

$$\min_{\mathbf{y}} g_0(\mathbf{y})$$
$$\text{s.t.}$$
$$g_i(\mathbf{y}) \leq 1, \ i = 1, \cdots, I, \tag{92}$$
$$f_\ell(\mathbf{y}) = 1, \ \ell = 1, \cdots, L,$$

where $g_i(\mathbf{y}) = \sum_{m=1}^{M_i} d_{i,m} y_1^{a_{i,m}^{(1)}}\cdots y_n^{a_{i,m}^{(n)}}$, $\forall i$, and $f_\ell(\mathbf{y}) = d_\ell y_1^{a_\ell^{(1)}}\cdots y_n^{a_\ell^{(n)}}$, $\forall \ell$. Since the log-sum-exp function $f(\mathbf{y}) = \log\sum_{j=1}^n e^{y_j}$ is convex, where $\log$ denotes the natural logarithm, with logarithmic change of variables and constants $z_i = \log(y_i)$, $b_{i,k} = \log(d_{i,k})$, $b_\ell = \log(d_\ell)$, and applying the $\log$ on the objective and constrains of (92), the GP in its standard format can be transformed to the following convex programming formulation:

$$\min_{\mathbf{z}} \ \log \sum_{m=1}^{M_0} e^{\left(\mathbf{a}_{0,m}^\top \mathbf{z} + b_{0,m}\right)}$$
$$\text{s.t.} \ \log \sum_{m=1}^{M_i} e^{\left(\mathbf{a}_{i,m}^\top \mathbf{z} + b_{i,m}\right)} \leq 0, \ i = 1, \cdots, I, \tag{93}$$
$$\mathbf{a}_\ell^\top \mathbf{z} + b_\ell = 0, \ \ell = 1, \cdots, L,$$

where $\mathbf{z} = [z_1, \cdots, z_n]^\top$, $\mathbf{a}_{i,m} = \left[a_{i,m}^{(1)}, \cdots, a_{i,m}^{(n)}\right]^\top$, $\forall i, m$, and $\mathbf{a}_\ell = \left[a_\ell^{(1)}, \cdots, a_\ell^{(n)}\right]^\top$, $\forall \ell$.

As can be seen, in $\mathcal{P}$, there are multiple terms in the objective function (in the upper bound of convergence of the ML model) that are in the format of ratio between two posynomials, which are not posynomial. We thus aim to transform a ratio of two posynomials to a ratio between a posynomial (in the numerator) and a monomial (in the denominator). Given the fact that the ratio between a posynomial and a monomial is a posynomial, we then aim to transform the problem to the standard GP format. To carry out this transformation, we exploit arithmetic-geometric mean inequality which lower bounds a posynomial with a monomial.

Lemma 5 (Arithmetic-geometric mean inequality [37]**).** *A posynomial* $g(\mathbf{x}) = \sum_{k=1}^K u_k(\mathbf{x})$, *where $u_k(\mathbf{x})$ is a monomial, $\forall k$, can be lower-bounded via a monomial as follows:*

$$g(\mathbf{x}) \geq \hat{g}(\mathbf{x}) \triangleq \prod_{k=1}^K \left(\frac{u_k(\mathbf{x})}{a_k(\mathbf{y})}\right)^{a_k(\mathbf{y})}, \tag{94}$$

where $a_k(\mathbf{y}) = u_k(\mathbf{y})/g(\mathbf{y})$, $\forall k$, *and* $\mathbf{y} > 0$ *is a fixed point.*

APPENDIX I
PROOF OF PROPOSITION 2

Let us first rewrite problem $(\mathcal{P})$ in its equivalent form as follows:

$$(\mathcal{P}): \min_{\boldsymbol{\rho},\boldsymbol{\varrho},\boldsymbol{\alpha},\boldsymbol{g},\Omega} \quad \Omega \tag{95}$$

$$\text{s.t.} \tag{96}$$

$$(25)-(46), \tag{97}$$

$$(1-\theta)(a) + \theta(b) \leq \Omega, \; \Omega \geq 0, \tag{98}$$

where $\Omega \in \mathbb{R}^+$ is an auxiliary variable used to move the objective function into the constraints, and (a), and (b) are those introduced in (24). The corresponding approximated problem can also be expressed as follows:

$$(\widehat{\mathcal{P}}_m): \min_{\boldsymbol{\rho},\boldsymbol{\varrho},\boldsymbol{\alpha},\boldsymbol{g},\Omega} \quad \Omega \qquad \text{s.t.} \;\; (25)-(46), \; (1-\theta)\widetilde{(a)}_m + \theta(b) \leq \Omega, \; \Omega \geq 0, \tag{99}$$

where $\theta\widetilde{(a)}$ follows from the procedure outlined to obtain $\widetilde{\mathcal{P}}_m$ in the main text, i.e., applying the posynomial condensation technique. It is easy to verify that the solution of $\widetilde{\mathcal{P}}_m$ coincides with that of $\widehat{\mathcal{P}}_m$. Thus to prove the proposition, it is enough to prove that solving $\widehat{\mathcal{P}}_m$ generates a sequence of improved feasible solutions for problem $\widehat{\mathcal{P}}_m$ that converge to a point $\boldsymbol{x}^\star$ satisfying the Karush-Kuhn-Tucker (KKT) conditions of $\mathcal{P}$. Note that $\widehat{\mathcal{P}}_m$ can be solved using the procedure Algorithm 1.
Following the justifications in **Observation 2** in Sec. IV-B, the constraints of $\widehat{\mathcal{P}}_m$ are separable with respect to each individual UAV swarm. Thus, the performance of the distributed algorithm proposed to solve $\widehat{\mathcal{P}}_m$ distributedly at each UAV swarm coincides with that of the centralized one for a fixed set of estimated parameters. Under the approximations described in (48) and (51), the algorithm in fact solves an *inner approximation* of problem $\mathcal{P}$ [52]. Hence, it is sufficient to prove the following three conditions for the sequence of generated solutions by the algorithm [52]:

1) *The approximations used in problem $\widehat{\mathcal{P}}_m$ should tighten the constraints of problem $\mathcal{P}$:* Since the constraints (25)-(46) are common to both Problems $(\mathcal{P})$ and $\widehat{\mathcal{P}}_m$, it is enough to show that $(1-\theta)(a) + \theta(b) \leq (1-\theta)\widetilde{(a)} + \theta(b)$ for the approximated constraint, assuming some solution $\boldsymbol{x}[m]$ ($\boldsymbol{x}$ is the solution vector defined in Algorithm 1). Equivalently, it is sufficient to show that $(a) \leq \widetilde{(a)}$ for $\boldsymbol{x}[m]$.

   To show this, it is sufficient to show (i) $\Xi(s) \leq \widetilde{\Xi(s)}$, and (ii) $\widehat{\Xi}(k) \leq \widetilde{\widehat{\Xi}(k)}$, $\forall k$, where $\Xi(s), \widehat{\Xi}(k)$ are the two terms in (a) (see (24)) and $\widetilde{\Xi(s)}$ and $\widehat{\Xi}(k)$ are their respective approximations obtained via (48) and (51). Condition (i) holds since $\overline{\Delta}_u(k) \geq \widehat{\overline{\Delta}}_u(k)$, and condition (ii) also holds since $D_j(k) \geq \widehat{D}_j(k) \; \forall k$, as in equations (48) and (51).[3]

2) *Upon convergence, the value of each approximated constraint in problem $\widehat{\mathcal{P}}_m$ should coincide with that of the corresponding original constraint in $\mathcal{P}$:* Since the constraints (25)-(46) are common in problems $(\mathcal{P})$ and $(\widehat{\mathcal{P}})$, we need to show $(1-\theta)(a)+\theta(b) = (1-\theta)\widetilde{(a)} + \theta(b)$ for the approximated constraint upon convergence. Note that when the algorithm converges we have

$$\widetilde{\delta}_{j,i,n}(k) = \widetilde{\delta}_{j,i,n}(k;m), \quad \delta_{j,h,i,n}(k) = \delta_{j,h,i,n}(k;m), \quad n \in \{1,2,3\}, \tag{100}$$

where $\widetilde{\delta}_{j,i,n}(k;m)$ and $\delta_{j,h,i,n}(k;m)$ are the approximations used in (48) (see (49) for the definition of $\widetilde{\delta}_{j,i,n}(k)$ and $\delta_{j,h,i,n}(k)$). Also, upon convergence, we have:

$$\lambda_{i,j}(k) = \lambda_{i,j}(k;m), \quad \widetilde{\lambda}_{h,i,j}(k) = \widetilde{\lambda}_{h,i,j}(k;m), \tag{101}$$

where $\lambda_{i,j}(k;m), \widetilde{\lambda}_{h,i,j}(k;m)$ are the approximations used in (51) (see (52) for the definition of $\lambda_{i,j}(k), \widetilde{\lambda}_{h,i,j}(k)$).

Considering the terms inside (a), we need to show that upon convergence (i) $\Xi(s) = \widetilde{\Xi(s)}$, and (ii) $\widehat{\Xi}(k) = \widetilde{\widehat{\Xi}}(k)$. To demonstrate that (i) holds, it is sufficient to show that $\overline{\Delta}_u(k) = \widehat{\overline{\Delta}}_u(k)$, and $D_j(k) = \widehat{D}_j(k)$, upon convergence,. In the following, we demonstrate that $\overline{\Delta}_u(k) = \widehat{\Delta}_u(k)$ upon convergence:

$$\widehat{\overline{\Delta}}_u(k)\Bigg|_{(100),(101)} = \prod_{j \in \mathcal{W}_u} \Bigg[\prod_{i \in \mathcal{C}(u,s)} \Bigg[\left(\frac{\widetilde{\delta}_{i,j,1}(k;m)\left[\overline{\Delta}_u(k;m)\right]}{\widetilde{\delta}_{i,j,1}(k;m)} \right)^{\frac{\widetilde{\delta}_{i,j,1}(k;m)}{\overline{\Delta}_u(k;m)}}$$

$$\times \left(\frac{\widetilde{\delta}_{i,j,2}(k;m)\left[\overline{\Delta}_u(k;m)\right]}{\widetilde{\delta}_{i,j,2}(k;m)} \right)^{\frac{\widetilde{\delta}_{i,j,2}(k;m)}{\overline{\Delta}_u(k;m)}} \times \left(\frac{\widetilde{\delta}_{i,j,3}(k;m)\left[\overline{\Delta}_u(k;m)\right]}{\widetilde{\delta}_{i,j,3}(k;m)} \right)^{\frac{\widetilde{\delta}_{i,j,3}(k;m)}{\overline{\Delta}_u(k;m)}} \Bigg]$$

---

[3]Note that we replace $\overline{\Delta}_u(k)$ and $D_j(k)$ with their approximated versions in (48) and (51) only when they appear in the denominator of the terms in (a). Thus lower bounding these terms result in upper bounding (a) as desired.

$$
\times \prod_{h \in \widehat{\mathcal{W}}_u} \prod_{i \in \mathcal{C}(u,s)} \left[\left(\frac{\delta_{j,h,i,1}(k;m) \left[\overline{\Delta}_u(k;m) \right]}{\delta_{j,h,i,1}(k;m)} \right)^{\frac{\delta_{j,h,i,1}(k;m)}{\overline{\Delta}_u(k;m)}} \right.
$$

$$
\left. \left(\frac{\delta_{j,h,i,2}(k;m) \left[\overline{\Delta}_u(k;m) \right]}{\delta_{j,h,i,2}(k;m)} \right)^{\frac{\delta_{j,h,i,2}(k;m)}{\overline{\Delta}_u(k;m)}} \times \left(\frac{\delta_{j,h,i,3}(k;m) \left[\overline{\Delta}_u(k;m) \right]}{\delta_{j,h,i,3}(k;m)} \right)^{\frac{\delta_{j,h,i,3}(k;m)}{\overline{\Delta}_u(k;m)}} \right] \right]
$$

$$
= \prod_{j \in \mathcal{W}_u} \left[\prod_{i \in \mathcal{C}(u,s)} \left[\left(\left[\overline{\Delta}_u(k;m) \right] \right)^{\frac{\widetilde{\delta}_{i,j,1}(k;m)}{\overline{\Delta}_u(k;m)}} \right. \right.
$$

$$
\times \left(\left[\overline{\Delta}_u(k;m) \right] \right)^{\frac{\widetilde{\delta}_{i,j,2}(k;m)}{\overline{\Delta}_u(k;m)}} \times \left(\left[\overline{\Delta}_u(k;m) \right] \right)^{\frac{\widetilde{\delta}_{i,j,3}(k;m)}{\overline{\Delta}_u(k;m)}} \right]
$$

$$
\times \prod_{h \in \widehat{\mathcal{W}}_u} \prod_{i \in \mathcal{C}(u,s)} \left[\left(\left[\overline{\Delta}_u(k;m) \right] \right)^{\frac{\delta_{j,h,i,1}(k;m)}{\overline{\Delta}_u(k;m)}} \left(\left[\overline{\Delta}_u(k;m) \right] \right)^{\frac{\delta_{j,h,i,2}(k;m)}{\overline{\Delta}_u(k;m)}} \right. \tag{102}
$$

$$
\left. \left. \times \left(\left[\overline{\Delta}_u(k;m) \right] \right)^{\frac{\delta_{j,h,i,3}(k;m)}{\overline{\Delta}_u(k;m)}} \right] \right]
$$

$$
= \overline{\Delta}_u(k)^{\left(\frac{\sum_{j \in \mathcal{W}_u} \sum_{i \in \mathcal{C}(u,s)} \widetilde{\delta}_{i,h,1}(k;m) + \widetilde{\delta}_{i,h,2}(k;m) + \widetilde{\delta}_{i,h,3}(k;m)}{\overline{\Delta}_u(k;m)} \right)}
$$

$$
\times \overline{\Delta}_u(k)^{\left(\frac{\sum_{j \in \mathcal{W}_u} \sum_{h \in \widehat{\mathcal{W}}_u} \sum_{i \in \mathcal{C}(u,s)} \delta_{j,h,i,1}(k;m) + \delta_{j,h,i,2}(k;m) + \delta_{j,h,i,3}(k;m)}{\overline{\Delta}_u(k;m)} \right)}
$$

$$
= \overline{\Delta}_u(k)^{\left(\frac{\overline{\Delta}_u(k;m)}{\overline{\Delta}_u(k;m)} \right)} = \overline{\Delta}_u(k) \Big|_{(100),(101)}.
$$

Using a similar technique, it can be shown that $D_j(k) = \widehat{D}_j(k)$, upon convergence, and thus (i) holds. The proof for (ii) is similar, which is omitted for brevity.

3) *The KKT conditions of $\mathcal{P}$ should be satisfied after the series of approximations converges in problem $\widehat{\mathcal{P}}_m$:* Since the constraints (25)-(46) are common in problems $(\mathcal{P})$ and $(\widetilde{\mathcal{P}})$, for the approximated constraint we should have $\nabla(1 - \theta)((a) + \theta(b)) = \nabla((1 - \theta)(\widetilde{a}) + \theta(b))$, upon convergence ($\nabla$ denotes the gradient sign). Note that (a) involves the product between two ratios of posynomials (as can be seen from (18), one is $\frac{\Delta_j(k)}{\Delta_u(k)}$, where both the numerator and denominator are posynomial with respect to the optimization variables according to (37), and the other one is σ_j^{F}, which can be written as ratio of two posynomials according to (12)), where we approximate the denominator of each ratio via a monomial in $\widetilde{(a)}$. For compactness, let us define $(a) \triangleq \frac{A_1(\boldsymbol{x}) A_2(\boldsymbol{x})}{B_1(\boldsymbol{x}) B_2(\boldsymbol{x})}$, where $A_1(\boldsymbol{x})$ and $B_1(\boldsymbol{x})$ are posynomials corresponding to the numerator and denominator of $\frac{\Delta_j(k)}{\Delta_u(k)}$ (encompassing all the coefficients) and $A_2(\boldsymbol{x})$ and $B_2(\boldsymbol{x})$ are posynomials corresponding to the numerator and denominator of σ_j^{F} ($\boldsymbol{x}$ denotes the set of optimization variables). We provide the proof for the general case. In general, the posynomials in the denominators can be described as $B_1(\boldsymbol{x}) = \sum_{k=1}^{K_1} u_k^{(1)}(\boldsymbol{x})$ and $B_2(\boldsymbol{x}) = \sum_{k=1}^{K_2} u_k^{(2)}(\boldsymbol{x})$, where $u_k^{(1)}$-s and $u_k^{(2)}$-s are monomial functions. Accordingly, we can write $\widetilde{(a)}$ as $\widetilde{(a)} = \frac{A_1(\boldsymbol{x}) A_2(\boldsymbol{x})}{\widetilde{B}_1(\boldsymbol{x}) \widetilde{B}_2(\boldsymbol{x})}$, where $\widetilde{B}_1(\boldsymbol{x}) = \prod_{k=1}^{K_1} \left(\frac{u_k^{(1)}(\boldsymbol{x})}{a_k^{(1)}(\boldsymbol{y})} \right)^{a_k^{(1)}(\boldsymbol{y})}$ and $\widetilde{B}_2(\boldsymbol{x}) = \prod_{k=1}^{K_2} \left(\frac{u_k^{(2)}(\boldsymbol{x})}{a_k^{(2)}(\boldsymbol{y})} \right)^{a_k^{(2)}(\boldsymbol{y})}$ are the monomial approximation of $B_1(\boldsymbol{x})$ and $B_2(\boldsymbol{x})$ obtained according to (94) (equivalent to the condensations carried out in (48) and (51)). In the following, we show that the desired result holds for partial derivative with respect to an arbitrary element x_i considering $\boldsymbol{x} = [x_1, \cdots, x_i, \cdots, x_n]$ (note that upon convergence $\boldsymbol{x} = \boldsymbol{y}$ in (94), which in our problem translates to the equalities in (100),(101)):

$$
\frac{\partial \left(\frac{A_1(\boldsymbol{x}) A_2(\boldsymbol{x})}{\widetilde{B}_1(\boldsymbol{x}) \widetilde{B}_2(\boldsymbol{x})} \right)}{\partial x_i} \Bigg|_{\boldsymbol{x}=\boldsymbol{y}} = \frac{\frac{\partial A_1(\boldsymbol{x}) A_2(\boldsymbol{x})}{\partial x_i} \widetilde{B}_1(\boldsymbol{y}) \widetilde{B}_2(\boldsymbol{y}) - \frac{\partial \widetilde{B}_1(\boldsymbol{x}) \widetilde{B}_2(\boldsymbol{x})}{\partial x_i} A_1(\boldsymbol{y}) A_2(\boldsymbol{y})}{\left(\widetilde{B}_1(\boldsymbol{y}) \widetilde{B}_2(\boldsymbol{y}) \right)^2} \Bigg|_{\boldsymbol{x}=\boldsymbol{y}}
$$

$$
\stackrel{(a)}{=} \frac{\frac{\partial A_1(\boldsymbol{x}) A_2(\boldsymbol{x})}{\partial x_i} \Big|_{\boldsymbol{x}=\boldsymbol{y}} B_1(\boldsymbol{y}) B_2(\boldsymbol{y}) - \frac{\partial \prod_{k=1}^{K_1} \left(\frac{u_k^{(1)}(\boldsymbol{x})}{a_k^{(1)}(\boldsymbol{y})} \right)^{\alpha_k^{(1)}(\boldsymbol{y})}}{\partial x_i} \Big|_{\boldsymbol{x}=\boldsymbol{y}} B_2(\boldsymbol{y}) A_1(\boldsymbol{y}) A_2(\boldsymbol{y})}{(B_1(\boldsymbol{y}) B_2(\boldsymbol{y}))^2}
$$

$$
+\frac{-\dfrac{\partial \displaystyle\prod_{k=1}^{K_2}\left(\dfrac{u_k^{(2)}(\boldsymbol{x})}{a_k^{(2)}(\boldsymbol{y})}\right)^{a_k^{(2)}(\boldsymbol{y})}}{\partial x_i}\Bigg|_{\boldsymbol{x}=\boldsymbol{y}}B_1(\boldsymbol{y})A_1(\boldsymbol{y})A_2(\boldsymbol{y})}{\left(B_1(\boldsymbol{y})B_2(\boldsymbol{y})\right)^2}
$$

$$
=\frac{\dfrac{\partial A_1(\boldsymbol{x})A_2(\boldsymbol{x})}{\partial x_i}\Big|_{\boldsymbol{x}=\boldsymbol{y}}B_1(\boldsymbol{y})B_2(\boldsymbol{y})}{\left(B_1(\boldsymbol{y})B_2(\boldsymbol{y})\right)^2}
$$

$$
+\frac{-\displaystyle\sum_{n=1}^{K_1}\dfrac{\partial u_n^{(1)}(\boldsymbol{x})}{\partial x_i}\Big|_{\boldsymbol{x}=\boldsymbol{y}}\left(\dfrac{u_n^{(1)}(\boldsymbol{y})}{\alpha_n^{(1)}(\boldsymbol{y})}\right)^{\alpha_n^{(1)}(\boldsymbol{y})-1}\left(\displaystyle\prod_{k=1,k\neq n}^{K_1}\left(\dfrac{u_k^{(1)}(\boldsymbol{y})}{\alpha_k^{(1)}(\boldsymbol{y})}\right)^{\alpha_k^{(1)}(\boldsymbol{y})}\right)B_2(\boldsymbol{y})A_1(\boldsymbol{y})A_2(\boldsymbol{y})}{\left(B_1(\boldsymbol{y})B_2(\boldsymbol{y})\right)^2}
$$

$$
+\frac{-\displaystyle\sum_{n=1}^{K_2}\dfrac{\partial u_n^{(2)}(\boldsymbol{x})}{\partial x_i}\Big|_{\boldsymbol{x}=\boldsymbol{y}}\left(\dfrac{u_n^{(2)}(\boldsymbol{y})}{a_n^{(2)}(\boldsymbol{y})}\right)^{a_n^{(2)}(\boldsymbol{y})-1}\left(\displaystyle\prod_{k=1,k\neq n}^{K_2}\left(\dfrac{u_k^{(2)}(\boldsymbol{y})}{a_k^{(2)}(\boldsymbol{y})}\right)^{a_k^{(2)}(\boldsymbol{y})}\right)B_1(\boldsymbol{y})A_1(\boldsymbol{y})A_2(\boldsymbol{y})}{\left(B_1(\boldsymbol{y})B_2(\boldsymbol{y})\right)^2}
$$

$$
\overset{(b)}{=}\frac{\dfrac{\partial A_1(\boldsymbol{y})A_2(\boldsymbol{y})}{\partial x_i}\Big|_{\boldsymbol{x}=\boldsymbol{y}}B_1(\boldsymbol{y})B_2(\boldsymbol{y})}{\left(B_1(\boldsymbol{y})B_2(\boldsymbol{y})\right)^2}
$$

$$
+\frac{-\displaystyle\sum_{n=1}^{K_1}\dfrac{\partial u_n^{(1)}(\boldsymbol{x})}{\partial x_i}\Big|_{\boldsymbol{x}=\boldsymbol{y}}B_1(\boldsymbol{y})^{\alpha_n^{(1)}(\boldsymbol{y})-1}\left(B_1(\boldsymbol{y})^{\sum_{k=1,k\neq n}^{K_1}\alpha_k^{(1)}(\boldsymbol{y})}\right)B_2(\boldsymbol{y})A_1(\boldsymbol{y})A_2(\boldsymbol{y})}{\left(B_1(\boldsymbol{y})B_2(\boldsymbol{y})\right)^2}
$$

$$
+\frac{-\displaystyle\sum_{n=1}^{K_2}\dfrac{\partial u_n^{(2)}(\boldsymbol{x})}{\partial x_i}\Big|_{\boldsymbol{x}=\boldsymbol{y}}B_2(\boldsymbol{y})^{a_n^{(2)}(\boldsymbol{y})-1}\left(B_2(\boldsymbol{y})^{\sum_{k=1,k\neq n}^{K_2}a_k^{(2)}(\boldsymbol{y})}\right)B_1(\boldsymbol{y})A_1(\boldsymbol{y})A_2(\boldsymbol{y})}{\left(B_1(\boldsymbol{y})B_2(\boldsymbol{y})\right)^2}
$$

$$
\overset{(c)}{=}\frac{\dfrac{\partial A_1(\boldsymbol{y})A_2(\boldsymbol{y})}{\partial x_i}\Big|_{\boldsymbol{x}=\boldsymbol{y}}B_1(\boldsymbol{y})B_2(\boldsymbol{y})}{\left(B_1(\boldsymbol{y})B_2(\boldsymbol{y})\right)^2}
$$

$$
+\frac{-\displaystyle\sum_{n=1}^{K}\dfrac{\partial u_n^{(1)}(\boldsymbol{x})}{\partial x_i}\Big|_{\boldsymbol{x}=\boldsymbol{y}}B_2(\boldsymbol{y})A_1(\boldsymbol{y})A_2(\boldsymbol{y})-\displaystyle\sum_{n=1}^{K}\dfrac{\partial u_n^{(2)}(\boldsymbol{x})}{\partial x_i}\Big|_{\boldsymbol{x}=\boldsymbol{y}}B_1(\boldsymbol{y})A_1(\boldsymbol{y})A_2(\boldsymbol{y})}{\left(B_1(\boldsymbol{y})B_2(\boldsymbol{y})\right)^2}
$$

$$
=\frac{\dfrac{\partial A_1(\boldsymbol{x})A_2(\boldsymbol{x})}{\partial x_i}\Big|_{\boldsymbol{x}=\boldsymbol{y}}B_1(\boldsymbol{y})B_2(\boldsymbol{y})-\dfrac{\partial B_1(\boldsymbol{x})}{\partial x_i}\Big|_{\boldsymbol{x}=\boldsymbol{y}}A_1(\boldsymbol{y})A_2(\boldsymbol{y})B_2(\boldsymbol{y})-\dfrac{\partial B_2(\boldsymbol{x})}{\partial x_i}\Big|_{\boldsymbol{x}=\boldsymbol{y}}A_1(\boldsymbol{y})A_2(\boldsymbol{y})B_1(\boldsymbol{y})}{\left(B_1(\boldsymbol{y})B_2(\boldsymbol{y})\right)^2}
$$

$$
=\frac{\partial\left(\dfrac{A_1(\boldsymbol{x})A_2(\boldsymbol{x})}{B_1(\boldsymbol{x})B_2(\boldsymbol{x})}\right)}{\partial x_i}\Bigg|_{\boldsymbol{x}=\boldsymbol{y}}. \tag{103}
$$

In (a), we used the fact that $B_1(\boldsymbol{y}) = \widetilde{B}_1(\boldsymbol{y})$ and $B_2(\boldsymbol{y}) = \widetilde{B}_2(\boldsymbol{y})$ (this is the equality of monomial approximation with the original posynomial upon convergence that we showed in bullet point 2 above). In (b), we used the fact that (see (94)) $a_k^{(1)}(\boldsymbol{y}) = u_k^{(1)}(\boldsymbol{y})/B_1(\boldsymbol{y})$ and $a_k^{(2)}(\boldsymbol{y}) = u_k^{(2)}(\boldsymbol{y})/B_2(\boldsymbol{y})$, $\forall k$. Also, in (c) we use the fact that $\sum_{k=1}^{K_1}\alpha_k^{(1)}(\boldsymbol{y}) = 1$ and $\sum_{k=1}^{K_2}a_k^{(2)}(\boldsymbol{y}) = 1$. The proof for the rest of partial derivatives, and thus the gradient, is similar.

Verification of the three aforementioned bullet points results in the conclusion of the proof.

APPENDIX J
PROOF OF LEMMA 4

Given the previous computed gradient for the local model at the device cluster, i.e., $\nabla F_c(\mathbf{w}_c(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t_c^{\mathsf{V}}))$, the value of the local gradient given the outdated model $\mathbf{w}_c(t_c^{\mathsf{V}})$ for the recent data distribution $\widetilde{\mathcal{D}}_c(t)$ at time t can be expressed as follows:

$$
\begin{aligned}
\left\| \nabla F_c\left(\mathbf{w}_c(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t)\right) \right\|^2 =& \left\| \nabla F_c\left(\mathbf{w}_c(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t)\right) - \nabla F_c\left(\mathbf{w}_c(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t-1)\right) + \nabla F_c\left(\mathbf{w}_c(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t-1)\right) \right. \\
& - \nabla F_c\left(\mathbf{w}_c(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t-2)\right) + \nabla F_c\left(\mathbf{w}_c(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t-2)\right) - \cdots \\
& \left. - \nabla F_c(\mathbf{w}_c(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t_c^{\mathsf{V}})) + \nabla F_c(\mathbf{w}_c(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t_c^{\mathsf{V}})) \right\|^2 \\
\overset{(a)}{\leq} & \left(\left\| \nabla F_c\left(\mathbf{w}(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t)\right) - \nabla F_c\left(\mathbf{w}_c(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t-1)\right) \right\| + \left\| \nabla F_c\left(\mathbf{w}_c(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t-1)\right) \right. \right. \\
& \left. - \nabla F_c\left(\mathbf{w}_c(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t-2)\right) \right\| + \left\| \nabla F_c\left(\mathbf{w}_c(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t-2)\right) - \nabla F_c\left(\mathbf{w}_c(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t-3)\right) \right\| + \cdots + \\
& \left. \left\| \nabla F_c(\mathbf{w}_c(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t_c^{\mathsf{V}}+1)) - \nabla F_c(\mathbf{w}_c(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t_c^{\mathsf{V}})) \right\| + \left\| \nabla F_c(\mathbf{w}_c(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t_c^{\mathsf{V}})) \right\| \right)^2 \\
\overset{(b)}{\leq} & (t - t_c^{\mathsf{V}} + 1) \left\| \nabla F_c(\mathbf{w}_c(t_c^{\mathsf{V}})|\widetilde{\mathcal{D}}_c(t_c^{\mathsf{V}})) \right\|^2 + (t - t_c^{\mathsf{V}} + 1) \sum_{t'=t_c^{\mathsf{V}}+1}^{t} \Lambda_c(t'), \quad (104)
\end{aligned}
$$

where in (a) we used triangle inequality, and (b) is the result of Cauchy–Schwarz inequality (i.e, for any N numbers $A_1, \cdots, A_N$, we have $\left(\sum_{i=1}^{N} A_i\right)^2 \leq N \sum_{i=1}^{N}(A_i)^2$).