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.

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], [8], [9], [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], [8], [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 *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 Section 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-UV network 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], [16], [17], [18], [19], [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. 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 (Section II-A), UAV swarms, recharging stations, and access points (Section II-B). We also provide an overview of our entire methodology, including the macro- and micro-level design in Section II-C.

A. Device Clusters and Data Distributions

We consider a set of geo-distributed devices collected into C clusters, denoted by $C = \{C_1, \ldots, C_C\}$, based on geographic proximity. Hereafter, we will refer to an arbitrary cluster via $c$ for brevity. At time instance $t \in T \doteq \{1, 2, \ldots\}$, we denote $D_i(t)$ as the set of datapoints device $i \in c$ has collected at that time. Each $x \in D_i(t)$ is a data sample containing model features and (possibly) a target variable.

Motivated by the real-world applications in Section 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], [16], [17], [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 Section 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 $B_i$ of finite size, and that each device is capable of transmitting data to nearby UAVs to perform model training.

| Device Cluster, UAV Swarm, and Network | Data Processing/Offloading Optimization and Network Energy Consumption |
|---------------------------------------|---------------------------------------------------------------------|
| $c$ | $\rho_{ij}(t)$: Data transfer ratio from device $i$ to $j$ |
| $R$ | $g_{ij}(t)$: Adjusted CPU cycle frequency at UAV $j$ |
| $A$ | $\tau_{ij}(k)$: Time used by UAV $j$ to process data |
| $U_i$ | $M_i$: Number of bits per datapoint |
| $M$ | $\tilde{M}$: Number of bits used for model parameters |

**Machine Learning Notation**

- $F_j$: Meta loss function at UAV $j$
- $f_j$: ML loss over all data in $D_j$
- $w_{ij}(t)$: Model parameters at UAV $j$
- $\mu_{ij}(t)$: Outer mini-batch ratio for UAV $j$
- $\nu_{ij}(t)$: Inner mini-batch ratio for UAV $j$
- $t_i$: Starting time for the $i$-th sequence
- $\tilde{t}_i$: Local aggregation period for the $i$-th sequence
- $\tilde{r}_i$: Global aggregation period for the $i$-th sequence
- $k$: Global aggregation index

**Swarms Trajectory Optimization**

- $X(s)$: Swarm positions at $s$-th sequence
- $Z(s)$: Network state encoding at $s$-th sequence
- $V(s)$: Reward of the $s$-th sequence

| Summary of Key Notations for Devices, UAVs, Network Optimization, Machine Learning, and Swarm Trajectory Design |
|---------------------------------------------------------------------------------------------------------------|
| $\mathcal{C}$: Set of all device clusters |
| $\mathcal{R}$: Set of recharging stations |
| $\mathcal{A}$: Set of access points |
| $U_i$: Number of active swarms at $i$-th sequence |
| $M_i$: Set of active swarms at $i$-th sequence |
| $L_i$: Leader UAV of $i$-th swarm |
| $W_i$: Worker UAVs in the $i$-th swarm |
| $S_i$: Non-leader UAVs in $i$-th swarm |
| $\mathcal{D}_i(t)$: Dataset of $i$-th device |
| $T_i$: Duration of $i$-th sequence |
| $\mathcal{D}_i(t)$: Dataset of $j$-th UAV |
| $\mathcal{D}_i(t)$: Union of datasets at cluster $c$ |
| $\mathcal{D}_i(t)$: Max dataset size for UAV $j$ |
| $\tau_{ij}$: Local aggregation period for $i$-th sequence |
| $\tilde{r}_i$: Global aggregation period for $i$-th sequence |
| $k$: Global aggregation index |
| $X(s)$: Swarm positions at $s$-th sequence |
| $Z(s)$: Network state encoding at $s$-th sequence |
| $V(s)$: Reward of the $s$-th sequence |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
We consider $B_i$ 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 (Section 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 (Section II-B2). Also, to synchronize the ML model training and UAV orchestration, we consider a set of access points in the network (Section 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 $\mathcal{U} = \{U_1, \ldots, U_U\}$, 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 200 grams that have data storage capabilities to medium-sized fixed/rotatory wing UAVs that have more advanced 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 $\ell_u$ of each swarm $u$, with the set of leaders across swarms denoted by $\mathcal{L} = \bigcup_{u \in \mathcal{U}} \ell_u$; (ii) a set of workers $W_u$ in swarm $u$, with the set across swarms denoted by $\mathcal{W} = \bigcup_{u \in \mathcal{U}} W_u$, and (iii) a set of coordinators $\mathcal{W}_u$ in $u$, gathered via the set $\mathcal{W} = \bigcup_{u \in \mathcal{U}} \mathcal{W}_u$. For convenience, we refer to the workers and coordinators together as $\widehat{u} = W_u \cup \mathcal{W}_u = u \setminus \{\ell_u\}$. UAVs in $\widehat{u}$ collect data from nearby IoT devices, with the workers $W_u$ conducting model training based on their gathered data, and the coordinators $\mathcal{W}_u$ 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 $\ell_u$ 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 $C$, we will present a methodology to determine UAV swarm trajectories so that all device clusters can be supported in Section 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 $D_j(t)$ as the dataset UAV $j \in \widehat{u}$, with buffer size $D_j^D(t)$, obtains at time $t$. Workers $W_u$ 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 $\ell_u$ regarding their training results. On the other hand, coordinators $\mathcal{W}_u$ 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 $D_j(t), \forall j$, through our optimization in Section IV-A.

2) Recharging Stations: We consider a set of geodistributed 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-AP communication, i.e., among leader, coordinator, and worker UAVs; (ii) device-AP 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 in the supplementary material.

### C. Components of Our Methodology and Roadmap

We break down our methodology for conducting geodistributed 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 Section III. For (ii), we formulate a network-aware ML performance and swarm-wide energy consumption optimization problem, and then develop a distributed solver in Section IV. For (iii), we capture the temporal variation of data across 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 Section 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 Sections III, IV, and V, respectively.

### D. Metrics of Interest

The evaluation of our methodology requires several primary metrics: classification loss at the UAVs (Theorem 1), the
mismatch between the model performance at the UAVs vs. at the device clusters (Proposition 1), and energy consumption (Section IV). We will further demonstrate that our learning method provides substantial improvements compared to baselines in terms of convergence rate (Section VI-A), network resource savings (Section VI-B), and trajectory design efficiency (Section 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 (Section III-A), and then present the HN-PFL algorithm (Section III-B). Finally, we theoretically analyze the convergence of our distributed ML method (Section 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 \( U \) 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, \( s = 1, 2, \ldots \), by \( t_s \in T \), its scheduled interval as \( T_s = \{ t_s, \ldots, t_s + T_s - 1 \} \), and the active swarms for the \( s \)-th sequence as \( U(s) \subseteq U \). Defining the active swarms \( U(s) \) 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 \( T_s \).

At each time \( t \in T_s \), each worker UAV \( j \in W_u, u \in U(s) \) conducts a local model update. This consists of computing...
its next ML model parameter vector $w_j(t+1)$ using a meta-gradient update [30] defined as:

$$w_j(t+1) = w_j(t) - \eta_j \nabla F_j(w_j(t)), \quad t + 1 \in T_s,$$  

(1)

where $\eta_j > 0$ is the meta-update step-size, and $\nabla F_j(w_j(t))$ is the mini-batch approximation of the meta-gradient $\nabla F_j(w_j(t))$. $\nabla F_j(w_j(t))$ is the gradient of the meta-function $F_j$, defined as the loss of the gradient descent procedure:

$$F_j(w_j(t)) = f_j\left(\frac{w_j(t) - \eta_j \nabla F_j(w_j(t))}{\eta_j}\right),$$  

(2)

where $\eta_j > 0$ is the step size for gradient descent and $f_j$ is the local loss function over the local dataset $D_j(t)$ at UAV $j$:

$$f_j(w_j(t)) = \sum_{x \in D_j(t)} \tilde{f}(w_j(t); x),$$  

(3)

and $\tilde{f}(w_j(t); x)$ 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 $\nabla F_j$ by mini-batch methods:

$$\nabla F_j(w_j(t)) = \nabla \tilde{f}_j\left(\frac{w_j(t) - \eta_j \nabla \tilde{f}_j(w_j(t); D_j,1(t))}{\eta_j} \left| D_j,2(t) \right. \right),$$  

(4)

where $\tilde{f}_j$ is the mini-batch loss defined similarly to (3) over a specific data batch, $\nabla$ is the Hessian operator, and $D_j,1(t), D_j,2(t), D_j,3(t)$ are three independent mini-batches sampled with replacement from $D_j(t)$. 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., $D_j,1(t) = \alpha_j,1(t) D_j(t), i = 1, 2, 3$, and the total ratio of data at time $t$ used for processing by worker $j \in W_u$, $u \in U(s)$ 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 $T_s$. We conduct local aggregations with period of $\tau_s^l$ (i.e., $\tau_s^l$ local meta-gradient updates prior to each local aggregation), and global aggregations with period of $\tau_s^g$, defined such that $T_s = \tau_s^l + \tau_s^g$, where $K_s^l$ is the total number of local aggregations, and $K_s^g$ is the total number of global aggregations conducted in $T_s$. Using $k$ for the local aggregation index, $\tau_s^l(k) = t_s + k\tau_s^l$ is the time of the $k$-th local aggregation in $T_s$. Using $k'$ as the global aggregation index, $\tau_s^g(k') = t_s + k'\tau_s^g$ will denote the time of the $k'$-th global aggregation in $T_s$.

Within a swarm, workers’ datasets will be controlled through a device-to-UAV offloading optimization procedure in Section 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 $t = \tau_s^l(k)$, HN–PFL performs a weighted average at each leader $\ell_u$, $\forall u$:

$$\overline{w}_u(t) = \frac{\sum_{j \in W_u} w_j(t) \sum_{t' = \tau_s^l(k-1)+1}^{\tau_s^l(k)+1} \alpha_j(t') D_j(t')}{\sum_{j \in W_u} \sum_{t' = \tau_s^l(k-1)+1}^{\tau_s^l(k)+1} \alpha_j(t') D_j(t')}.$$  

(5)

Leader $\ell_u$ then broadcasts $\overline{w}_u(\tau_s^l(k'))$ to all worker UAVs $W_u$. This completes our swarm-wide aggregation, and we define the swarm-wide/local meta-function $\overline{F}_u$ for $t = \tau_s^l(k)$ as follows:

$$\overline{F}_u(\overline{w}_u(t)) = \frac{\sum_{j \in W_u} \overline{F}_j(\overline{w}_u(t)) \sum_{t' = \tau_s^l(k-1)+1}^{\tau_s^l(k)+1} \alpha_j(t') D_j(t')}{\sum_{j \in W_u} \sum_{t' = \tau_s^l(k-1)+1}^{\tau_s^l(k)+1} \alpha_j(t') D_j(t')}.$$  

(6)

After $\tau_s^g$ swarm-wide aggregations at the active swarms, the leaders $\ell_u, u \in U(s)$, travel to their nearest AP, and transmit their swarm-wide parameters to the core network. At the $k'$-th global aggregation, when $t = \tau_s^g(k')$, the core network determines the global model parameters as:

$$w(t) = \frac{1}{U(s)} \sum_{u \in U(s)} w_u(t),$$  

(7)

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

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

(8)

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. Equation (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 $A$ then broadcast $w(\tau_s^g(k'))$ 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 $U(s)$ 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 $u \in U(s) \cup U(s+1)$, leader $\ell_u$ will receive (i) final global parameters $w(t_s + T_s - 1)$, and (ii) trajectory decisions from the APs. Swarms then travel to their next cluster (for swarms $u \in U(s) \setminus U(s+1)$, this
would be a recharging station). Upon arrival, swarm leaders $l_u, u \in U(s+1)$, 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 Section V-A). We next obtain the convergence bound of HN-PFL, which will be employed in the optimization formulation in Section 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 $F_u (w)$ at $t \in T_u$ as:

$$F_u (w(t)) = \sum_{j \in W_u} \frac{\Delta_j (k)}{\Delta_k (u)} f_j (w(t)), \quad (9)$$

where $f_j$ is defined as in (3), $k$ is the most recent local aggregation index, $\Delta_j (k) = \sum_{t=t_j (k-1)+1} t_j (k) a_j (t) D_j (t)$, and $\Delta_k (u) = \sum_{j \in W_u} \Delta_j (k)$. Also, we define the global average loss function $F (w)$ at time $t \in T_u$ as:

$$F (w(t)) = \frac{1}{|U|} \sum_{u \in U} F_u (w(t)). \quad (10)$$

While (5), (6), (9) are only realized by HN-PFL at a local aggregation, $t = t_j (k')$, and (7), (8), (10) are only realized at a global aggregation, $t = t^G (k')$, defining them for each $t \in T_u$ 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 $j \in W_u, \forall u \in U$:

1) $f_j$ is bounded below: $f_j (w) > -\infty, \forall w$.

2) The gradient of $f_j$ is $\mu^G$-Lipschitz and bounded by $B_j$: (i) $\| \nabla f_j (w) - \nabla f_j (w') \| \leq \mu^G \| w - w' \|, \forall w, w'$, (ii) $\| \nabla f_j (w) \| \leq B_j$.

3) $f_j$ is twice continuously differentiable, and the Hessian of $f_j$ is $\mu^H$-Lipschitz continuous: $\| \nabla^2 f_j (w) - \nabla^2 f_j (w') \| \leq \mu^H \| w - w' \|, \forall w, w'$.

4) Bounded local data variability: $\mathbb{E} [\| \nabla f_j (x, w) - \nabla f_j (w) \| ^2 ] \leq \sigma^F_j$ and $\mathbb{E} [\| \nabla^2 f_j (x, w) - \nabla^2 f_j (w) \| ^2 ] \leq \sigma^H_j, \forall w, x$.

5) The variance of the gradient and Hessian of $f_j (w)$ are bounded: $\sum_{j \in W_u} \frac{\Delta_j (k)}{\Delta_k (u)} \| \nabla f_j (w) - \nabla f_j (w) \| ^2 \leq \gamma^G$, $\sum_{j \in W_u} \frac{\Delta_j (k)}{\Delta_k (u)} \| \nabla^2 f_j (w) - \nabla^2 f_j (w) \| ^2 \leq \gamma^H, \forall w, k$.

6) The variance of the gradient and Hessian of $F (w)$ are bounded: $\frac{1}{|U|} \sum_{u \in U} \| \nabla f_j (w) - \nabla f_j (w) \| ^2 \leq \gamma^G$, $\frac{1}{|U|} \sum_{u \in U} \| \nabla^2 f_j (w) - \nabla^2 f_j (w) \| ^2 \leq \gamma^H, \forall w, k$. 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 $w$, let $f^D_c (w) = \sum_{x \in D_c} f_c (x; w) / D_c (t)$, where $D_c (t) \triangleq \cup_{j \in D_c} D_j (t)$, denote the local loss at time $t$. We assume that (i) the gradient of the loss function is $\mu^G_c$-Lipschitz continuous, (ii) the Hessian is $\mu^H_c$-Lipschitz continuous, (iii) $\mathbb{E} [\| \nabla f_c (x, w) - \nabla f^D_c (w) \| ^2 ] \leq \sigma^G_c \forall x, w$, and (iv) $\mathbb{E} [\| \nabla^2 f_c (x, w) - \nabla^2 f^D_c (w) \| ^2 ] \leq \sigma^H_c \forall x, w$.

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 $B = \max_j \{ B_j \}$, $\mu^G = \max_j \{ \mu^G_j \}$, and $\mu^H = \max_j \{ \mu^H_j \}$.

Our main result in this section will be the convergence behavior of the global meta-function $F$ in HN-PFL (Theorem 1). To obtain this, we first derive bounds on the expected error of the mini-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 $T_u$, the expected error of the mini-batch approximation $\nabla F_j (w^u (t))$ of the true meta-gradient $\nabla F_j (w^u (t))$ at each worker UAV $j \in W_u, \forall u \in U(s)$ satisfies:

$$\mathbb{E} [\| \nabla F_j (w^u (t)) - \nabla F_j (w^u (t)) \| ^2 ] \leq \sigma^F_j (t), \quad (11)$$

where

$$\sigma^F_j (t) = \frac{3\sigma^G_j \sigma^H_j}{\alpha_j (s)} D_j (t) \left[ B^2 + \frac{\sigma^G_j (\alpha_j (t) + (\mu^G_j \eta_j)^2 \alpha_j (2))}{\alpha_j (t) \alpha_j (2) D_j (t)} \right] + \frac{12\sigma^H_j (\alpha_j (t) + (\mu^G_j \eta_j)^2 \alpha_j (2))}{\alpha_j (t) \alpha_j (2) D_j (t)} \quad (12)$$

Proof: See Appendix C in the supplementary materials.

We make a few observations from Lemma 1 regarding the mini-batch ratios $\alpha_j (1), \alpha_j (2)$, and $\alpha_j (3)$. Intuitively, if any of these are zero, 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, $\alpha_j (3)$ weighs the Hessian data variability $\sigma^H_j$ in (a), while $\alpha_j (1)$ and $\alpha_j (2)$ scale the gradient data variability $\sigma^G_j$ 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 \( \alpha_{j,s}(t) \) also weights the impact of \( \sigma^G_u \). 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 Section IV-A.

Lemma 2 (Intra- and Inter-Swarm Meta-Gradient Variability): During each training interval \( T_s \), the intra-swarm meta-gradient variability at each UAV swarm \( u \in U(s) \) obeys the following upper bound at local aggregation \( k \):

\[
\sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k)}{\Delta_u(k)} \left\| \nabla F_j(w) - \nabla F_u(w) \right\|^2 \leq \gamma_u^F, \quad (13)
\]

where \( \gamma_u^F = 3B^2 \eta_i^2 \gamma^H_s + 192\eta_i^G \). Also, the variability of inter-swarm meta-gradients is upper bounded as follows:

\[
\frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \left\| \nabla F_u(w) - \nabla F(w) \right\|^2 \leq \gamma_s^F, \quad (14)
\]

where \( \gamma_s^F = 3B^2 \eta_i^2 \gamma^H_s + 192\eta_i^G \).

Proof: See Appendix D in the supplementary materials.

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 \( T_s \), the variance in model parameters across the active worker UAVs in the network at the \( k \)-th global aggregation, i.e., \( t = t^G(k') \), satisfies the following upper bound:

\[
\mathbb{E} \left( \left\| \frac{1}{U(s)} \sum_{u \in \mathcal{U}(s)} \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k'^G)}{\Delta_u(k'^G)} (w_j(t) - w(t)) \right\|^2 \right) \leq \Upsilon(t),
\]

(15)

where \( \Upsilon(t) \) is defined in (16), shown at the bottom of the page, with \( \mu^F \triangleq 4\mu^G + \eta_i \mu^H \), and \( \sigma^F_u(t) = \sum_{j \in \mathcal{W}_u} \frac{\Delta_j(k'^G)}{\Delta_u(k'^G)} \sigma^F_j(t) \).

Proof: See Appendix E in the supplementary materials.

Considering the two terms in (16), term (a) captures the impact of the local aggregation frequency \( \tau^L_s \) while term (b) captures the nested impact of the local-global aggregation frequency \( \tau^G_s \). It can be seen that, to reduce the variance of ML models \( w_j \) 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 \( \tau^L_s, \tau^G_s \) need to be reduced, i.e., more frequent aggregations. Proposition 1 suggests that the improvement is exponential. However, global aggregations 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 \( \tau^L_s \) and \( \tau^G_s \) (Section IV), and then jointly optimizing \( \tau^L_s \) and \( \tau^G_s \) and UAV trajectories under optimal UAV orchestration (Section 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 \( \gamma_u^F \), the inter-swarm variability \( \gamma_s^F \), and the mini-batch approximation error (the \( \sigma^F_u(t) \) and \( \sigma^F_s(t) \) 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 \( T_s \), if \( \eta_i < \frac{1}{68} \), we have the following upper bound on the expected cumulative average magnitude of the global meta-gradient across the active UAV swarms:

\[
\frac{1}{T_s} \sum_{t=t_0-1}^{t_s} \mathbb{E} \left( \left\| \nabla F(w(t)) \right\|^2 \right) \leq \Xi(s), \quad (17)
\]

where \( \Xi(s) \) is given in (18), shown at the bottom of the next page.

Proof: See Appendix F in the supplementary materials.

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 \( \Delta_j \) and \( \Delta_u \)), the initial performance of the ML model for the \( s \)-th training sequence (embedded in \( F^U(w(t_0-1)) \)), the data variability
(embedded in $\sigma^F$, $\sigma^u$, and $\gamma$), the gradient/Hessian characteristics (embedded in $\mu^F$, $\gamma^F$, and $\gamma^F$), the local/global aggregation periods (through the nested sums), and the inner and outer step-sizes ($\eta_1$, $\eta_2$).

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, we refer to $D_j(t)$ as $D_j(k)$.

Lemma 3 (Meta-Gradient Mismatch Between Clusters and Swarms): Let $F^D_s$ denote the meta-function defined based on $f^D_c$ in Assumption 2, $C(s)$ denote the set of actively trained device clusters for training sequence $s$, and $T^D(w(t)) = \frac{1}{|C(s)|} \sum_{c \in C(s)} F^D_c(w(t)), \ t \in T_s$, be the average meta-function for all actively trained device clusters for sequence $s$ for a given parameter $w(t)$ 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:

$$E \left[ \left\| \nabla F^s(w(t)) - \nabla F^D(w(t)) \right\|^2 \right] \leq \Xi(k), \ t = r_s^k(k), \quad (19)$$

where $\Xi(k)$ is given by:

$$\Xi(k) \triangleq \frac{1}{U(s)} \sum_{u \in U(s)} \sum_{j \in W_u} \Delta_j \frac{3\eta_2^2 \sigma^F_{G(s,u)}}{D_j(k)} \times \left[ b^2 + \sigma^H_{G(s,u)} (1 + \mu^2 \sigma^F_{G(s,u)}) \right] + \gamma^F K + 12 \sigma^H_{G(s,u)} (1 + \mu^2 \sigma^F_{G(s,u)}) \right], \quad (20)$$

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

Proof: See Appendix G in the supplementary materials.

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/OFFLOADING OPTIMIZATION FOR HN–PFL

Given our HN–PFL algorithm and its convergence behavior in Section 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 in Section V. Our formulation will account for the interplay between the overall energy cost of the system and the developed HN–PFL performance metrics from Section III via a control optimization of key network parameters (Section 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 (Section IV-B).

A. Data Processing/Offloading Configuration

1) Offloading and Processing Models: At each training sequence $s$, each active UAV swarm $u \in U(s)$ is located above a device cluster denoted by $C(u, s) \in C$. Let $A(u, s) \in A$ denote the nearest AP to UAV swarm $u$, which leader $\ell_u$ will periodically visit during the model training to conduct global model aggregation. Also, let $\psi_F^u$ denote the energy consumed per unit time by worker/coordinator UAV $j \in W_u$ for flying/hovering, and $\psi_u$ that of leader UAV $\ell_u$. We denote the geographical distance between leader $\ell_u$ and its nearest AP as $d(\ell_u, A)$, and denote the move energy consumption of the leader UAV per unit distance as $\psi_u^M$.

Device-UAV and UAV-UAV data transfers are carried out at the beginning of local aggregation rounds, i.e., when $t = t_s^k(k)$. The data received by the worker UAVs is used throughout the local aggregation period. Upon engaging in data transmission, each device $i \in C(u, s)$ samples data points uniformly at random from its local buffer and transmits them to the UAVs. We let $\rho_{i,j}(k)$ denote the fraction of datapoints in the local dataset $D_i(k)$ of the device that is transmitted to worker/coordinator UAV $j \in W_u$. 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 $\gamma_{h,j}(k) \in [0, 1]$ denote the fraction of the local dataset at the coordinator UAV $h \in \hat{W}_u$ that is forwarded to another UAV $j \in W_u$. Each worker UAV $j \in W_u$ processes data with CPU frequency $g_j(k) \in [g_j^{\min}, g_j^{\max}]$.

For sequence $s$, we denote the transmit powers of device $i$, UAV $j$, and leader $\ell_u$ by $P_i(s)$, $P_j(s)$, and $P_{\ell_u}(s)$, respectively.
respectively. Through transmissions, either data or model parameters are transferred. We denote the number of bits used to represent one data point as \( M \), 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 \( b \) (\( a, b \in U \cup C \)), the path-loss model for the LoS link between two nodes \( a \) and \( b \) is then given by [32]:

\[
L_{a,b}^{\text{A2A}} = \eta_{\text{LoS}} \left( \mu_{\text{Tx}} d(a,b) \right)^{\alpha_{\text{PL}}},
\]

(21)

where \( \eta_{\text{LoS}} > 1 \) denotes the excessive path loss factor for the LoS link, \( \alpha_{\text{PL}} \) is the path-loss exponent, and \( \mu_{\text{ Tx}} = 4\pi f_{\text{Tx}} / c_{\text{light}} \) with \( c_{\text{light}} \) denoting the speed of light and \( f_{\text{Tx}} \) 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]:

\[
P_{a,b}^{\text{LoS}} = \left( 1 + \psi_{\text{LoS}} \exp(-\beta_{\text{LoS}} (d_{a,b})) \right)^{-1},
\]

(22)

where \( \psi_{\text{LoS}} \) and \( \beta_{\text{LoS}} \) 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 \( P_{a,b}^{\text{NLoS}} = 1 - P_{a,b}^{\text{LoS}} \). With this, the path-loss of an A2G/G2A link from node \( a \) to node \( b \) can be obtained as:

\[
L_{a,b}^{\text{A2G}} = \left( \mu_{\text{Tx}} d(a,b) \right)^{\alpha_{\text{PL}}} \left[ P_{a,b}^{\text{LoS}} \times \eta_{\text{LoS}} + P_{a,b}^{\text{NLoS}} \times \eta_{\text{NLoS}} \right],
\]

(22)

where \( \eta_{\text{NLoS}} > \eta_{\text{LoS}} \) denotes the excessive path loss factor for the NLoS link. Finally, the data-rate between two nodes \( a \) and \( b \) is given by:

\[
R_{a,b} = B_{a,b} \log_2 \left( 1 + \frac{P_a}{L_{a,b} \sigma^2} \right), \quad a, b \in U \cup C,
\]

(23)

where \( B_{a,b} \) denotes the bandwidth, \( \sigma^2 = N_0 B_{a,b} \) 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 \( \zeta_j^G(k) \), and the time used for data processing as \( \zeta_j^P(k) \). We also define \( \zeta_j^\text{Local} \) 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 \( s \):

\[
\begin{align*}
\min_{\theta, \Xi(s)} & \quad (1 - \theta) \left( \theta_i \Xi(s) + \theta_0 \sum_{k=1}^{K_0} \hat{E}_k \right) \\
\text{s.t.} & \quad E_j^P(k) = \frac{\tau_j \alpha_j c_j}{2} \left( \alpha_j(k)D_j(k) \sigma_j^2 \right), \quad j \in W_u, \forall u \\
& \quad E_j^\text{Tx}(k) = \sum_{i \in C} \rho_{j,i}(k) \tilde{D}_j(k) \tilde{M}_i / R_{j,i}(k), \quad j \in W_u, \forall u \\
& \quad E_j^\text{W} = K_j^W P_j(s) M / R_{j,W}(k), \quad j \in W_u, \forall u \\
& \quad E_j^{\text{FL}} = \rho_j(s) \mu_j(s) \max \left( M / R_{j,W}, s \right), \quad j \in W_u, \forall u \\
& \quad E_j^{\text{L}} = T_s \psi_j f_j, \quad j \in \tilde{U}, \forall \tilde{u} \\
& \quad E_j^F = T_s \psi_j f_j + 2s^2 + 2K_j^G \mu_j^G d_i(A_j), \quad \forall \tilde{u} \\
& \quad \sum_{j \in C} \rho_{j,i}(k) \leq 1, i \in C(u, s), \forall u \\
& \quad \rho_{j,i}(k) \leq 1, i \in C(u, s), \forall u \\
& \quad D_j(k) = \sum_{i \in C(u, s)} \rho_{j,i}(k) \tilde{D}_j(k), \quad j \in W_u, \forall u \\
& \quad \rho_{j,i} \leq \zeta_j^\text{Local}, \quad j \in u, \forall u \\
& \quad \zeta_j^P(k) + \zeta_j^G(k) \leq \zeta_j^\text{Local}, \quad j \in u, \forall u \\
& \quad \zeta_j^P(k) = \sum_{i \in C(u, s)} \rho_{j,i}(k) \tilde{D}_j(k) \tilde{M}_i / R_{j,i}(k) + \sum_{i \in \tilde{W}_u} \rho_{j,i}(k) \tilde{D}_j(k) \tilde{M}_i / R_{j,i}(k), \quad j \in \tilde{u}, \forall \tilde{u} \\
& \quad \zeta_j^G(k) = \tau_j \alpha_j c_j \alpha_j(k)D_j(k) / g_j \left( k \right), \quad j \in W_u, \forall u \\
& \quad \zeta_j^P \leq g_j \left( k \right), \quad j \in W_u, \forall u \\
& \quad 0 \leq \rho_{j,i} \leq 1, i \in C(u, s), \forall u \\
& \quad 0 \leq \rho_{j,i} \leq 1, h \in \tilde{W}_u, j \in W_u \\
& \quad \alpha_j = \alpha_j(k) + \alpha_j(k) + \alpha_j(k), \quad j \in W_u, \forall u \\
& \quad \zeta_j^G(k) \leq \zeta_j^\text{Local}, \quad \forall \tilde{u}, \forall \tilde{u} \\
& \quad 0 < \alpha_j(k), \alpha_j(k), \alpha_j(k) \leq 1, j \in W_u, \forall u
\end{align*}
\]

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 \( \Xi \)) and the mismatch to the ML performance at the device
cluster (through Ξ), which we quantified in Theorem 1 and Lemma 3. We weight the importance of the two terms in (a) with normalized positive coefficients θ₁ and θ₂. The coefficient θ ∈ [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 ρ = {ρ(i,j)}i∈C(u,s),j∈Ω∈U(s), the UAV-to-UAV data transfer configurations η = {η(h,j)}h∈V_u,j∈Ω∈U(s), the mini-batch ratios α = {α_j1,α_j2,α_j3}j∈Ω∈U(s), and the UAV CPU frequency cycles g = {g(j)}j∈Ω∈U(s).

For the energy objective terms, E_j^P(k) in (25) denotes the energy consumption used for processing data during each meta-gradient update at worker UAV j ∈ V_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_j^{Tx,C}(k) is the energy used for data transmission (Tx) from device i in cluster C(u,s) to the UAVs. In (27), we define E_j^{Tx,U}(k) as the energy used for data transmissions from coordinator UAV j ∈ V_u to other UAVs. In (28), E_j^{Tx,W} denotes the total energy consumed for ML model parameter transmission from worker UAV j ∈ V_u to its associated leader, which occurs K_j^L times during the training sequence. In (29), E_j^{ε,C} captures the energy consumption used for parameter transmission through broadcasting from leader u to worker UAVs j ∈ V_u. In (30), E_j^{ε,U} represents the total energy used for flying/hovering by UAV j ∈ V_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 rotatory wing UAVs. Similarly, in (31), E_j^{ε,L} captures the energy consumption for flying by leaders 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^{Ba}(s) − E_j^{Th}(s), where E_j^{Ba}(s) represents the battery energy at j at the start of the s-th training sequence and E_j^{Th}(s) encompasses both (i) surplus idle energy needed for extra hovering time caused by potential asynchronicity 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) imposes 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 ∈ C(u,s) and each coordinator UAV h ∈ V_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 ξ_j^{G}(k) and local data processing ξ_j^{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 ξ^{Local}(k). 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 \( E_j^P(n) \) in (25), there are multiplications between all four variable types: \( \alpha, \gamma, \rho, \) and \( \varphi \) (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 in the supplementary materials. Although the objective in \( \mathcal{P} \) is composed of multiplications between the variables, it does not follow the format of GP: the expressions of \( \Xi \) and \( \bar{\Xi} \) consist of terms that are in the format of ratio of two posynomials, which are not posynomials (e.g., \( \frac{\Delta_j}{\sum_i \gamma_i} \) and \( \sigma_j^F \), each of which contain arithmetic relationships of \( \rho \) and \( \varphi \) 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 \( \Xi \) and \( \bar{\Xi} \) (via approximating \( \Delta_u \) 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 \( \Xi \) and \( \bar{\Xi} \)) can be written as a sum of separable functions with respect to the UAV swarm index. In practice, however, \( \gamma^F \) (defined in Lemma 2) and \( \mu^F \) (defined in Proposition 1) cannot be locally and independently computed by UAV swarms \( u \in U(s) \), as it depends on the maximum over all worker UAVs in the network. We can approximate \( \gamma^F \) and \( \mu^F \) 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 \( \gamma^F \) and \( \mu^F \) at the swarm leaders, the problem can then be decomposed among the local aggregation instances \( k \in \{1, \ldots, K_u\} \) and solved distributively by each swarm \( u \in U(s) \) 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 \( \Delta_u \) and \( D_j \) in our case, using the arithmetic-geometric mean inequality (see Lemma 5 in Appendix H in the supplementary materials). 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 \( x[m] = \{ q[m], \rho[m], a[m], g[m] \} \). \( \overline{x}_n(k; m), \delta_{i,j,n}(k; m), \) and \( \delta_{i,j,n}(k; m), n \in \{1, 2, 3\} \) in (48) and \( D_j(k; m), \lambda_{h,i,j}(k; m), \) and \( \overline{\lambda}_{i,j}(k; m) \) in (51) are scalar values obtained by substituting the solution at iteration \( m \) (i.e., \( x[m] \)) in the corresponding expressions in (47, 49, 50), and (52), shown at the bottom of the page, respectively. It can be verified that the approximations in (48) and (51), shown at the bottom of the page, are in fact the best local monomial approximations to their respective posynomials near the fixed point \( x[m] \) 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 \( \mathcal{P}_m \) at iteration \( m \). In \( \mathcal{P}_m \), each constraint is an inequality on posynomials, and the objective function is a sum of posynomials, admitting the GP format:

\[
(\mathcal{P}_m): \min_{x \in \mathcal{P}} g(a) + \theta(b) \text{ s.t. } (25) - (46),
\]

where term \( \sum_{x \in \mathcal{P}} m \) 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 \( a \), and the constraints of \( \mathcal{P} \) 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 \( \mathcal{P}_m \) that converge to a point \( x^* \) satisfying the Karush-Kuhn-Tucker (KKT) conditions of \( \mathcal{P} \).

**Proof:** See Appendix I in the supplementary materials.

## V. SWARM TRAJECTORY AND TEMPORAL DESIGN

Our developed HN-PFL methodology (Section I) and ML performance and energy optimization (Section 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 Sections III, IV, and V, complete our methodology.

Upon completion of a training sequence using our HN-PFL and performance optimization components (Sections 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,
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., $\mu^2$, $\mu^4$, and $\mu^6$), estimated data variability (i.e., $\sigma^2$ and $\sigma^4$), 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 $k^j \in \{1, \ldots, K^2\}$ do
3    // Procedure at the core network
4    for $c \in C$ do
5       if Procedure conducted across the UAV swarms in parallel using their leader
6          for $k \in \{\tau^c - 1\} + x^G + 1, \ldots, k^c + x^G\}$ do
7              Initialize the iteration count $m = 0$.
8              Choose an initial feasible point $x[0] = [\rho[0], \sigma[0], \alpha[0], \theta[0]]$ for $\mathcal{P}_0$ at $k$.
9              while convergence criterion is not met OR $m = 0$ do
10                 Obtain the scalar values for $\mathcal{S}_w(k, m)$, $\tilde{\mathcal{S}}_w(k, m)$, $\lambda^c_{\tau^c} \tilde{\lambda^c}_{\tau^c} (k, m)$, $D_j(k, m)$, and $\Lambda_{\tau^c} (k, m)$; $\tilde{\Lambda}_{\tau^c} (k, m)$ by substituting $x[m]$ 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 $\mathcal{P}_m$.
12                 Apply the logarithmic change of variables and take the log from the constraints of $\mathcal{P}_m$ and convert it to a convex optimization problem (as in (93) in Appendix H in the supplementary materials).
13                 Solve the resulting convex optimization problem to obtain $x[m + 1]$ (e.g., using CVX [38]), and set $m = m + 1$.
14                 Choose the obtained point as $x^* = \mathcal{S}_w[k, x^*] = [\rho^*, \sigma^*, \alpha^*, \theta^*]$. The leader broadcasts the solution $x^*$ 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.

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 $\Lambda_c(t) \in \mathbb{R}_{+, \text{inf}}$, which captures the maximum variation of the gradient for any given model parameter across two consecutive time instances; mathematically:

$$\|\nabla F_c(w|\tilde{D}_c(t)) - \nabla F_c(w|D_c(t-1))\|^2 \leq \Lambda_c(t), \forall w.$$  \hspace{1cm} (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 Model Performance via Model Drift):** Let $t^v_c$ denote a time instance when device cluster $c$ was visited by a UAV swarm, $w_c(t^v_c)$ denote the corresponding local model, and $\nabla F_c(w_c(t^v_c)|D_c(t^v_c))$ denote the local gradient. Consider time instance $t \geq t^v_c + 1$, where during the time interval $[t^v_c + 1, t]$ cluster $c$ remains unvisited by UAV swarms. Given the updated data at device cluster $c$, i.e., $\tilde{D}_c(t)$, the local meta gradient at the device cluster for the outdated local model is upper bounded as:

$$\|\nabla F_c(w_c(t^v_c)|D_c(t^v_c))\|^2 \leq \left( t - t^v_c + 1 \right) \left( t - t^v_c + 1 \right) \sum_{t' = t^v_c + 1}^{t^v_c + 1} \Lambda_c(t').$$  \hspace{1cm} (54)

where $\Lambda_c(t)$ is defined in (53). Subsequently, assuming an upper bound on the model drift at the device cluster $\Lambda_c(t') \leq \Lambda_c$, $t' \in [t^v_c + 1, t]$, we have: $\|\nabla F_c(w_c(t^v_c)|D_c(t^v_c))\|^2 \leq (t - t^v_c + 1) \left( t - t^v_c + 1 \right) \sum_{t' = t^v_c + 1}^{t^v_c + 1} \Lambda_c(t')$.  \hspace{1cm} (54)

**Proof:** See Appendix J in the supplementary materials.

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 $c$, $E_{\tau^c}(s)$ 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 Section 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: 
\[
G_c(s) = \| \nabla F_c(w_c(t_{s} + T_s)) D_c(t_{s} + T_s) \|^2 \text{ measured at the end of the training sequence; however, if it is not visited, i.e., } c \in \overline{\mathcal{C}}(s), \text{ then } G_c(s) = \| \nabla F_c(w_c(t_s)) D_c(t) \|^2, \text{ which is computed via the bound in } (54). \]

Also, we let \(E^M(s)\) 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:

\[
\begin{aligned}
\min_{\{X(s)\}_{s=1}^{S}, \{T(s)\}_{s=1}^{S}} & \quad \frac{1}{S} \sum_{s=1}^{S} \left[ c_1 E^M(s) + c_2 O(s) + c_3 \sum_{c \in \mathcal{C}(s)} G_c(s) \right] \\
\text{s.t.} \quad & E_{c}(s) \geq E^T, u \in \mathcal{U}(s), s \in \{1, \ldots, S\} \\
& T(s) \in T^F, s \in \{1, \ldots, S\}
\end{aligned}
\]  

(55)

where \(\chi(s) \triangleq \{X_1(s), \ldots, X_M(s)\}\), and \(T(s) \triangleq \{T_u \cup T_{1}^u \cup T_{G}^u\}\). The objective function in (55) consists of three parts, the energy required for swarms to move to their next destination (\(E^M(s)\)), 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 (\(G_c(s), c \in \overline{\mathcal{C}}(s)\)). 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 \(T(s) \triangleq \{T_u \cup T_{1}^u \cup T_{G}^u\}\), \(\forall s\), are always within some feasible set \(T^F\). Finally, \(c_1, c_2, \text{ and } c_3\) in (55) scale the objective components.

Solving \(\mathcal{P}\) 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) \(\mathcal{P}\) 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) \(\mathcal{P}\) is defined over an unknown environment, i.e., neither the data distributions at the device clusters nor the model/concept drift are known a priori. 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 \(s\), \(Z(s)\), as:

\[
Z(s) = U(s) \cup \chi(s) \cup \Lambda(s) \cup E(s) \cup G(s) \cup T(s),
\]

(58)

where \(U(s)\) denotes the set of active UAV swarms, \(\chi(s)\) denotes the locations of all the swarms, \(\Lambda(s) = \{\Lambda_1(s), \ldots, \Lambda_C(s)\}\) are the latest model drift estimates observed at the device clusters, \(E(s) = \{E_1(s), \ldots, E_U(s)\}\), \(G(s) = \{G_1(s), \ldots, G_C(s)\}\), and \(T(s) = \{T_u \cup T_{1}^u \cup T_{G}^u\}\).

### 4) Action of the DRL

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

\[
H(s) = \chi(s+1) \cup U(s+1) \cup T(s+1).
\]

(59)

### 5) Reward of the DRL

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

\[
V(s) = C \left( c_1 O(s) + c_2 \sum_{c \in \mathcal{C}(s)} G_c(s) + c_3 E^M(s) \right) \left( \frac{1}{u \in U} \mathbb{I}_{E_u \leq E^T} \right)
\]

(60)

where (a) captures the (inverse) value of the objective function of \(\mathcal{P}\) 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 \(\mathbb{I}_{E_u \leq E^T}\) takes the value of one for UAV swarm \(u\).

**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:

\[
Q^*(Z(s), H(s)) = E \left[ V(s) \right] + \gamma \max_{H(s+1)} Q^*(Z(s+1), H(s+1)) | Z(s), H(s) \right]
\]

(61)

where \(\gamma\) is the future reward discount. We use one RNN to approximate \(Q^*\) called the train Q-network \(Q_{\theta}(Z(s), H(s))\),
which we train by adjusting its parameters $\theta$ 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 $\theta$. This self-coupling produces over-estimation, so we use an additional RNN called target Q-network $\hat{Q}_{\theta}$ with parameters $\theta$ 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 $\theta$ in the train Q-network, we use experience replay to sample a randomized mini-batch of $M^Q$ experience tuples, each of which is of the form $(s, \mathcal{H}(s), V(s), \mathcal{Z}(s+1))$, to calculate MSE. In particular, we compute the mean-squared error as:

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

where $y(s) = V(s) + \gamma \max_{s' \geq s} \hat{Q}_{\theta}(\mathcal{Z}(s+1), \mathcal{H}(s+1))$ is the target reward. Using the MSE, we then perform SGD to adjust the parameters $\theta$. 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 $\epsilon$-greedy policy [41], wherein with probability $\epsilon$ the agent will select an action randomly and with probability $(1 - \epsilon)$ the agent will determine the action based on the train Q-network $\hat{Q}_{\theta}$. As the train Q-network improves over time, $\epsilon$ decreases with limit $\epsilon_{\text{min}} \leq \epsilon$.

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 Section III, the data processing/offloading optimization from Section IV, and the swarm trajectory and ML design from Section V. The interconnection between the different components is visualized in Fig. 2, which we provided a high level discussion on in Section II-C. Given our introduced variables and parameters in Sections 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., $\rho, \alpha, \alpha, \mathbf{f}$) 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., $\mathbf{X}(s)$) and temporal training decisions (i.e., $T(s), \tau^L, \tau^G$) 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 in the supplementary materials. Descriptions of each dataset and UAV network parameters can also be found there. The key findings for each dataset are qualitatively consistent.

A. HN–PFL Proof-of-Concept

We start by investigating the HN–PFL procedure from Section III, by comparing its performance (measured via classification accuracy, loss, and energy consumption) to that 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 $B^Q$ and setting each batch ratio for HN–PFL...
Fewer aggregations leads to worse performance as a result. Nonetheless, HN-PFL attains better final model accuracy and classification loss over its H-FL counterpart. For example, when \( \tau^G = \tau^s = 1 \), 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 \( \tau^G \) or \( \tau^s \) increases for both MNIST and CIFAR-10.

**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 Section 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 (\( \rho \)), coordinator-to-worker data offloading (\( \varrho \)), aggregate data processing (i.e., \( D_j \times \alpha_{ij} \)), and worker CPU frequency determination (\( g \)). 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 \( \rho \) 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 \( \varrho \) 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 \( \rho \) and \( g \) 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

---

**TABLE II**

| Ratio | MNIST (kJ) | HFL | HNPFL | Savings |
|-------|------------|-----|-------|---------|
| \( \tau^G \) | \( \tau^s \) |     |       |         |
| 1     | 1          | 1   | 1.13  | 0.0%    |
| 1     | 2          | 1.5 | 2.24  | 32.7%  |
| 1     | 4          | 3   | 4.34  | 51.6%  |
| 1     | 8          | 5   | 5.84  | 53.2%  |

**TABLE III**

| Ratio | CIFAR-10 (kJ) | HFL | HNPFL | Savings |
|-------|---------------|-----|-------|---------|
| \( \tau^G \) | \( \tau^s \) |     |       |         |
| 1     | 1             | 1   | 1.77  | 0.0%    |
| 1     | 2             | 1.2 | 2.47  | 24.5%  |
| 1     | 4             | 2   | 4.26  | 50.2%  |
| 1     | 8             | 3   | 5.84  | 53.8%  |

CIFAR-10 to reach a specific training accuracy. For each table, we selected model accuracies that are reachable for all combinations of \( \tau^G \) and \( \tau^s \). 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.
methodology for our optimization problem exists, we develop 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 $P$ 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 $1 - \theta = 0$. 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 Section 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, ···, 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 (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 in the supplementary materials. 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 $\epsilon$ and $\gamma$ in Appendix A in the supplementary materials.

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 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. Abdulkarim, H. Tout, H. Ould-Slimane, A. Mourad, C. Tafhi, 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, Apr. 2021.
[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, 2nd Quart., 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., vol. 71, no. 5, pp. 4977–4990, May 2022.
[4] S. Hosseinipour, C. G. Brinton, V. Aggarwal, H. Dai, and M. Chiang, “From federated to fog learning: Distributed machine learning over heterogeneous wireless networks,” IEEE Commun. Mag., vol. 58, no. 12, pp. 41–47, Dec. 2020.
[5] Y. Jiang et al., “Model pruning enables efficient federated learning on edge devices,” IEEE Trans. Neural Netw. Learn. Syst., early access, Apr. 25, 2022, doi: 10.1109/TNNLS.2022.3166101.
[6] O. M. Rosabal, O. A. López, D. E. Pérez, M. Shehab, H. Hillesheim, and H. Alves, “Minimization of the worst-case average energy consumption in UAV-assisted IoT networks,” IEEE Internet Things J., vol. 9, no. 17, pp. 1–12, Sep. 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., vol. 8, no. 3, pp. 1566–1593, Sep. 2022.
[8] T. Zeng, O. Semiari, M. Moazzami, M. Chen, W. Saad, and M. Bennis, “Federated learning in the sky: Joint power allocation and scheduling with UAV swarms,” 2020, arXiv:2002.08196.
[9] H. Zhang and L. Hanzo, “Federated learning assisted multi-UAV networks,” IEEE Trans. Veh. Technol., vol. 69, no. 11, pp. 14104–14109, Nov. 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,” 2020, arXiv:2011.14197.
[11] R. Cioccare, “Amazon sidewalk privacy and security Whitepaper,” Amazon, Seattle, WA, USA, Rep., Sep. 2020. [Online]. Available: https://m.media-amazon.com/images/G/01/sidewalk/final_privacy_security_whitepaper.pdf
[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,” Criminol. 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, Jan. 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, Jan. 2020.
[15] M. M. Amir and D. Gündüz, “Federated learning over wireless fading channels,” IEEE Trans. Wireless Commun., vol. 19, no. 5, pp. 3546–3557, May 2020.
[16] S. Wang et al., “Adaptive federated learning in resource constrained edge computing systems,” IEEE J. Sel. Areas Commun., vol. 37, no. 6, pp. 1205–1221, May 2020.
[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, Aug. 2021.
[18] S. Hosseinipour et al., “Parallel successive learning for dynamic distributed model training over heterogeneous wireless networks,” 2022, arXiv:2202.02947.
[19] S. Hosseinipour et al., “Multi-stage hybrid federated learning over large-scale D2D-enabled fog networks,” IEEE/ACM Trans. Netw., vol. 30, no. 4, pp. 1569–1584, Aug. 2022.
[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, Mar. 2020.
