

Stitching Satellites to the Edge: Pervasive and Efficient Federated LEO Satellite Learning

Mohamed Elmahallawy, Tie Luo*

Department of Computer Science, Missouri University of Science and Technology, Rolla, MO 65401, USA
Emails: {meqxx, tluo}@mst.edu

Abstract—In the ambitious realm of space AI, the integration of federated learning (FL) with low Earth orbit (LEO) satellite constellations holds immense promise. However, many challenges persist in terms of feasibility, learning efficiency, and convergence. These hurdles stem from the bottleneck in communication, characterized by sporadic and irregular connectivity between LEO satellites and ground stations, coupled with the limited computation capability of satellite edge computing (SEC). This paper proposes a novel FL-SEC framework that empowers LEO satellites to execute large-scale machine learning (ML) tasks onboard efficiently. Its key components include i) *personalized learning via divide-and-conquer*, which identifies and eliminates redundant satellite images and converts complex multi-class classification problems to simple binary classification, enabling rapid and energy-efficient training of lightweight ML models suitable for IoT/edge devices on satellites; ii) *orbital model retraining*, which generates an aggregated “orbital model” per orbit and retrains it before sending to the ground station, significantly reducing the required communication rounds. We conducted experiments using Jetson Nano, an edge device closely mimicking the limited compute on LEO satellites, and a real satellite dataset. The results underscore the effectiveness of our approach, highlighting SEC’s ability to run lightweight ML models on real and high-resolution satellite imagery. Our approach dramatically reduces FL convergence time by nearly 30 times, and satellite energy consumption down to as low as 1.38 watts, all while maintaining an exceptional accuracy of up to 96%.

Index Terms—Low Earth orbit (LEO) satellite, satellite edge computing (SEC), federated Learning (FL).

I. INTRODUCTION

The fusion of AI and edge computing with low Earth orbit (LEO) satellites holds promise for enhancing a multitude of space applications such as monitoring remote regions (deserts, forests, polar areas, etc.) and maritime zones for disaster management [1]. This potential is propelled by ongoing technological advancements in hardware capabilities and the burgeoning support for AI within the global community. Notably, rapid progress in camera technologies and the emergence of quantum computing are poised to usher in a new era of on-orbit AI [2]. However, many challenges need to be addressed in this endeavor. For example, the irregular and intermittent visibility of LEO satellites to ground stations, constrained communication bandwidth, and data privacy and security, are substantial bottlenecks hindering the integration of AI into space applications. Prominent among these challenges is the severe limitation in the computing capacity of LEO satellites (e.g., CubSat and nanosatellite), which significantly impedes the training of large machine learning (ML) models.

Federated learning (FL) has emerged as a privacy-aware distributed ML approach [3]. It enables clients (in this case LEO satellites) to collaboratively train an ML model onboard without transmitting raw data. This collaboration is orchestrated through a parameter server (\mathcal{PS}), which is typically a ground station (GS) in the case of LEO satellites. However, integrating FL with LEO satellites as edge nodes within the satellite edge computing (SEC) framework, presents significant challenges. These challenges arise from the inherent conflict between the restricted computing capabilities of LEO satellites and the vast volume of Earth observation images they collect, often necessary for training large-scale ML models [4]. The process of training such models is power-intensive, demanding substantial computation, time, and energy—a heavy burden for LEO satellites. Consequently, the feasibility of conducting onboard ML for training global FL models across multiple orbits, as in an LEO constellation, becomes highly impractical.

On the other hand, addressing computing and power consumption concerns is not the sole obstacle to achieving rapid FL convergence and high efficiency within SEC. A critical factor lies in the inherent nature of FL, which relies on iterative communication rounds between clients and the \mathcal{PS} . This leads to a notably prolonged learning process within SEC due to the low satellite bandwidth and, more importantly, the highly intermittent and sporadic connections to the \mathcal{PS} . The latter arises from the substantial disparities between the trajectory of the Earth (rotating around the Axis) and that of LEO satellites (varying in any angle between 0 and 90 degrees w.r.t. the equator), as well as the large speed difference (29.78 km/s for the Earth and 7.8 km/s for satellites). Moreover, the visible windows of LEO satellites to the \mathcal{PS} are typically short, lasting only a few minutes on average, rendering them inadequate for the transmission of large ML models. Consequently, these factors collectively extend the FL convergence time in SEC significantly, taking several days or even weeks [5, 6].

In response to these challenges, this paper proposes a pragmatic FL framework tailored for SEC towards achieving pervasive space AI. This framework empowers LEO satellites, equipped with limited computing power, memory, and energy resources, to participate in collaborative onboard ML model training across extensive datasets spanning multiple orbits. Furthermore, it accelerates FL convergence significantly by substantially reducing the number of communication rounds while upholding competitive model accuracy. The following summarizes our main contributions.

- We propose an scheme, *personalized learning via divide-and-conquer*. This scheme identifies and eliminates redun-

*Corresponding author.
This work was supported by the National Science Foundation (NSF) under Grant No. 2008878.

dant satellite images on each individual satellite, effectively reducing data volume from billions to thousands without impairing model quality. Furthermore, it simplifies the original ML problem from a complex multi-class classification into a streamlined binary classification, enabling the training of lightweight ML models on each satellite. This approach is particularly advantageous for conserving energy and accommodating computing and memory limits on those small satellites. Notably, a subsequent model aggregation process restores the multi-class classification problem, thereby preserving the intended model quality and purpose.

- We propose an *orbital model retraining* scheme, wherein a designated satellite within each orbit aggregates all models from satellites within the same orbit, forming an “orbital model”, and subsequently, retrains it before transmitting to the \mathcal{PS} . This approach i) minimizes the waiting time for all satellites to individually enter their respective visible zones for model transmission to \mathcal{PS} , and ii) reduces the number of required communication rounds significantly, from typically tens to only a few. Ultimately, it leads to rapid global model convergence.
- Using a real-world satellite imagery dataset for classification tasks, our experimental evaluations showcase remarkable efficiency. The proposed FL-SEC framework demonstrates swift convergence within a mere 2 hours while maintaining competitive accuracy levels. We measure accuracy using various metrics, including accuracy, precision, recall, and F1-score. Furthermore, our approach yields lightweight models with minimal computation, communication overhead, and energy consumption—significantly lower than previous solutions in the literature. These desired properties hold substantial promise, especially for resource-constrained LEO satellites that often rely on solar power.

II. RELATED WORK

Despite the nascent stage of integrating FL into SEC, early attempts have been made [5–15]. For instance, some *synchronous FL approaches*, such as [5], employed the vanilla FedAvg [3] with SEC without specific adaptations for communication challenges in LEO constellations. In the work of [7], FedISL was proposed, utilizing inter-satellite links (ISL) to expedite FL training in SEC. Another study, [8], incorporated high altitude platforms (HAPs) like aerostats, stratospheric balloons, and stratollites as \mathcal{PS} s to expedite FL convergence. In [9], the author proposed a clustering and edge selection approach wherein the GS forms clusters, each containing an LEO satellite as server and nearby LEO clients, based on channel quality. However, all these previous works simplified the problem by relying on simple datasets (e.g., MNIST [16] and CIFAR-10 [17]) for training tasks, and even then, they experienced long convergence times. This limitation significantly impedes their applicability in practical scenarios where each satellite trains an ML model using real satellite images, highlighting the need for more efficient and practical training approaches. Although the authors of [10, 11] demonstrated the effectiveness of their approach on a realistic satellite dataset containing satellite images, they overlooked the necessary computing and communication resources required for model execution and transmission to the \mathcal{PS} .

On the other hand, some *asynchronous FL approaches* have emerged as well. For instance, FedSat was introduced in [12], which involves averaging received satellite models in a regular order based on their visibility. It assumes regular satellites visit the GS, once per orbital period, aiming to ensure an equal contribution to the global model and to mitigate staleness issues typical in standard asynchronous FL approaches. In a similar vein, [13] introduced a graph-based routing and resource reservation algorithm to optimize FL model parameter transfer delays faced by [12]. AsyncFLEO [14] tackled staleness by grouping models collected from satellites in different orbits based on their similarity, maintaining balance by selecting an equal number of models from each group. Fresh models are added without weighting, while stale ones are down-weighted based on their staleness. Another approach, known as FedSpace and proposed by So et al. [6] attempted to strike a balance between the idleness associated with synchronous FL and the staleness issues associated with asynchronous FL. However, FedSpace relies on satellites uploading a portion of their raw data to the GS, which conflicts with the principles of efficient communication and data privacy in FL. Finally, Wu et al. [15] propose FedGSM, a method addressing gradient staleness through a compensation mechanism. By harnessing the deterministic and time-varying orbit topology, FedGSM mitigates the adverse effects of staleness, yet it still exhibits slow convergence.

All these previous efforts to integrate FL with SEC have primarily aimed at addressing intermittent connections between LEO and the \mathcal{PS} and expediting FL convergence. However, they commonly overlook the limited satellite computing resources and neglect power consumption optimization for on-edge model training, which limits their practicality in SEC scenarios.

III. SYSTEM MODEL AND PRELIMINARIES

For clarity, Table I presents all the main notations used in this paper along with their corresponding descriptions.

A. System Model

We consider a generic LEO constellation in the context of Satellite Edge Computing (SEC). The constellation consists of N orbits, each having a set \mathcal{I} of equal-distanced satellites with unique IDs. Within each orbit, the satellites’ computing and storage resources allocated to the FL task are denoted as $\mathcal{C} = \{c_1, c_2, \dots, c_I\}$ and $\mathcal{S} = \{s_1, s_2, \dots, s_I\}$, where $c_i \in [0, 1]$, $s_i \in [0, 1]$, and $I = |\mathcal{I}|$. Satellites in orbit n travel around the Earth at an altitude h_n with a speed of $v_n = \frac{2\pi(R_E + h_n)}{T_n}$, where R_E is the radius of the Earth and T_n is the orbital period given by $T_n = \frac{2\pi}{\sqrt{G_E M}}(R_E + h_n)^{3/2}$, with G_E denoting the gravitational constant and M representing the mass of the Earth. While orbiting the Earth, these satellites capture high-resolution Earth observation images that will be used for training ML models for a variety of classification tasks. This takes place under an FL framework, where each satellite initially receives a global model from the \mathcal{PS} when it enters its respective visibility window. Subsequently, the satellite independently retrains the global model using its locally collected data and transmits the updated model back to the \mathcal{PS} , when its next visible window arrives. After collecting

TABLE I: Key notations for computation&communication models.

Computation Parameters	
β	Index to the global communication rounds
V	Predefined number of orbital training rounds
w^β	Global ML model
w_i^β	Local model of a satellite i in round β
$ w $	Size of the w in bits
η	Learning rate
D_i	Private data of a satellite i
m_i	Size of a SEC i 's data
m^β	Total data size of all satellites in round β
π_i	Filtering Policy for a satellite i 's collected Images
C_i^{CPU}	Number of CPU cores of a satellite i
f_i	CPU clock frequency of a satellite i [GHz]
t_{filter}^i	Time require for a satellite i to filter its images [s]
t_{train}^i	Time require for a satellite i to train an ML model [s]
Communication Parameters	
P	Transmitter power (SEC or \mathcal{PS}) [dBm]
G_i/\mathcal{PS}	Antenna gain for a satellite i or \mathcal{PS} [dBi]
T	Noise temperature [K]
K_B	Boltzmann constant [J/K]
B	Bandwidth [GHz]
$\mathcal{L}_{i,\mathcal{PS}}$	Free-space path loss between a satellite i and the \mathcal{PS}
$d_{i,\mathcal{PS}}$	Threshold distance for feasible LoS communication between a satellite i and the \mathcal{PS} [km]
λ	Signal wavelength [mm]
R	Achievable data rate between SEC and the \mathcal{PS} [Mb/s]
t_{wait}^i	Waiting time for a satellite i to enter \mathcal{PS} 's visible zone [s]
$t_{\text{trans}}^{i,\mathcal{PS}}$	Transmission time of a model between satellite i and \mathcal{PS} [s]
$t_{\text{prop}}^{i,\mathcal{PS}}$	Propagation time for a model by a satellite i or \mathcal{PS} [s]

the model parameters from all the orbits, the \mathcal{PS} aggregates them into an updated global model and distributes it once again to all satellites, initiating another communication round. This iterative process continues until the global model achieves convergence (i.e., meets a predefined termination criterion, such as reaching a target accuracy or loss, a maximum number of communication rounds, or negligible changes in the global model parameters).

B. Preliminaries of Federated Learning in SEC

Computation Model.

In each communication round $\beta = 0, 1, \dots, B$:

- 1) The \mathcal{PS} transmits the global model w^β to each satellite when that satellite comes into its visible zone.
- 2) Each satellite i retrains w^β using a local optimization method such as stochastic gradient descent (SGD) on its collected Earth observation data, as

$$w_i^{\beta,j+1} = w_i^{\beta,j} - \eta \nabla F_k(w_i^{\beta,j}; \{(X_i^q, y_i^q)\}) \quad (1)$$

where $w_i^{\beta,j}$ denotes i 's own copy of the model (referred to as a "local model") at the j -th local training epoch ($j = 1, 2, \dots, J$), $(X_i^q, y_i^q) \subset D_i$ signifies the q -th mini-batch for each epoch j , D_i is satellite i 's dataset, and η denotes the

learning rate. After completing the training, the satellite sends the updated local model $w_i^{\beta,j}$ back to \mathcal{PS} .

- 3) Upon receiving the updated local models from all satellites, the \mathcal{PS} aggregates them into an updated global model, as $w^{\beta+1} = \sum_{i \in \mathcal{I}} \frac{m_i}{m} w_i^{\beta,j}$, where $m_i = |D_i|$ is the dataset size for satellite i and $m = \sum_{i \in \mathcal{I}} m_i$. The \mathcal{PS} then distributes this updated global model to all satellites again, during their respective visibility windows, as described in (1).

Communication Model. In an LEO constellation \mathcal{I} , a satellite i and the \mathcal{PS} can establish a line-of-sight (LoS) communication if the angle between their respective trajectories satisfies $\angle(r_{\mathcal{PS}}(t), (r_i(t) - r_{\mathcal{PS}}(t))) \leq \frac{\pi}{2} - \alpha_{min}$. Here, $r_i(t)$ and $r_{\mathcal{PS}}(t)$ are the trajectories of satellite i and the \mathcal{PS} , respectively, and α_{min} is the minimum elevation angle. Furthermore, given an additive white Gaussian noise (AWGN) channel, the signal-to-noise ratio (SNR) between them can be expressed by

$$\text{SNR}_{i,\mathcal{PS}} = \frac{PG_iG_{\mathcal{PS}}}{TBK_B\mathcal{L}_{i,\mathcal{PS}}} \quad (2)$$

where P denotes the transmission power, G_i and $G_{\mathcal{PS}}$ correspond to the antenna gains of satellite i and \mathcal{PS} , respectively, T stands for the noise temperature, B is the channel bandwidth, K_B denotes the Boltzmann constant, and $\mathcal{L}_{i,\mathcal{PS}}$ signifies the free-space path loss. This path loss can be determined as

$$\mathcal{L}_{i,\mathcal{PS}} = \left(\frac{4\pi \|i, \mathcal{PS}\|_2}{\lambda} \right)^2 \quad (3)$$

under the condition that the Euclidean distance between satellite i and \mathcal{PS} satisfies $\|i, \mathcal{PS}\|_2 \leq d_{i,\mathcal{PS}}$, where $d_{i,\mathcal{PS}}$ is the minimum distance that allows for LoS communication between satellite i and \mathcal{PS} , and λ is the wavelength of the transmitted signal.

IV. PROPOSED FRAMEWORK

Recall that the hurdle of inefficiency in FL-SEC arises from several factors including the large number of communication rounds, sporadic and irregular visibility pattern of satellites, vast volume of data and limited computation and communication resources. These factors not only substantially slow down convergence but also give rise to notable temperature and power concerns [18].

To tackle these challenges, we propose a two-fold approach: 1) In Section IV-A, we reduce the quantity of satellite images by filtering them based on their relevance to target classes, enabling each satellite to train a local model customized to a particular class associated with itself. This not only shrinks the model size to fit within the onboard processing capacity, but also involves resizing images to ensure they fall below the maximum allowable training time, all without sacrificing model accuracy. 2) In Section IV-B, we introduce a scheme where a designated satellite within each orbit generates and retrains an "orbital model" before transmitting it to the \mathcal{PS} . Not only does this overcome computation and energy constraints, but it also drastically reduces the number of required communication rounds.

Algorithm 1: Personalized Learning via Divide-and-Conquer

Input: Satellite constellation \mathcal{I} , all orbits \mathcal{N} , set of satellites \mathcal{I}_n for each orbit $n \in \mathcal{N}$, satellite i 's captured data D_i , number of classes L .

Output: Set of binary-classifiers $\{\mathbf{w}_i^\beta\}_n$ for each orbit n .

- 1 **foreach** $n \in \mathcal{N}$ **do**
- \triangleright Divide Multi-classes Task into L Binary Tasks
- 2 **if** $L \leq |\mathcal{I}_n|$ **then**
- 3 Assign L binary training tasks to L satellites who have sufficient computing resources
- 4 $\mathcal{I}_n = \{\text{the above } L \text{ satellites}\}$
- 5 **foreach** $i \in \mathcal{I}_n$ **do**
- 6 $D_i^{\text{filtered}} \leftarrow$ Filter D_i for target class l using (6)
- 7 Train a binary classifier model \mathbf{w}_i^β using D_i^{filtered}
- 8 $\{\mathbf{w}_i^\beta\}_n \leftarrow \{\mathbf{w}_i^\beta\}_n \cup \mathbf{w}_i^\beta$
- 9 **else**
- 10 Partition L classes into $|\mathcal{I}_n|$ groups \mathcal{G}
- 11 **foreach** $i \in \mathcal{I}_n$ **do**
- 12 Assign satellite i with a group G of binary tasks
- 13 **foreach** $\text{task } y \in G$ **do**
- 14 $D_i^{\text{filtered}} \leftarrow$ Filter D_i for target class y using (6)
- 15 Train a binary classifier model \mathbf{w}_i^β on D_i^{filtered}
- 16 $\{\mathbf{w}_i^\beta\}_n \leftarrow \{\mathbf{w}_i^\beta\}_n \cup \mathbf{w}_i^\beta$
- 17 Forward $\{\mathbf{w}_i^\beta\}_n$ w/ the filtered data size of each satellite i to sink satellite for re-assembling (Section IV-B)

A. Personalized Learning via Divide and Conquer

We take a divide-and-conquer (DnC) approach to divide the task of training a multi-class model into L tasks, each involving the training of a binary classifier dedicated to a single class (where L is the number of classes). If $L \leq I_n$ (i.e., number of satellites per orbit), each satellite will handle one such binary task; otherwise, multiple binary tasks are assigned and independently executed on each satellite. After completion of all such ‘‘personalized’’ training tasks, the trained binary models will be merged into the original intended multi-class classifier. The benefit of this method is that it substantially reduces each satellite’s computation load, aligning well with the capability of SEC.

Algorithm 1 describes this approach which encompasses the following key elements. Each satellite utilizes only a subset of collected images, selected by their relevance to the satellite’s target class while disregarding the rest. This strategic filtering reduces the number of training samples from millions to thousands, all while upholding the model’s accuracy through an orbital retraining process described in Section IV-B. Additionally, each satellite resizes the chosen subset of images, further enhancing efficiency. These combined actions enable each satellite to complete its local model training within a matter of minutes, despite its limited computation capacity. Moreover, we formulate the minimum time required by each communication round while ensuring the gathering of satellite models from all orbits by the \mathcal{PS} . Fig. 1 provides an overview.

We begin with the conventional FL-SEC approach, which uses a star topology as its communication architecture. Later in Section IV-B, we will replace it with a much more efficient architecture. In this conventional framework, we introduce a data filtering scheme. As such, the total time required for each

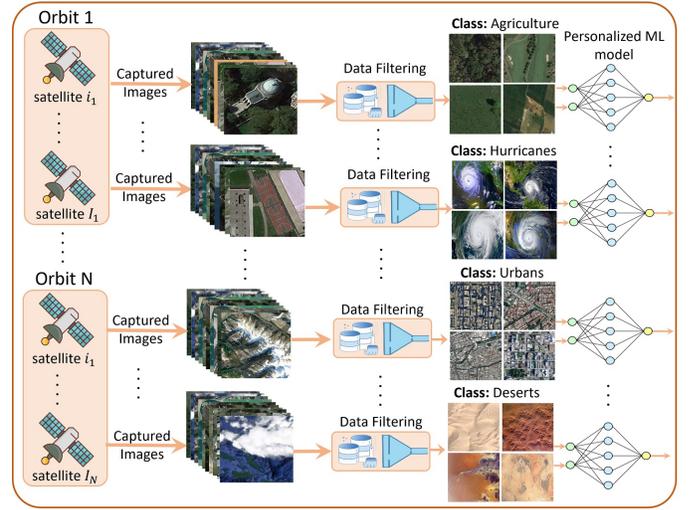


Fig. 1: Personalized Learning via Divide-and-Conquer.

satellite i to receive the global model \mathbf{w}^β from the \mathcal{PS} , filter its collected images, train \mathbf{w}^β to obtain its local model \mathbf{w}_i^β , and transmit \mathbf{w}_i^β to the \mathcal{PS} , can be formulated as

$$t_{\text{req}}^i = t_{\text{wait}}^i + 2(t_{\text{trans}}^{i,\mathcal{PS}} + t_{\text{prop}}^{i,\mathcal{PS}}) + t_{\text{filter}}^i + t_{\text{train}}^i \quad (4)$$

where t_{wait}^i is the waiting time for satellite i to enter its visible zone, $t_{\text{trans}}^{i,\mathcal{PS}} = \frac{|w|}{R}$ is the transmission time of a model between satellite i and the \mathcal{PS} (note that \mathbf{w}^β and \mathbf{w}_i^β have the same size, differing in parameter values only), $R = B \log_2(1 + \text{SNR}_{i,\mathcal{PS}})$ is the achievable data rate between satellite i and the \mathcal{PS} , $t_{\text{prop}}^{i,\mathcal{PS}} = \frac{\|i,\mathcal{PS}\|_2}{c}$ is the propagation delay between satellite i and the \mathcal{PS} , t_{filter}^i is the image filtering time, and t_{train} is model train time. The filtering time can be expressed by

$$t_{\text{filter}}^i = \frac{D_i \pi_i}{C_i^{\text{CPU}} f_i} \quad (5)$$

where π_i is the policy employed by satellite i to filter its collected images down to a threshold, C_i^{CPU} is the number of CPU cores available on satellite i , and f_i is its CPU clock frequency, which is adjustable through dynamic voltage and frequency scaling (DVFS) to minimize energy consumption.

Our innovation toward efficient learning is to re-distribute the learning tasks among the satellites within the same orbit, such that each satellite is tasked with training a local model for classifying a single, specific class y . For instance, each class could pertain to a particular phenomenon such as a hurricane, forest fire, building footprint, flooding, or others. Hence, satellite i utilizes a sampling policy π_i to filter out images not belonging to its designated target class, thereby converting the original multi-class problem into one that can be solved by a binary classifier. These binary models from all satellites in orbit n will later be re-assembled into a multi-class classifier in Section IV-B using a one-vs-all (OvA) strategy. The reduced dataset (after filtering), denoted by D_i^{filtered} , is given by

$$D_i^{\text{filtered}} = \{X_i \mid (X_i, y) \in D_i, \pi_i(y) = 1\} \quad (6)$$

To identify images belonging to a target class, satellites have three options: (i) use the metadata of captured images, such as time, GPS location, satellite's attributes, etc. to determine whether an image is taken in the area of interest to that satellite that contains the target class of objects [19]; (ii) alternatively, satellites can store some labeled images beforehand or pre-train a model on them, and subsequently employ a semi-supervised classification method to label the newly collected data [20]; (iii) satellites can also filter their data during the collection phase, where each satellite activates its camera on or off to only capture images of interest that belong to its target class [21]. Thus, after our data reduction (6), the learning task is converted from multi-class to binary classification, leading to lightweight models and lower satellite energy consumption. In Eq. (4), t_{train} can be expressed by

$$t_{\text{train}}^i = \frac{\nu_i \cdot J \cdot c_{\text{process}} + c_{\text{overhead}}}{f_i} \quad (7)$$

where $\nu_i = \left\lceil \frac{m_i^{\text{filtered}}}{\kappa} \right\rceil$ is the total number of the mini-batches on satellite i with $m_i^{\text{filtered}} = |D_i^{\text{filtered}}|$ denotes the number of selected images and κ is the batch size, J is the number of epochs as defined below (1), c_{process} is the average number of processing cycles needed for training a mini-batch on a satellite, and c_{overhead} is the CPU/GPU cycles required for processing any additional overheads. Here, implicitly, we assume homogeneous satellites and image sizes, which is justified by the fact that they all belong to the same constellation. Also note that our data filtering will lead to a much-reduced value for ν_i , minimizing the training time.

Thus, the required time for completing all the local models within an orbit $n \in \mathcal{N}$ can be calculated by

$$t_{\text{req}}^{\mathcal{I}_n} = \begin{cases} \sum_{i \in \mathcal{I}_n} t_{\text{req}}^i, & \text{if } t_{\text{req}}^i < t_{\text{visible}}^i \\ \sum_{i \in \mathcal{I}_n} (t_{\text{req}}^i + \alpha t_{\text{wait}}^i), & \text{if } t_{\text{req}}^i \geq t_{\text{visible}}^i. \end{cases} \quad (8)$$

where \mathcal{I}_n is the set of all the satellites on orbit n , α denotes the number of revolutions (cycles) for the satellite to become visible to the \mathcal{PS} for sending its trained model, and t_{visible}^i is the length of the visible window. Thus, the minimum time required for the \mathcal{PS} to collect all the satellite models, as in a global communication round, is

$$t_{\text{req}}^{\mathcal{I}} = \max_{n \in \mathcal{N}} \{t_{\text{req}}^{\mathcal{I}_n}\} \quad (9)$$

Recall that this subsection assumes the conventional FL-SEC with a star topology. That approach follows a mostly sequential process where each satellite individually uploads its model to the \mathcal{PS} , requiring substantial time per communication round. In contrast, our proposed scheme needs only one satellite per orbit to communicate with \mathcal{PS} , as detailed in Section IV-B.

B. Orbital Retraining and Convergence

The purpose of this scheme is to reconstruct the multi-class learning task by aggregating binary classifiers from all satellites within each orbit $n \in \mathcal{N}$ into an orbital model, using the OvA strategy. Subsequently, this orbital model is retrained through multiple orbital epochs until it achieves an acceptable

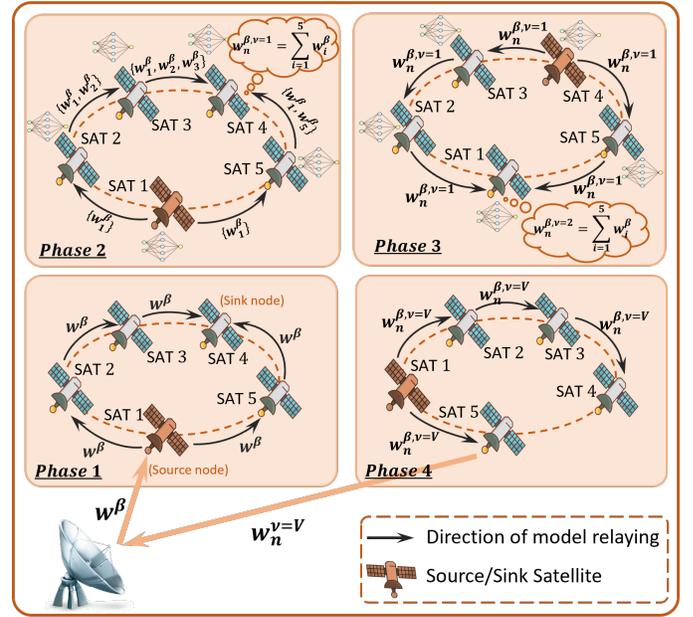


Fig. 2: Illustration of Orbital Retraining (Section IV-B).

level of accuracy before it is sent back to the \mathcal{PS} . This orbital training not only improves the accuracy of the orbital model, but also eliminates a large number of time-consuming global communication rounds by introducing the much faster orbit epochs. Algorithm 2 summarizes this approach.

Assumption 1. Each satellite i has sufficient training samples for its target class y . More formally,

$$F_y(\mathbf{w}_i^J; D_i^{\text{filtered}}) < \tau$$

where $F_y(\mathbf{w}_i^J; D_i^{\text{filtered}})$ is the training loss of a satellite i over the filtered dataset for a target class y , and τ is a loss threshold to control the convergence of the satellite's local model.

Our proposed scheme works as follows (cf. Fig. 2):

- Phase 1:** Within each orbit $n \in \mathcal{N}$, the first satellite who receives the global model \mathbf{w}^β from the \mathcal{PS} , in this case #1 and referred to as the “source” node, will initiate a model relaying process by forwarding \mathbf{w}^β to its next-hop neighbors. The neighbors will then continue to relay \mathbf{w}^β onward, and so forth until eventually reach a “sink” node. Each node starts to train \mathbf{w}^β locally right after forwarding it (except that the sink node does not forward). Note that data filtering and resizing are also performed as per Section IV-A prior to training.
- Phase 2:** After training, each satellite i obtains an updated \mathbf{w}^β which is a local model \mathbf{w}_i^β that satisfies Assumption 1. The source satellite, #1, then initiates another round of relay by forwarding its trained model \mathbf{w}_1^β to its next-hop neighbors, #2 and #5, who will then pass their own \mathbf{w}_2^β and \mathbf{w}_5^β , together with the received \mathbf{w}_1^β , onto their respective neighbors ahead. This process continues until all satellites' models within that orbit reach the sink node #4. This sink then aggregates all the received models into an “orbital model” $\mathbf{w}_n^{\beta,v}$, where the orbit index n replaces satellite i , and $v = 1, 2, \dots, V$ indexes the intra-orbital training

Algorithm 2: Orbital Retraining and Convergence Process

Input: $\{\mathbf{w}_i^\beta\}_n$ for each orbit $n \in \mathcal{N}$, m_i^{filtered} for all satellites, and β .

Output: Trained global model \mathbf{w}^β .

- 1 **foreach** $\beta = 0, 1, \dots, B$ **do**
- 2 **foreach** $n \in \mathcal{N}$ **do**
 - ▷ Convert Binary tasks back into a multi-class task
 - 3 Generate $\mathbf{w}_n^\beta = \sum_{i \in \mathcal{I}_n} \frac{m_i^{\text{filtered}}}{m_{\mathcal{I}_n}} \mathbf{w}_i^\beta$
 - 4 Retrain \mathbf{w}_n^β for V orbital epochs as in (10)
 - 5 Generate $\mathbf{w}^{\beta+1} = \sum_{n \in \mathcal{N}} \frac{|D_n|}{|D_N|} \mathbf{w}_n^{\beta,V}$
 - 6 $\beta \leftarrow \beta + 1$

rounds. Trivially, any redundant models (in this case $\mathbf{w}_1^{\beta,v}$) will be removed before aggregation. Note that the orbital model *recovers* the intended multi-class classifier from the binary classifiers.

- 3) **Phase 3:** After generating the orbital model $\mathbf{w}_n^{\beta,v=1}$, the sink satellite initiates a “reverse relay” and retraining process which is the same as Phases 1 and 2 except that the direction is from sink to source and the orbital model $\mathbf{w}_n^{\beta,v=1}$ replaces the global model \mathbf{w}^β . Subsequently, the source satellite # will aggregate all the trained models $\mathbf{w}_{n,i}^{\beta,v=1}$ into an updated orbital model $\mathbf{w}_n^{\beta,v=2}$. Thus, Phases 1-3 repeat (with the orbital model in place of the global model) to retrain the orbital model until $v = V$.
- 4) **Phase 4:** Finally, the satellite who aggregates $\mathbf{w}_n^{\beta,v=V}$, either source or sink, forwards $\mathbf{w}_n^{\beta,v=V}$ to all satellites on the same orbit (via relay) so that *any* satellite that becomes *visible first* can send $\mathbf{w}_n^{\beta,V}$ to the \mathcal{PS} (for global aggregation among all orbits).

Note that each model relay, in either a forward or reverse direction, is almost instantaneous because of the short inter-satellite distance and, optionally, the potential use of FSO in place of RF antenna. In contrast, in the model exchange between satellite and \mathcal{PS} , as in the conventional FL star topology, the delay is remarkably substantial because of the idle waiting for visible windows, as well as the long-distance (where only RF can be used).

This intra-orbit retraining process is important for two reasons: (1) it recovers the intended multi-class model from the binary classifiers “personalized” to each individual satellite; (2) the multi-round intra-orbit retraining takes maximal advantage of the feature learning capability from each personalized model and combines their best-learned patterns to effectively enhance the final aggregated model performance across all classes.

Below, we provide more details about Phases 1-4 outlined above. In **Phase 1**, the intra-orbit model relay eliminates the need for each node to wait for its visible window, thus ensuring each satellite to receive the global model at the earliest possible time. In **Phase 2**, the orbital model is aggregated as follows:

$$\mathbf{w}_n^{\beta,v+1} = \sum_{i \in \mathcal{I}_n} \frac{m_i^{\text{filtered}}}{m_{\mathcal{I}_n}} \mathbf{w}_i^{J,v}, \text{ where } m_{\mathcal{I}_n} = \sum_{i \in \mathcal{I}_n} m_i^{\text{filtered}} \quad (10)$$

where $\mathbf{w}_n^{\beta,v}$ is the orbital model for orbit n at the v -th orbital iteration for a global round β . In **Phase 4**, when the visible satellite uploads the orbital model to the \mathcal{PS} , it will also

upload *metadata* including the total data size $m_{\mathcal{I}_n}$ utilized for training $\mathbf{w}_n^{\beta,V}$, and the distribution of data (in terms of classes) across satellites within orbit n .

Corresponding to Eq. (4) and (9) which are formulated under the star topology, now we reformulate them under the new communication architecture involving model relay and retraining for V iterations. Expressing the time needed for transmitting a model between two adjacent satellites via a single ISL hop as

$$t_n^{\text{ISL}} = \frac{h |\mathbf{w}|}{B^{\text{ISL}} \varrho} \quad (11)$$

where $h = 1, 2, \dots, H$ indexes the relay hops between source and sink satellites, $H = \lceil I_n/2 \rceil$ is the total hops due to the concurrent bilateral relaying in the same direction, B^{ISL} denotes the allocated bandwidth for the ISL communication between two adjacent satellites, and ϱ stands for the spectral efficiency of the communication link. Consequently, the overall model relay and retraining time on orbit n is given by

$$t_n^{\text{ISL}} = V \left(\sum_{h=1}^H t_n^{\text{ISL}} + t_{\text{train}} \right) + t_{\text{filter}} + t_{\text{wait}}^n + 2(t_{\text{trans}}^{n,\mathcal{PS}} + t_{\text{prop}}^{n,\mathcal{PS}}) \quad (12)$$

Note that t_{wait}^n is significantly smaller than t_{wait}^i in Eq. (4), because we only need the *first* node to become visible to transmit the orbital model to the \mathcal{PS} , while an arbitrary node as in (4) would wait for much longer.

Now, we can express the minimum time required for the \mathcal{PS} to obtain all the orbital models since the beginning of each round. This lower bound of time encompasses several components captured by (9) and (12): (i) the waiting time for a satellite to enter its visible window, (ii) the transmission and propagation time between the \mathcal{PS} and a visible satellite, either for the satellite to receive the global model \mathbf{w}^β or for the \mathcal{PS} to receive an orbital model $\mathbf{w}_n^{\beta,V}$, (iii) the time for data filtering and orbital model retraining within each orbit, and (iv) the time for intra-orbit model relay. As a result, Eq. (9) can be reformulated as

$$t_{\text{req}}^{\mathcal{I}} = \max_{n \in \mathcal{N}} \{ t_n^{\text{ISL}} \} \quad (13)$$

where t_n^{ISL} as given by (12) captures the above four time components.

Eqn. (13) reveals the benefit of orbital retraining which accelerates convergence by minimizing local model training time. In addition, it implies that the required time $t_{\text{req}}^{\mathcal{I}}$ for each round is controllable, by allowing each satellite to use only a subset of training images belonging to its target class, to reduce t_{train} (since t_{wait} , t_{trans} and t_{prop} cannot be further reduced). On the other hand, this would compromise the global model accuracy. Therefore, a trade-off needs to be made between (i) crafting efficient ML models that align with satellite resources and (ii) achieving the desired convergence speed and accuracy. Via meticulous and empirical assignment of each satellite with reduced size of training data belonging to its target class, we have achieved significantly low computation and communication overheads yet competitive accuracy as elaborated in Section V (see Table V).

TABLE II: Parameter Setting For our Performance Evaluation.

Computation	
Number of orbital epochs (V)	5
Number of CPU cores of each satellite (C_i^{CPU})	4
CPU clock frequency of each satellite (f_i)	1.43 GHz
Communication	
Transmitter power (P)	60 dBm
Antenna gain for any satellite or $\mathcal{P}\mathcal{S}$ ($G_i/\mathcal{P}\mathcal{S}$)	6.98 dBi
Noise temperature (T)	354.81 K
Bandwidth (B)	0.5 GHz
Signal wavelength (λ)	15 mm
Maximum data rate (R)	16 Mb/s
SEC Training Models	
Mini-batch size (κ)	4
Learning rate (η)	0.001

V. PERFORMANCE EVALUATION

A. Experimental Setup

1) *LEO Constellation*: We evaluate the convergence speed of our approach by employing two different configurations of the Walker Delta satellite constellation [22]. These configurations differ in orbital inclination: one with an inclination angle of 45° (referred to as an *inclined constellation*) and the other with an inclination angle of 85° (referred to as *near-polar constellation*). Each constellation consists of 6 orbits, with each orbit n containing $I_n = 10$ equally spaced satellites. In both constellations, all satellites are positioned at an altitude h_n of 530 km. We consider a GS located in the central region of the U.S. (Rolla, Missouri) as our $\mathcal{P}\mathcal{S}$, although it can be situated anywhere on the Earth. The GS has a minimum elevation angle of 10° . To determine satellite-GS connectivity, we utilize a Systems Tool Kit simulator developed by AGI [23]. We assign the computation and communication parameters with their respective values as presented in Table II.

2) *Implementation*: To ensure the compatibility of our approach with real satellite equipment, we implemented our approach on a Jetson Nano Dev-Kit¹. This energy-efficient device boasts a 128-core NVIDIA Maxwell GPU, a Quad-core ARM Cortex®-A57 MPCore processor, 4GB of 64-bit memory, and runs on a remarkably low power consumption of just 5 watts. During our experiments, we meticulously monitored both the computation overhead and the energy consumption associated with training tasks. Fig. 3 shows our system setup.

3) *Datasets & Satellites Training*: To evaluate our approach, we train the satellite’s local models using a real satellite dataset, EuroSat [24]. Additionally, we employ three other datasets—MNIST, CIFAR-10, and CIFAR-100—commonly used in state-of-the-art (SOTA) comparisons to assess our approach. Below, we provide a brief description of each dataset.

- **EuroSat [24]**: is a dataset that specializes in land use and land cover classification, tailored for remote sensing and

¹We are working with an Aerospace group in collaboration with NASA where an LEO satellite prototype has been built and the computing devices are indeed edge devices like Raspberry Pi and Jetson Nano.

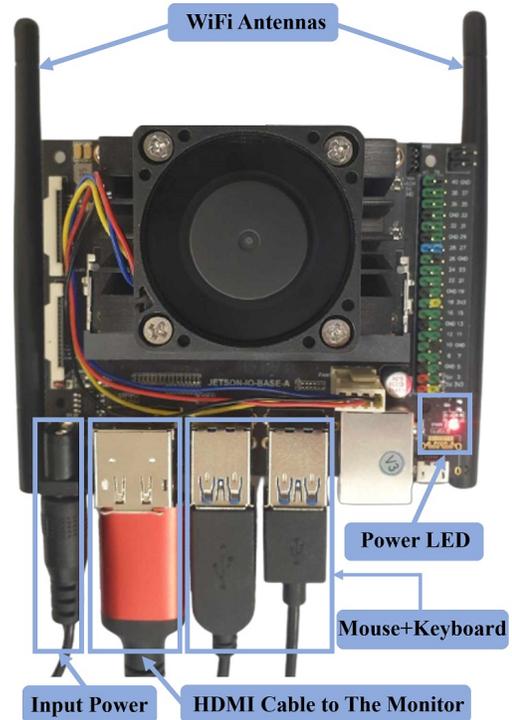


Fig. 3: Our implementation, training, and testing on Jetson Nano with experimental setup of essential peripherals.

Earth observation applications. This dataset encompasses high-resolution multispectral images captured by ESA’s Sentinel-2 satellite mission. It consists of 27,000 satellite images, split into 21,600 training samples and 5,400 test samples. EuroSat delineates land into ten distinct classes, including agricultural land, industrial areas, residential areas, forests, rivers, and others. Each image is precisely labeled with its corresponding land cover type and boasts a spatial resolution of 10 meters per pixel. For training our satellites, we utilize the VGG-16 model with trainable parameters of 8,413,194.

- **MNIST [16]**: is a dataset consisting of 70,000 grayscale images of handwritten numbers of size 28×28 pixels. For training, we use a convolutional neural network (CNN) with three convolutional layers, three pooling layers, and one fully connected layer with 437,840 trainable parameters.
- **CIFAR-10 & CIFAR-100 [17]**: are two datasets, with the former containing 10 different classes and the latter containing 100 classes. Each dataset comprises 60,000 color images, evenly distributed across the classes. The images are all 32×32 pixels in size and feature various animals and vehicles. For CIFAR-10, we use the same CNN architecture as MNIST, but with 798,653 trainable parameters. In the case of CIFAR-100, we use a CNN model constructed with six convolutional layers and two fully connected layers, totaling 7,759,521 training parameters.

For the EuroSat, MNIST, and CIFAR-10 datasets, we evenly distributed the data size of each dataset across all orbits, with each orbit containing the 10 classes. Consequently, each satellite within each orbit was assigned only images of one

TABLE III: Comparison of convergence time and accuracy under non-IID settings (near-polar Constellation).

	FL-LEO Approaches	Accuracy (%)			Convergence time (h)
		MNIST	CIFAR-10	CIFAR-100	
Sync FL	FedAvg [3]	79.41	70.68	61.66	60 (<i>PS</i> located anywhere)
	FedISL [7]	82.76	73.62	66.57	8 (<i>PS</i> located at the NP)
	FedISL [7]	61.06	52.11	47.99	72 (<i>PS</i> located anywhere)
	NomaFedHAP [8]	82.73	77.36	62.81	24 (<i>PS</i> located anywhere)
Async FL	FedAsync [25]	70.36	61.81	56.37	48 (<i>PS</i> located anywhere)
	FedSpace [6]	52.67	39.41	36.04	72 (satellite uploads some of its data)
	AsyncFLEO [14]	79.49	69.88	61.43	9 (sink satellite has sufficient visible period)
	Ours	94.64	89.69	82.65	2.13 (<i>PS</i> located anywhere)

class for training its local model. However, for the CIFAR-100 dataset, we assigned each satellite in each orbit 10 classes for training. We adopt this data distribution strategy to align with our goal of enabling each satellite to train a lightweight ML model.

B. Comparison with SOTA

We compare our approach with six other baseline methods: three synchronous FL methods, namely FedAvg [3], FedISL [7], and NomaFedHAP [8], as well as three asynchronous FL methods, including FedAsync [25], FedSpace [6], and AsyncFLEO [14]. However, our approach is the first to account for computation and energy consumption overheads, ensuring that each satellite can run the FL approach with its limited computational and storage resources.

Compared to the baselines, our approach achieves convergence in approximately 2 hours with accuracies of 94.64%, 89.69%, and 82.65% on MNIST, CIFAR-10, and CIFAR-100, respectively, outperforming all the baseline methods. The second-fastest approach is FedISL [7], synchronous FL approach, that attains accuracies of 82.76%, 73.62%, and 66.57% on MNIST, CIFAR-10, and CIFAR-100, respectively, after 8 hours. When compared to asynchronous FL approaches, AsyncFLEO is considered the third-fastest method to converge, achieving accuracies of 79.49%, 69.88%, and 61.43% on MNIST, CIFAR-10, and CIFAR-100, respectively. These results demonstrate the effectiveness of our approach, not only in making satellites applicable to realistic FL scenarios but also in achieving faster convergence with high accuracy. Table III summarizes the comparison of our approach with the rest of the baseline approaches.

C. In-Depth Evaluation on Eurostat Dataset

1) *Convergence Results*: To evaluate the performance of our approach on the EuroSAT dataset, we utilize four evaluation metrics: accuracy, precision, recall, and F1-score. We first evaluate the convergence of our proposed approach under the near-polar constellation scenario for *only two* global communication rounds (because our approach can converge within very few rounds). After the first round, our approach achieved an average accuracy of 95.741%. This first round required 124.86 minutes of communication time to ensure that at least one satellite per orbit received the global model, allowing each orbit to initiate the orbital training process. Furthermore, the training of orbital models was completed within a maximum training time of 27.0371 minutes across all orbits, resulting

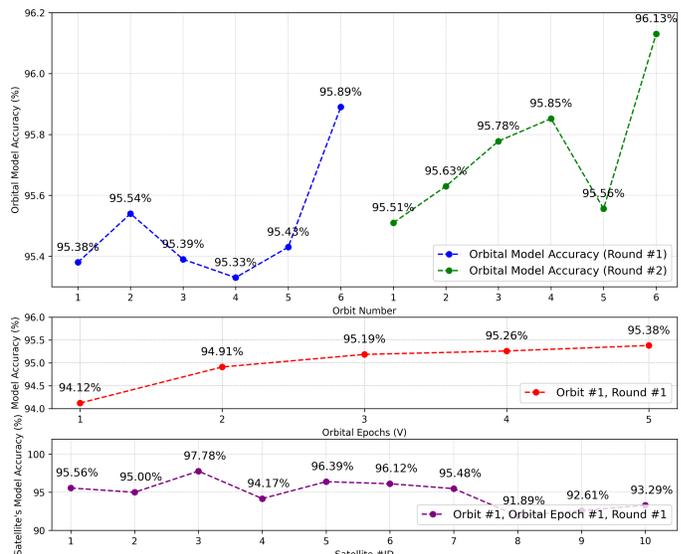


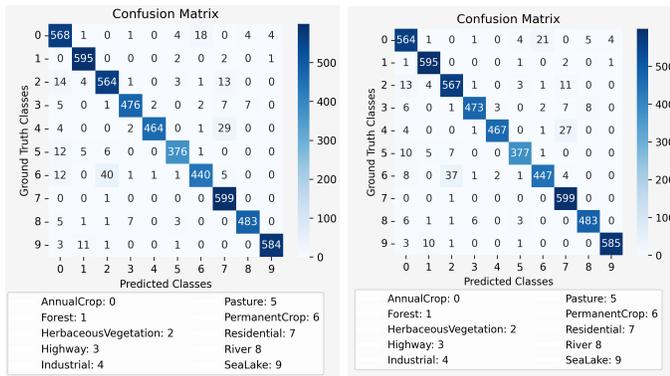
Fig. 4: Orbital model accuracy comparison across global communication rounds and orbital epochs. The upper subplot shows the accuracy of orbital models over two global communication rounds. The middle subplot illustrates the evolution of orbital model accuracy for orbit #1 in global round #1 over 5 orbital epochs. The lower subplot shows the training accuracy for each satellite model on orbit #1 in orbital epoch #1 and global round #1. The high accuracy is attributed to our DnC approach that converts complex tasks into binary tasks.

TABLE IV: Evaluation of our approach on EuroSat Dataset.

Class	Metric	# of images	Near-polar constellation (85°)				Inclined constellation (45°)			
			ACC(%)	PC (%)	RC (%)	F1 (%)	ACC(%)	PC (%)	RC (%)	F1 (%)
AnnualCrop		600	97.63	92.06	94.67	93.34	98.39	91.71	94.0	92.84
Forest		600	98.19	96.73	98.67	97.69	99.52	96.59	99.17	97.86
HerbaceousVegetation		600	99.32	93.05	93.67	93.36	98.5	92.12	94.5	93.33
Highway		500	99.61	97.74	95.0	96.35	99.31	97.93	94.60	96.24
Industrial		500	98.89	98.95	93.80	96.30	99.30	98.94	93.40	96.09
Pasture		400	99.23	96.42	94.25	95.32	99.33	96.67	94.25	95.44
PermanentCrop		500	97.46	95.14	90.0	92.50	98.54	94.50	89.40	91.88
Residential		600	99.01	91.45	99.83	95.46	99.04	92.15	99.83	95.84
River		500	99.56	98.17	96.60	97.38	99.44	97.37	96.60	96.99
SeaLake		600	99.11	98.66	98.0	98.33	99.63	99.15	97.50	98.32

in an overall communication and training time of 151.891 minutes (~ 2.53 hours).

After the second round, our approach's accuracy showed a marginal increase to 95.778%, with the total time for both rounds amounting to 277.167 minutes (~ 4.619 hours). The upper subplot of Fig. 4 shows the accuracy of the orbital model for each orbit after 5 orbital epochs for two communication rounds. The middle subplot of Fig. 4 illustrates the improvement of retraining the orbital model accuracy for a single orbit for each orbital epoch within the first global round. In addition, the lower subplot of Fig. 4 displays the individual training accuracy of each satellite that trains a binary-class classifier before aggregating them into an orbital model. After averaging them and testing the orbital model's accuracy, it yields an accuracy of 94.12%. These results underscore the effectiveness of our orbital retraining approach in significantly



(a) Near-polar constellation (85°). (b) Inclined constellation (45°).

Fig. 5: Confusion matrix that compares 10 predicted and ground-truth classes for 5400 test images.

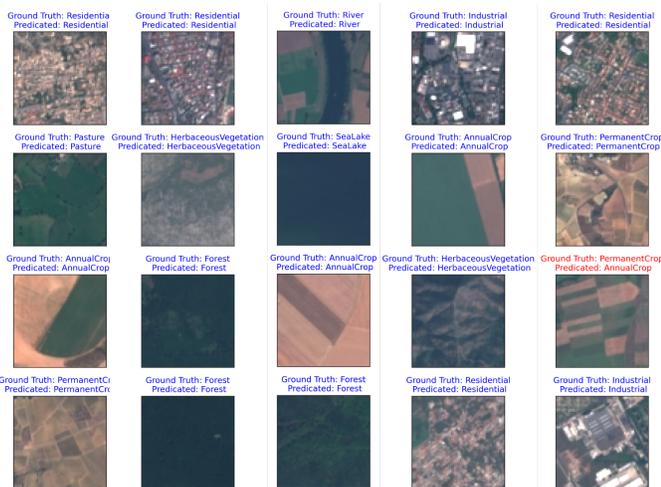


Fig. 6: Twenty randomly selected images from a Eurostat test set of 5400 samples, illustrating the predicted vs. ground truth labels. Blue and red color represent correct and incorrect predictions, respectively.

reducing the required number of communication rounds to just a few iterations and expediting convergence.

Table IV summarizes our approach’s evaluation results under various metrics for each class after only two global training rounds. Furthermore, in Fig. 5, we present the confusion matrix, which visually illustrates the relationship between actual ground truth classes and predicted classes, providing insights into the efficacy of our approach. Finally, in Fig. 6, we showcase 20 randomly selected images from a pool of 5400 test samples to demonstrate the effectiveness of our approach. Remarkably, among these randomly selected images, only one is misclassified. This outcome further underscores the effectiveness of our approach in accurately classifying images.

2) *Overheads and Energy Consumption Results:* We analyze the required computation and communication overheads for running our proposed approach on edge devices, specifically the Jetson-Nano, to determine its affordability for deployment on satellites with limited resources. Our findings reveal

TABLE V: Comparison of computation and communication overheads of our approach in different settings.

(a) Computation overhead.		(b) Communication overhead.	
Model	FLOPS (G)	Dataset	Size (MB)
CNN using MNIST	11.91	MNIST	0.437
CNN using CIFAR-10	15.58	CIFAR-10	0.798
CNN using CIFAR-100	28.13	CIFAR-100	7.76
VGG-16 using EuroSat	43.84	EuroSat	26.68

that GPU usage during training ranges from 17% to 58%, with corresponding energy consumption levels ranging between 1.38 and 2.25 watts for each local model, demonstrating the model’s lightweight nature.

In Table V, we present both the computation and communication overheads of our approach for various models trained on different datasets, where each satellite is assigned a reduced data size belonging to its target class. To assess computation overheads, we use floating-point operations per second (FLOPS) as a metric to measure the computation cost. In Table V.a, we can observe that the highest computation cost incurred by our approach is 43.84 GFLOPS. This is significantly lower than the computational capacity of the Jetson-Nano, which can handle up to 472 GFLOPS.

On the other hand, we evaluate the communication overhead of our approach in terms of the orbital model parameters that need to be uploaded to the $\mathcal{P}S$. In Table V.b, we present the communication cost for all the training datasets used to evaluate our approach, with a maximum of 26.68 MB. Considering that a satellite has a short visible window of 5 minutes and needs to upload the orbital model of size 26.68 MB, this requires a small data rate of 0.69 Mb/s, which is significantly lower than the available data rate in Satcom. In summary, our approach exhibits efficient computation and communication overheads, making it suitable for space-edge computing scenarios and pervasive computing.

VI. CONCLUSION

This paper represents a significant stride towards enabling pervasive space AI by seamlessly integrating FL with SEC in the form of LEO satellites. Our innovative approach comprises (1) *personalized learning via DnC*, which reduces the complexity of learning at SEC by transforming multi-class training into binary training, and (2) *orbital model retraining*, which aggregates and retrains an orbital model in multiple orbital epochs in a back-and-forth manner among all the satellites within an orbit, before sending it to the $\mathcal{P}S$. Our experiments with a real satellite imagery dataset validate the effectiveness of our approach, converging rapidly within only 4.619 hours while achieving a high classification accuracy of 95.778%. Notably, it significantly reduces energy consumption for training local models at each satellite to as low as 1.38 watts, computation overhead to 11.91 GFLOPs, and communication overhead to 0.437 MB. These results demonstrate the suitability and applicability of our approach in real SEC scenarios with constrained resources.

REFERENCES

- [1] A. Perez-Portero, J. F. Munoz-Martin, H. Park, and A. Camps, "Airborne gnss-r: A key enabling technology for environmental monitoring," *IEEE Jnl. of Sel. Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 6652–6661, 2021.
- [2] M. Y. Abdelsadek, A. U. Chaudhry, T. Darwish, E. Erdogan, G. Karabulut-Kurt, P. G. Madoery, O. B. Yahia, and H. Yanikomeroglu, "Future space networks: Toward the next giant leap for humankind," *IEEE Transactions on Communications*, vol. 71, no. 2, pp. 949–1007, 2022.
- [3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017, pp. 1273–1282.
- [4] S. Wang and Q. Li, "Satellite computing: Vision and challenges," *IEEE Internet of Things Journal*, 2023.
- [5] H. Chen, M. Xiao, and Z. Pang, "Satellite-based computing networks with federated learning," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 78–84, 2022.
- [6] J. So *et al.*, "Fedspace: An efficient federated learning framework at satellites and ground stations," *arXiv preprint arXiv:2202.01267*, 2022.
- [7] N. Razmi *et al.*, "On-board federated learning for dense LEO constellations," in *IEEE ICC*, May 2022.
- [8] M. Elmahallawy, T. Luo, and K. Ramadan, "Communication-efficient federated learning for LEO satellite networks integrated with HAPs using hybrid NOMA-OFDM," *IEEE Journal on Selected Areas in Communications*, pp. 1–1, 2024.
- [9] C.-Y. Chen, L.-H. Shen, K.-T. Feng, L.-L. Yang, and J.-M. Wu, "Edge selection and clustering for federated learning in optical inter-LEO satellite constellation," *arXiv preprint arXiv:2303.16071*, 2023.
- [10] M. Elmahallawy and T. Luo, "Optimizing federated learning in LEO satellite constellations via intra-plane model propagation and sink satellite scheduling," in *ICC 2023 - IEEE International Conference on Communications*, 2023, pp. 3444–3449.
- [11] M. Elmahallawy, T. Luo, and M. I. Ibrahim, "Secure and efficient federated learning in LEO constellations using decentralized key generation and on-orbit model aggregation," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*, 2023.
- [12] N. Razmi, B. Matthiesen, A. Dekorsy, and P. Popovski, "Ground-assisted federated learning in LEO satellite constellations," *IEEE Wireless Communications Letters*, 2022.
- [13] P. Wang, H. Li, and B. Chen, "FL-task-aware routing and resource reservation over satellite networks," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 2382–2387.
- [14] M. Elmahallawy and T. Luo, "AsyncFLEO: Asynchronous federated learning for LEO satellite constellations with high-altitude platforms," in *2022 IEEE International Conference on Big Data (BigData)*, 2022, pp. 5478–5487.
- [15] L. Wu and J. Zhang, "FedGSM: Efficient federated learning for LEO constellations with gradient staleness mitigation," *arXiv preprint arXiv:2304.08537*, 2023.
- [16] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.
- [17] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-10 (canadian institute for advanced research)," *URL http://www.cs.toronto.edu/kriz/cifar.html*, vol. 5, no. 4, p. 1, 2010.
- [18] Q. Ouyang, N. Ye, J. Gao, A. Wang, and L. Zhao, "Joint in-orbit computation and communication for minimizing download time from LEO satellites," *IEEE Transactions on Mobile Computing*, 2023.
- [19] M. Pritt and G. Chern, "Satellite image classification with deep learning," in *2017 IEEE applied imagery pattern recognition workshop (AIPR)*. IEEE, 2017, pp. 1–7.
- [20] J. Östman, P. Gomez, V. M. Shreenath, and G. Meoni, "Decentralised semi-supervised onboard learning for scene classification in low-earth orbit," *arXiv preprint arXiv:2305.04059*, 2023.
- [21] B. Denby and B. Lucia, "Orbital edge computing: Nanosatellite constellations as a new class of computer system," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 939–954.
- [22] J. G. Walker, "Satellite constellations," *Journal of the British Interplanetary Society*, vol. 37, p. 559, 1984.
- [23] "Software for digital mission engineering and systems analysis," <https://www.ansys.com/products/missions/ansys-stk>, Jan. 2023.
- [24] P. Helber, B. Bischke, A. Dengel, and D. Borth, "Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 7, pp. 2217–2226, 2019.
- [25] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, 2019.