ORBITALBRAIN: HARNESSING DISTRIBUTED TRAINING IN SPACE UNDER STRINGENT PHYSICAL CONSTRAINTS

ABSTRACT

The rapid increase in earth observation nanosatellites generates massive near real-time data, opening new frontiers to address global challenges. Many of these challenges require near real-time insights using up-to-date machine learning (ML) models. However, the downlink bandwidth does not scale with the data volume, resulting in days or weeks of turnaround time to update ML models on the ground. In this work, we develop the first simulation platform for ML training in space that respects the physical constraints in satellites, validated through an actual satellite deployment. Based on the findings from our simulation platform, we introduce OrbitalBrain, an efficient in-space ML training algorithm that leverages limited and predictable satellite resources to intelligently balance data transfer, model aggregation, and local training. Our extensive evaluations demonstrate that OrbitalBrain achieves $1.52\times$-$12.4\times$ speedup in time-to-accuracy with 1.9%-49.5% final model accuracy improvement compared to state-of-the-art ground-based or federated learning baselines. Furthermore, our approach is complementary to satellite imagery capturing and downloading, enhancing the overall efficiency of satellite-based applications.

1 INTRODUCTION

The rapid expansion of low Earth orbit (LEO) nanosatellite constellations for earth observation (EO) applications creates new opportunities to address various global issues, such as climate change, pandemic response, and disaster relief (Aragon et al., 2018; Barmpoutis et al., 2020; Chen et al., 2020a; Shukla et al., 2021; Franch-Pardo et al., 2020). These satellites can capture high-resolution, hyperspectral imagery of the planet in near real-time, which can help derive valuable and timely insights using ML. However, the ML models currently need to be trained on the ground, which is significantly constrained by the limited downlink bandwidth of satellite-ground communication.

The downlink bandwidth is constrained by several factors. First, LEO satellites can only transmit data to the ground when they pass over a ground station, which happens a few times a day (Denby & Lucia, 2020). Second, ground stations are costly and difficult to scale due to regulatory issues (Vasisht et al., 2021). Third, weather can affect the quality and reliability of the data transmission (Denby & Lucia, 2020). Fourth, the increasing number of satellites competing for the same wireless spectrum leads to interference (Barbulescu et al., 2004; Wang et al., 2011).

As a result, a diminishing fraction of the satellite data can be downloaded to the ground for ML training. For example, the current largest commercial EO constellation (Planet's Doves) can only download 5 TB of satellite imagery per day, which covers only 9% of the data that could have been captured by this constellation (dov, 2021). This problem is much worse for emerging applications that require finer resolutions and more spectral bands than currently used, generating much larger datasets for the same coverage (sky, 2020; WorldVu Satellites Limited, 2018). This means that the ML models trained on the ground will be lagging behind and inaccurate, as they miss most of the recent data. For instance, an ML model that detects wildfires or floods (Barmpoutis et al., 2020; NASA-IMPACT, 2021) needs to be updated frequently and quickly to account for the dynamic changes in the environment. Likewise, an ML model that monitors crop health (Maimaitijiang et al., 2020) or urban growth (Burke et al., 2021; Mundhenk et al., 2016) needs to be refined with new data to improve its precision and reliability.

Recent studies (Chen et al., 2022; Razmi et al., 2022; So et al., 2022) explore the potential of harnessing the computational power of satellites for ML training. This possibility is becoming increasingly feasible as satellite deployments integrate computing devices capable of running deep neural networks (Giuffrida et al., 2022). Most of these proposals utilize federated learning (FL) (McMahan et al., 2017), in which satellites (clients) and ground stations (servers) collaboratively train ML models without sharing the training data. However, these proposals often make simplistic assumptions about the physical constraints of satellites, such as those related to energy (assuming infinite energy for training), storage (assuming the constant ability to revisit "archived" training data), and data similarity (not differentiating between data samples). Furthermore, the restrictions on data sharing due to privacy concerns in FL are unneces-

Preliminary work. Under review by the Machine Learning and Systems (MLSys) Conference. Do not distribute.

sary in this context. Consequently, these works may serve as strawman solutions in this context, and their effectiveness remains uncertain when accounting for all the constraints present in real satellites.

Our Goal and Key Findings. In this study, we take the first step towards understanding the system challenges of distributed ML training in space by examining the physical constraints of satellites. To achieve this, we employ a nanosatellite constellation simulator (Microsoft Research, 2023) that is verified with a real satellite deployment. We integrate this simulator with an FL platform (Dimitriadis et al., 2022) to build a *first-of-its-kind* simulator for ML training in space that accounts for physical constraints, such as orbital dynamics, communication, energy, storage, and data availability over time. By conducting our study with real-world satellite datasets (Christie et al., 2018; Zhu et al., 2019) and constellations (spi, 2017; Safyan, 2020), we make several key findings: (1) These physical constraints are heterogeneous, interdependent, and time-varying; (2) when considering these constraints, existing in-space training solutions experience significant degradation in model convergence rates and do not surpass ground-based training; and (3) the importance of training data exhibits a time-varying dimension, influenced by the physical constraints.

Technical Challenges. At its core, our problem can be formulated as maximizing the ML training convergence rate based on satellites' decisions regarding (a) local training, (b) data transfer, and (c) model aggregation at each time step, while adhering to their physical constraints. We consider optical inter-satellite links (ISLs) in our formulation.[1] Unfortunately, solving this problem is computationally intractable due to the interdependence of decisions across different satellites and time steps, as well as the absence of an analytical model to accurately estimate their impacts on ML training convergence without actually executing it. Moreover, as our findings reveal, the time-varying and constellation-specific constraints introduce additional complexity to this already challenging problem.

Solution. We introduce OrbitalBrain, an efficient and practical solution for distributed ML training in space. OrbitalBrain capitalizes on the predictability of satellite paths, sunlight exposure, and resource availability. By combining these predictions with local training statistics, we can estimate the *utility* of various operations, such as local training, data transfer, and model aggregation. OrbitalBrain comprises a two-stage process: first, it determines whether to utilize ISLs for aggregating local models across satellites, which involves balancing the trade-off between local model staleness and the additional energy/latency costs. Next, it assesses whether each satellite should allocate its remaining energy and connectivity resources to local training and data transfer across satellites, which requires navigating the trade-off between data distribution and training progress. Our utility functions draw insights from general ML training and our findings on physical constraints. Notably, they are agnostic to specific ML tasks and satellite configurations, making them adaptable and versatile for various scenarios.

[1] Although many companies support ISLs (Werner, Retrieved in 2022; Zech et al., 2019; Jewett, Retrieved in 2022; Handley, 2018), they are not mandatory in our formulation.

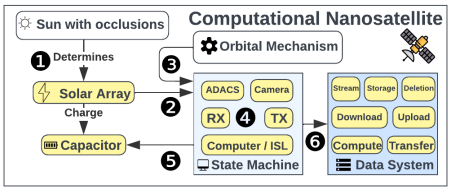


Figure 1. Components of a computational nanosatellite.

Evaluation Highlights. We evaluate OrbitalBrain's performance with two satellite constellations (spi, 2017; Safyan, 2020) and two space ML tasks (Christie et al., 2018; Zhu et al., 2019). We compare it extensively with five state-of-the-art (SOTA) centralized/FL baselines (Denby & Lucia, 2020; Wang et al., 2019; Bonawitz et al., 2019; Nguyen et al., 2021; So et al., 2022). We show OrbitalBrain achieves $1.52\times$-$12.4\times$ speedup in time-to-accuracy with 1.9%-49.5% better final model accuracy. We further demonstrate OrbitalBrain performs consistently under various cloud obstruction, imagery data sizes/resolutions, and participating satellites. It also supports collaborative training between satellites and ground stations.

Contributions. We make the following contributions:

- We demonstrate and characterize the challenges posed by physical constraints for distributed ML training in space.
- We formulate the problem for ML training in space under physical constraints, focusing on the trade-off among data transfer, model aggregation, and local training.
- We introduce a general technique to estimate the utility of various satellite operations and a two-stage decision process to allocate satellite resources for ML training.

To our knowledge, this is the first simulator for distributed ML training in space. We intend to release the simulator and OrbitalBrain's source code to promote further research on this emerging problem.

2 PRELIMINARIES

We outline a standard computational nanosatellite's architecture and constraints (§2.1), and discuss their impact on in-space ML training approaches (§2.2).

2.1 Physical Constraints of Nanosatellites

Computational Nanosatellite Architecture. An emerging satellite class, computational nanosatellites add onboard computing to small and cost-effective satellites, enabling real-time insights in space (Denby & Lucia, 2020; Giuffrida et al., 2022). Figure 1 showcases the architecture of

a computational nanosatellite, which includes energy, state machine (SM), and data systems. Our energy model relies on a public orbital simulator, `CosmicBeats` (Microsoft Research, 2023), chosen for its verified communication, power, and data storage models through an actual nanosatellite launch (Singh et al., 2024). The model comprises solar array charging (①), state machine initiation (②), energy-driven operation transitions (③, ④), available capacitor power assessment (⑤), and operation viability evaluation based on energy and communication constraints (⑥).

We identify three physical constraints of nanosatellites that are crucial for space-based and ground-based training, though prior work significantly simplifies these constraints.

(1) Energy. Satellites depend on solar power, leading to constrained and variable energy availability over time, as seen in the energy trace of a `Planet` satellite (in-the-sky.org, 2022). All satellite operations, including capturing images, communicating with ground or other satellites, and running or training ML models, must consider energy availability.

(2) Link States. A nanosatellite and a ground station can communicate only when the satellite passes over the ground station (GS-sat connections). Similarly, ISLs are viable only when two satellites are close enough and have a line-of-sight (cross-sat connections). We assess the average number of active links over 5-minute windows across satellites and the distribution of link bandwidths (Figure 13(a)-13(d) in Appendix A). Our findings indicate: (a) the number of active links significantly decreases when incorporating energy constraints (energy and link constraints are *dependent*); and (b) the median bandwidth for both GS-sat links and ISLs is around 100 Mbps, but ISLs exhibit higher variance due to the constantly changing relative positions. These observations suggest that simplistic orbital models cannot accurately determine link liveness and bandwidth availability.

(3) Data Availability and Heterogeneity. Satellites have limited storage capacity for imagery, and different satellites can possess significantly diverse images as they follow unique trajectories. In our study emulating a public real-world dataset, `fMoW` (Christie et al., 2018), we observe time-varying data availability, and the number of available labels also fluctuates over time. This high skew poses challenges for FL (McMahan et al., 2017; Zhao et al., 2018; Hsieh et al., 2020) as it biases local training, making global model convergence difficult.

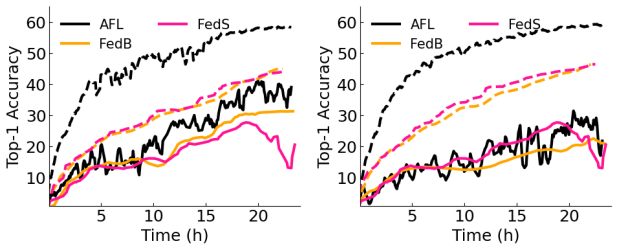
2.2 Effects of Physical Constraints on ML Training

Our results highlight the potential impact of energy, communication, and data availability on training. We explore how these constraints influence SOTA approaches in both ground-based and space-based training scenarios.

Impact on Ground-Based Training. Modern constellations depend on the `BentPipe` architecture (Denby & Lucia, 2020; Wertz et al., 1999), which involves sending control commands to satellites and transmitting data back to Earth. However, downloading all high-resolution imagery to the ground is becoming increasingly difficult (see §1) due to limited and intermittent bandwidth (Vasisht et al., 2021).

Table 1. `BentPipe`'s performance under varying image sizes with the same imaging frequency

Image Size	Ideal	100 MB	200 MB	300 MB
# of Images	363,563	111,737	76,586	42,384
Percentage	100%	30.7%	21.1%	11.7%
Accuracy	59.6%	50.9%	44.1%	43.4%



(a) `Planet` (Escher, 2018) (b) `Spire` (spi, 2017)

Figure 2. Training curves for SOTA FL proposals with all (solid) or simple (dashed) constraints.

To illustrate this, we model `Planet`'s constellation (Safyan, 2020) along with the `fMoW` (Christie et al., 2018) dataset, considering energy, link, and data availability constraints (§2.1). In this case, a single image is 200 MB (resp. 100 MB), which aligns with `Planet` Dove nanosatellites' specifications. We find that only $15k$ (resp. $33k$) out of $254k$ (resp. $531k$) images, or 3.3 TB, can be downloaded to the ground per day, a figure similar to the number reported by `Planet` (Devaraj, 2021). This means that if the constellation were at its full imaging capability, it would take 16 *days* to download the entire dataset, making it difficult for time-sensitive applications to achieve update-to-date ML models (Barmpoutis et al., 2020; Chen et al., 2020a; NASA-IMPACT, 2021; Franch-Pardo et al., 2020). This issue is further exacerbated for applications requiring higher-resolution images. As these satellites also face onboard storage constraints (e.g., 360 GB for FLOCK(Planet, 2022b)), they must delete older images. We demonstrate the impact on the training performance of the `BentPipe` architecture in Table 1. We find that `BentPipe` struggles when it has fewer images and when real-world physical constraints are imposed (compared to allowing all data to be sent to the ground immediately).

Impact on Space-based Training. We examine the effects of these physical constraints using three SOTA benchmarks: `AsyncFL` (AFL) (Wang et al., 2019), `FedBuff` (FedB) (Nguyen et al., 2021), and `FedSpace` (FedS) (So et al., 2022). We compare these methods with the `fMoW` dataset and two different satellite constellations (`Planet` (CelesTrack, Retrieved in 2022a) and `Spire` (CelesTrack, Retrieved in 2022b)). We consider two scenarios:

one with simple constraints (i.e., GS-sat contact window determined by orbital dynamics (So et al., 2022; Chen et al., 2022; Razmi et al., 2022)) and another with all constraints on energy, bandwidth, and data simultaneously. All three benchmarks experience significant performance degradation when subjected to the more comprehensive practical constraints (Figure 2): the training becomes slow and unstable. This degradation is due to the limited and heterogeneous local training progress resulting from energy constraints; the local model staleness caused by both link and energy constraints; and local model bias stemming from data availability constraints (see Appendix C).

Summary. We find that the physical constraints are time-varying and interconnected. These constraints present significant challenges for both ground-based training and distributed training using computational nanosatellites. It is essential to thoughtfully consider these constraints when developing an effective ML training solution. Fortunately, most of these resource limitations can be predicted using the satellites' orbital dynamics and physical mechanics. This understanding informs our design approach.

3 OrbitalBrain: System Design

Problem Formulation. Without loss of generality, we frame the issue of in-space training under physical constraints as an optimization problem, aiming to maximize the ML convergence rate based on each satellite's decisions regarding local training, data transfer, and model aggregation (MA). Appendix B provides a detailed description of the formulation. Our analysis reveals that this optimization problem is computationally intractable due to its vast search space and the unclear relationship between different decisions and the accumulated effect on global model accuracy. Moreover, the problem space varies significantly with different ML tasks and satellite configurations, making it difficult to design a general and effective solution.

System Overview. To reduce the search space for optimizing this problem, we propose a solution that disentangles the resource allocation for the computation, data transfer, and model aggregator in an approximately greedy manner. Figure 3 shows the system overview of our solution, OrbitalBrain, which consists of three main components, the performance profiler (§3.1), the model aggregator (§3.2), and the data transferer (§3.3). The performance profiler calculates the compute utility of each satellite based on the statistics of its local training data. The model aggregator then uses the compute utility to determine the feasibility and necessity of inter-sat MA by considering the trade-off between aggregation gain (e.g., per-satellite model staleness, training accuracy) and execution energy/latency cost. Finally, the data transferer considers the utility of sharing data with other satellites, which is then compared against the local compute utility to determine the operation of each satellite (i.e., transfer data or local compute). We describe these three components in detail here.

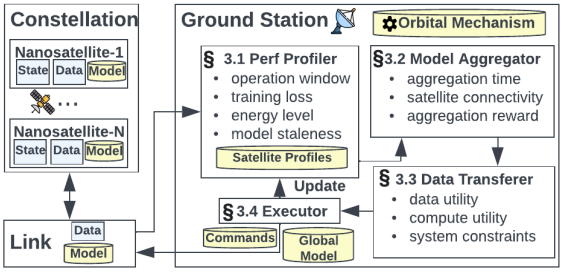


Figure 3. Overview of OrbitalBrain's ML framework.

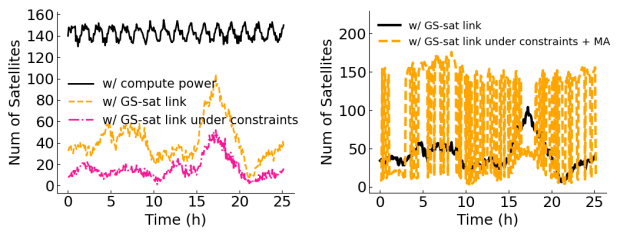
3.1 Guided Performance Profiler

OrbitalBrain aims to make the training decision for satellites based on their profiles. These profiles are built by retrieving information from the satellites connected to the ground at each scheduling window and performing estimations based on the orbital mechanism for the rest of the satellites (Vasisht et al., 2021).

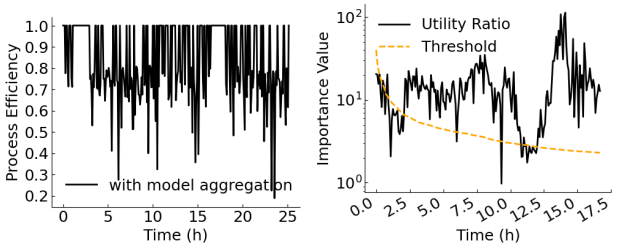
At the beginning of each scheduling window (e.g., every 5 mins), the ground station retrieves the information from the latest connected satellites, including the imagery data samples, the local model updates, their training traces, energy status, and data streaming/storage. It updates their profiles and estimates those unconnected satellites based on the previous feedback and orbital mechanisms. With $\mathcal{S}$ denoting the set of satellites in the constellation, the ground station also records the local model staleness $\eta_s \in H$ for each satellite $s \in \mathcal{S}$, which indicates the number of training windows since its last connection with the ground (So et al., 2022). To quantify satellite s's contribution of local compute to the global ML model, OrbitalBrain calculates its compute utility $\mathrm{Util}_{\mathrm{comp}}$ based on its profile as

$$\mathrm{Util}_{\mathrm{comp}}(s) = n_s^{\mathrm{comp}} \sqrt{\frac{(\eta_s+1)^{-a}}{|\mathcal{D}_s|} \sum_{d \in \mathcal{D}_s} \mathrm{Loss}(d)}, \quad (1)$$

where $\mathcal{D}_s$ is the set of data samples used for local computation in the current window and n_s^{comp} is an estimation (based on the energy trace) of the number of data samples to be computed in future rounds. Intuitively, the utility decreases with higher staleness η_s since a staler model should contribute less to the global model, with a decay factor a (default 1). Additionally, the utility increases with an increase in the average training loss of the data samples $\frac{1}{|\mathcal{D}_s|}\sum_{d\in\mathcal{D}_s}\mathrm{Loss}(d)$. Essentially, whenever the data samples in $\mathcal{D}_s$ are *not* well explained by the current model (high training loss) then it is more valuable to use these data samples for updating our model. This is in line with the notion of importance sampling used in ML (Katharopoulos & Fleuret, 2018; Alain et al., 2016; Zhao & Zhang, 2015), where a more important client is typically determined by



(a) Limited GS-sat connections (b) MA enables GS-sat



(c) MA reduces efficiency (d) MA utility varies with time

Figure 4. MA and its trade-off between extra time and energy on inter-satellite communications and additional ground connectivity.

a larger gradient norm or training loss (Lai et al., 2021). Finally, the utility is scaled by n_s^{comp} since there is more value to update the local model if we anticipate that this satellite will see many samples in future rounds.

3.2 Model Aggregation across Satellites

Based on the performance profiling, OrbitalBrain determines the feasibility and necessity of aggregating models across satellites over ISLs. By communicating through other satellites, this mechanism effectively enables more satellite connections to the ground stations.

Figure 4(a) shows the GS-sat connections of the `Planet`'s constellations for a whole day. As time evolves, the number of satellites connected to the ground (orange) fluctuates between 10 at 21h and 100 at 18h. Unfortunately, only approximately $1/3$ of these satellites (pink) can establish the GS-sat connections if we incorporate the energy constraints (see §2.1). As a result, most of the energy-sufficient satellites (black) can only perform local computations but have no GS-sat link to download their model updates to the ground or retrieve the updated global model from the ground, resulting in staleness in model updates. With the ISLs between each pair of satellites, OrbitalBrain enables more GS-sat connections by aggregating models across satellites and then downloading them to the ground. Figure 4(b) shows that OrbitalBrain downloads more local model updates from the space to the ground by almost $4\times$ when enabling MA intermittently for 4h-15h and 19h-25h. However, MA induces extra time and energy costs, which reduce the computation efficiency (Figure 4(c)). For example, at 24h, only 20% of data samples can be processed because most energy is used for inter-satellite MA. The efficiency of a given satellite s is denoted by ϵ_s and these are collected in the vector $\boldsymbol{\epsilon}$. Besides, we also need to consider the latency of the MA to make sure it can be done within a scheduling window.

To determine whether MA is feasible and valuable in a given scheduling window, OrbitalBrain follows a threshold-based decision-making strategy where it compares the MA-enabled rewards with the extra cost in energy and time associated with the aggregation. Given the performance profiler in §3.1, we first get a set of computing satellites $\mathcal{S}_c$ (black in Figure 4(a)), with the $\mathcal{S}_{cc}$ indicating those computing ones with the communication ability to the ground (pink in Figure 4(a)). We then build a shortest-path tree for $\mathcal{S}_c$, with the root being the satellite that has the most active ISLs in $\mathcal{S}_c$. The rationale behind this is to connect more satellites for aggregation via ISLs. Given the ISL status for each pair of satellites in $\mathcal{S}_c$, we estimate the latency for MA over the shortest-path tree, which must be shorter than the scheduling window to be feasible. If feasible, we determine whether it is valuable to perform MA by testing the following conditions.

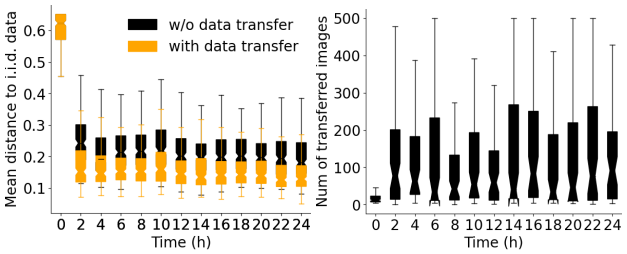
$$\frac{\text{Util}(\mathcal{S}_c \setminus \mathcal{S}_{cc}, \boldsymbol{\epsilon})}{\text{Util}(\mathcal{S}_{cc}, \mathbf{1})} > \theta_0 (t+1)^{-b}, \tag{2}$$

$$\text{where } \text{Util}(\mathcal{S}, \boldsymbol{\epsilon}) = \left(\frac{\sum_{s \in \mathcal{S}} \eta_s}{|\mathcal{S}|} + 1 \right)^{-a} \sum_{s \in \mathcal{S}} \epsilon_s \text{Util}_{\text{comp}}(s).$$

Notice that $\mathcal{S}_c \setminus \mathcal{S}_{cc}$ is a set containing the satellites that have sufficient energy but no GS-sat connections. Essentially, in (2), we compare the utility of the satellites that would be reachable via MA (with efficiency $\boldsymbol{\epsilon}$) with the utility of only considering the satellites with direct ground connection in $\mathcal{S}_{cc}$. Naturally, in this second case, we would not perform MA, thus, the associated efficiency is represented by a vector of all ones. In both cases, the utility of a set of satellites is given by the sum of individual compute utilities as given by (1), weighted by their corresponding efficiency ϵ_s and scaled by the inverse of the average staleness of the set of satellites under consideration. For MA to be beneficial, we want the ratio between these utilities to be larger than the threshold $\theta_0 (t+1)^{-b}$. We use a time-varying threshold with an initial θ_0 (default 40) and a decay factor b (default 0.5), which decreases with the time window t. The time-varying nature of the threshold responds to the empirical observation of the value of MA being larger at later stages of the training process. Figure 4(d) illustrates the utility ratio in (2) and the time-varying threshold based on `Planet`'s constellations.

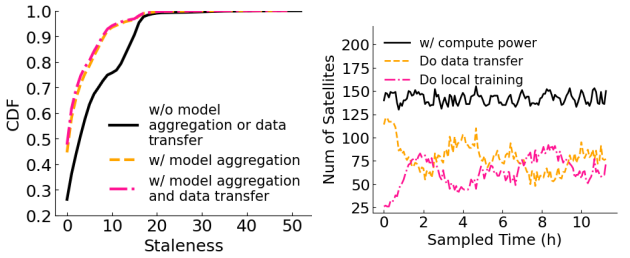
3.3 Enable Data Transfer for IID Data

To fully utilize the satellites' resources, OrbitalBrain also allocates satellites to transfer their data to nearby satellites to reduce local model bias due to non-i.i.d. data distributions. To do this, OrbitalBrain first retrieves the data distribution of each satellite from its performance profiler, including the number of samples, label distribution, and energy traces. It then calculates the data utility at each satellite. Intuitively, a satellite s has a larger data utility if it has more i.i.d data



(a) Distance to i.i.d data (b) Transferred images

Figure 5. OrbitalBrain makes each satellite's data near i.i.d over time by transferring images across them.



(a) Reduce the model staleness (b) Exploit compute resources

Figure 6. OrbitalBrain optimizes the data distribution and local model staleness.

and sufficient energy for local computation,

$$\text{Util}_{\text{data}}(s) = n_s^{\text{collect}}(1 - F_{\text{dis}}(\mathbf{l}_s))^{n_s^{\text{comp}}} \quad (3)$$

where $F_{\text{dis}}(\mathbf{l}_s)$ indicates the distance between its label distribution $\mathbf{l}_s$ and the i.i.d label distribution. n_s^{collect} and n_s^{comp} are the numbers of data samples to be collected and computed under energy constraints (estimated from the state machine in Figure 1). The transfer utility for inter-sat communication depends on the destination satellite and its data distribution. For example, a satellite s with less energy but a large number of data samples on balanced labels $\mathbf{l}_s$ can transfer its data to a destination satellite s' that has sufficient energy but non-i.i.d data $\mathbf{l}_{s'}$. As such, we formulate the transfer utility for inter-sat data communication as

$$\text{Util}_{\text{tr}}(s, s') = F_{\text{dis}}(\mathbf{l}_{s'}) - F_{\text{dis}}(n_{ss'}^t \frac{\mathbf{l}_s}{|\mathbf{l}_s|} + \mathbf{l}_{s'}), \quad (4)$$

where $n_{ss'}^t$ indicates the number of **Transferred** data samples from s to s' (Appendix A) . In (4), we see that the transfer utility is large if s' has a significantly smaller distance to the ideal i.i.d. data distribution after receiving the transferred data samples from s.

Based on the data and transfer utilities, we compute the link utility for each pair of satellites

$$\text{Util}_{\text{link}}(s, s') = \begin{cases} \text{Util}_{\text{tr}}(s, s')\,\text{Util}^*_{\text{data}}(s') & \text{if } s \neq s' \\ \xi\,\text{Util}_{\text{data}}(s') & \text{if } s = s' \end{cases} \quad (5)$$

where $\text{Util}^*_{\text{data}}(s')$ represents the data utility of satellite s' after receiving the data samples from s. ξ is a self-computation utility parameter (default 20). OrbitalBrain selects the link (s, s') with the largest utility iteratively for data transfer ($s \neq s'$) or local computation ($s = s'$) and terminates when all remaining link utilities are negative. We allocate all remaining satellites to local computation since the data transfer cannot make the data distribution of the destination closer to being i.i.d (a negative $\text{Util}_{\text{tr}}(s, s')$).

Figure 5 verifies the effectiveness of OrbitalBrain's data transfer strategy. With data transfer, the distance to the i.i.d. data distribution for each of the `Planet` constellations decreases significantly over time, with a lower median value (from 0.21 to 0.12) in Figure 5(a). Figure 5(b) shows the number of transferred images across those satellites at each scheduling window, spanning from 20 to 500, which is determined by the bandwidth of ISLs and the energy constraints.

3.4 Putting It All Together

We put all components together in Algorithm 1. In each scheduling window t of length T_t (e.g., 5 mins), the performance profiler first retrieves the satellite information for profiling, including the training loss, staleness H, energy trace E, on-board storage M, label distribution $\mathbf{l}$, and the link topology G (line 2). It updates the compute utility based on Eq. (1) (line 3) and detects each satellite's status under the energy constraints (line 4). Given the ISL connectivity topology G_{ISL}, OrbitalBrain checks the MA feasibility and resulting compute efficiency (line 5). If deemed beneficial, it executes the local training and MA sequentially for those energy-sufficient satellites $\mathcal{S}_c$, deriving the local model updates $\mathcal{P}$ (lines 6-8). The empty second argument in ExecuteTraining in line 8 indicates the absence of data transfer. Otherwise, it allocates compute/communication resources for data transfer based on the computed link utility iteratively (lines 9-15). It then executes the local computation and data transfer for different satellites (line 16). Finally, all local model updates and training statistics (implicit in $\mathcal{P}$) are sent to the ground to update the global model (line 17).

Figure 6 shows the OrbitalBrain's effectiveness. First, the inter-sat MA reduces the model staleness of participating satellites at each round, making 90% of model staleness smaller than 8 (from 18, Figure 6(a)). Second, assuming no inter-sat MA, OrbitalBrain's data transfer strategy assigns satellites for data transfer and local computing dynamically (Figure 6(b)). More satellites transfer data at the beginning when the data distribution is more unbalanced.

4 Implementation

System Implementation. We implement OrbitalBrain[2] in Python (3,556 lines) for orbital simulation and distributed ML training in space. Table 8 in the appendix summa-

[2]It will be open-sourced as an ML training simulator in space.

Algorithm 1: OrbitalBrain's ML training framework

Input: Time Steps $\mathcal{T}$, Sat constellation $\mathcal{S}$, GS set $\mathcal{G}$
Output: Decision-making strategy for $\forall\ s \in \mathcal{S}$ at each time window $t \in \mathcal{T}$

Loop $t \in \mathcal{T}$ with a scheduling window T_t:
  /* Step-1: Retrieve and update sat profiling */
  Loss,H,E,M,$\mathbf{l}$,G = UpdateProfiling($\mathcal{S}$,$\mathcal{G}$,t)
  $\text{Util}_{\text{comp}}(\mathcal{S})$ = UpdCompUtil(Loss,H,E,M) // Eq (1)
  $\mathcal{S}_c, \mathcal{S}_{cc}$ = SatStatusDetection($\mathcal{S}$,E)
  /* Step-2: Inter-sat MA feasibility check */
  T_{MA},$\boldsymbol{\epsilon}$ = ShortestPathTree($\mathcal{S}_c$,G_{ISL})
  If $T_{MA} < T_t$ and Eq (2) holds:
    UpdateCompEfficiency($\mathcal{S}_c, \epsilon$) // Fig 4(c)
    /* Step-3: Execute local training and MA */
    $\mathcal{P}$ = ExecuteTraining($\mathcal{S}_c, \emptyset$)
  Else :
    $\text{Util}_{\text{link}}$ = InitLinkUtil($\mathcal{S}_c, E, M, \mathbf{l}$) // Eq (3) to (5)
    $\mathcal{S}_t = \emptyset$ // Init Link Util and DT sats
    While $\exists\ \text{Util}_{\text{link}}(s, s') > 0$:
      $s, s' = \text{argmax}_{s,s'}\ \text{Util}_{\text{link}}(s, s')$
      If $s \neq s'$: $\mathcal{S}_t$.add(s)
      $\text{Util}_{\text{link}}$ = UpdLinkUtil($\mathcal{S}_c, E, M, \mathbf{l}, s, s'$)
    /* Step-4: Execute DT and local training */
    $\mathcal{P}$ = ExecuteTraining($\mathcal{S}_c \setminus \mathcal{S}_t$, $\mathcal{S}_t$)
  /* Step-5: Comm model and data with the ground */
  CommWithGround($\mathcal{P}$)

rizes the APIs exposed by OrbitalBrain to support the orbital emulation and ML training in space, respectively. Given various state machine setups and orbital mechanisms in Table 6, OrbitalBrain's orbital simulation based on `CosmicBeats` (Microsoft Research, 2023) outputs the energy traces, link topology, and data streamed for each satellite as the number of data samples manipulated for different purposes, such as the images collected, transferred, and computed at each scheduling window. Our ML simulator built on top of a federated learning framework (`FLUTE` (Dimitriadis et al., 2022)) takes these traces to update the satellite profiles/link utility for inter-sat communication and simulate distributed ML training in space under physical constraints. We use OpenMPI (Gabriel et al., 2004) as the backbone for multi-worker distributed ML training.

5 Evaluation

We evaluate OrbitalBrain's performance with two satellite constellations on three space tasks. The key results are:

- **Overall Performance.** OrbitalBrain outperforms SOTA baselines for various space tasks. It achieves 1.52×-12.4× speedup in time-to-accuracy with 1.9%-49.5% final model accuracy improvement.
- **Ablation Study.** OrbitalBrain optimizes the non-i.i.d data distribution and model staleness effectively with data transfer and modal aggregation under physical constraints.
- **Robustness & Sensitivity.** OrbitalBrain consistently performs well in various scenarios, such as varying cloudy images, and imagery data size/resolution, and participating satellites. We also show the effectiveness of incorporating GS and satellites for ML in space.
- **Approaching the Ideal Case.** OrbitalBrain narrows the performance gap in comparison to its ideal counterparts.

Table 2. Statistics of the evaluated dataset for OrbitalBrain.

Dataset	Type	Class	Sample	Metric
`fMoW`	RGB	62	363,563	Acc
`So2Sat`	MSI	17	400,673	Acc

5.1 Methodology

Constellations, Tasks, Dataset. For the orbital emulation, we utilize 24-hour TLE traces of `Planet` (CelesTrack, Retrieved in 2022a) (with 207 satellites) and `Spire` (CelesTrack, Retrieved in 2022b) (with 117 satellites), along with state machine settings outlined in Table 6. We employ the locations of `Planet`'s 12 global ground stations. We assess OrbitalBrain on two space tasks and provide their statistics in Table 2. Appendix F offers a detailed description of these tasks.

Baselines. We implement and compare five baselines with OrbitalBrain under satellites' physical constraints, including the centralized training (i.e., `BentPipe` (BP) (Denby & Lucia, 2020)) and four FL approaches (i.e., `AsyncFL` (AFL) (Wang et al., 2019), `SyncFL` (SFL) (Bonawitz et al., 2019), `FedBuff` (FedB) (Nguyen et al., 2021), and `FedSpace` (FedS) (So et al., 2022)). In `SyncFL` (Bonawitz et al., 2019), the GS waits for the local gradients from a certain number of non-straggler satellites (50%) before updating and distributing its global model. In contrast, GS in `AsyncFL` updates the global model at its best efforts whenever local gradients are available from satellites. `FedBuff` balances the `SyncFL` and `AsyncFL` by buffering the local gradients from satellites and updating the global model only when its size reaches a threshold B (default 32). The SOTA ML framework in space, `FedSpace`, dynamically schedules model aggregation based on the deterministic connectivity and its performance estimation from model staleness.

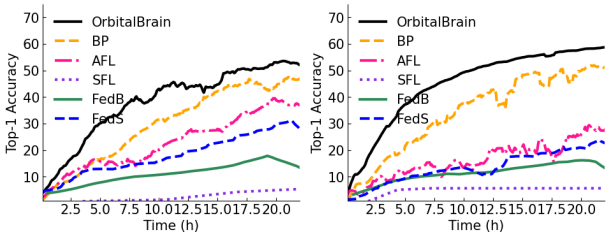
Metrics and Parameters. Our primary performance metric is *time-to-accuracy* (Lai et al., 2021; Li et al., 2022), which comprises the *final test accuracy* of the global ML model and its training wall clock time. Specifically, we calculate the final model accuracy by averaging the test accuracy over the last five scheduling windows. We measure OrbitalBrain's wall clock time required to achieve the final test accuracy of the baselines, enabling us to determine its *speedups* over them (refer to Table 4). We set the orbital sampling period T_t (scheduling window) to 5 minutes. Appendix F presents the remaining parameter settings.

5.2 End-to-End Performance

Table 3 and 4 summarize OrbitalBrain's improvements in final test accuracy and speedups with less wall clock time.

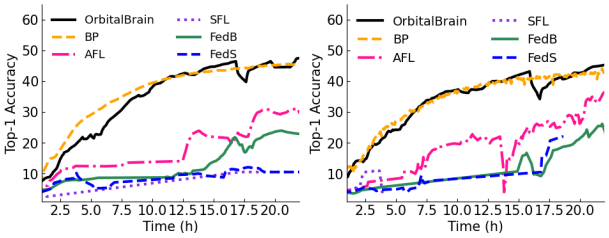
Table 3. OrbitalBrain improves the time-to-accuracy performance with a higher final test accuracy over all five baselines, under various space tasks, ML models, and satellite constellations.

Space Tasks	Dataset+Model	Const.	BP	SFL	AFL	FedB	FedS	OrbitalBrain
Land Function Recognition	fMoW + DenseNet	Planet	47.3%	3.3%	37.4%	17.8%	31.2%	**52.8%**
		Spire	50.2%	5.6%	24.2%	14.7%	18.0%	**59.2%**
Climate Zone Recognition	So2Sat + ResNet	Planet	46.0%	10.5%	17.8%	14.8%	10.5%	**47.9%**
		Spire	43.0%	9.4%	35.1%	20.6%	19.2%	**47.1%**



(a) under Planet constellation (b) under Spire constellation

Figure 7. Time-to-accuracy on fMoW.



(a) under Planet constellation (b) under Spire constellation

Figure 8. Time-to-accuracy on So2Sat.

OrbitalBrain improves the final test accuracy with more i.i.d data. For the most difficult task, fMoW, OrbitalBrain achieves much higher final test accuracy of 52.8% and 59.2% over existing baselines, with the improvement of 5.5%-49.5% on Planet and Spire. Figure 7 shows the time-to-accuracy curve for fMoW. fMoW's high non-i.i.d data renders a fluctuating trend for most baselines except the steady but slow SyncFL and BentPipe. In contrast, OrbitalBrain allocates some satellites for inter-satellite data transfer instead of the local computation to make data distribution more balanced for each satellite, resulting in a more steady training process with higher final test accuracy.

Figure 8 shows that OrbitalBrain achieves similar steady training trends with much higher test accuracy than the distributed ML baselines on the So2Sat dataset, under Planet and Spire constellations. Compared to fMoW, So2Sat only has 17 labels for the climate zone recognition, which favors BentPipe to train a good model even with much fewer data samples collected in the ground stations. Therefore, OrbitalBrain only improves BentPipe slightly, with a higher accuracy of 1.9% and 4.1% for climate zone recognition. We attribute this to our inter-satellite model aggregation to reduce model staleness. With only a few satellites connecting to the ground per window, satellites with a stale global model degrade aggregation quality when downloading their local models to the ground. OrbitalBrain sacrifices certain compute resources and time for inter-satellite communication to improve the quality of global model aggregation with fewer stale model updates.

OrbitalBrain speeds up the global model training by enabling frequent model aggregation across satellites. Table 4 presents the time it takes for OrbitalBrain's to reach each baseline's final test accuracy after a 24-hr ML training

Table 4. Time for OrbitalBrain to achieve the same final accuracy as baselines with the speedup in parenthesizes

Dataset	Const.	BP	AFL	FedS
fMoW	Planet	1.55×	3.25×	4.33×
	Spire	1.52×	8.73×	9.58×
So2Sat	Planet	2.22×	8.65×	12.42×
	Spire	1.61×	3.21×	9.24×

Table 5. OrbitalBrain's ablation study indicates the test accuracy decrement and training slowdown on fMoW (Christie et al., 2018).

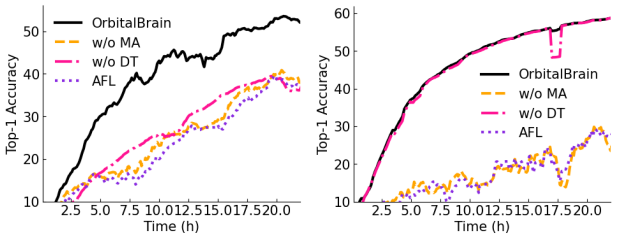
Constell.	w/o Model Aggreg.		w/o Data Transfer	
	Acc	Slow↓	Acc	Slow↓
Planet	14.7%↓	2.5×	11.3%↓	2.2×
Spire	36.1%↓	8.5×	0.3%↓	1.0

for fMoW and So2Sat.

We make two observations. First, OrbitalBrain is robust speedup across various ML models, image types, and constellations. For these two ML tasks, OrbitalBrain achieves higher speedup for fMoW and So2Sat because these two datasets were captured from the Earth with greater diversity, which leads to more difficult non-i.i.d. data issues for the baselines. Second, OrbitalBrain only needs several rounds of training warm-up and its efficiency increases over time. OrbitalBrain fully exploits the computation resources of all satellites with sufficient energy even if they cannot communicate to the ground. Such an edge computing framework makes it start faster than the centralized BentPipe (Denby & Lucia, 2020), especially for the diverse fMoW dataset with much more data labels.

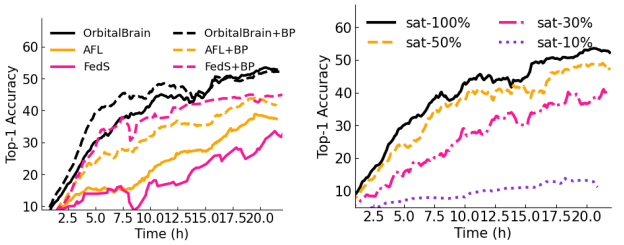
5.3 Component-wise Analysis

To understand the effectiveness of OrbitalBrain's key components, we evaluate its two breakdown versions.



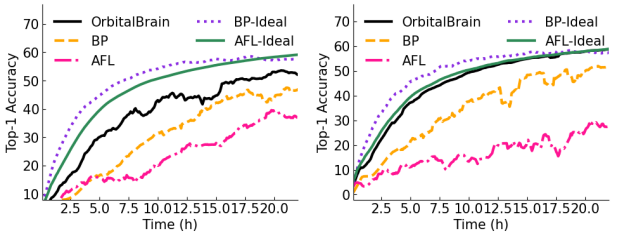
(a) fMoW +Planet (b) fMoW +Spire

Figure 9. Breakdown of OrbitalBrain's time-to-accuracy perf. under different datasets and constellations.



(a) Ground stations (b) Participating satellites

Figure 10. Performance of (a) collaborating with ground stations, and (b) varying participating satellites



(a) fMoW +Planet (b) fMoW +Spire

Figure 11. OrbitalBrain vs. ideal BentPipe/AsyncFL.

OrbitalBrain ensures the most up-to-date global model through inter-satellite model aggregation. After removing inter-satellite MA, Table 5 shows a decrease of 14.7% and 36.1% in final test accuracy for fMoW. The time taken by its breakdown counterparts to reach the same test accuracy is 2.5× to 8.5× longer. Figure9 compares the time-to-accuracy performance of OrbitalBrain without inter-satellite MA (orange) and AsyncFL (purple). Although OrbitalBrain without inter-satellite MA still slightly outperforms AsyncFL, it falls short of the complete OrbitalBrain (black). A unique case occurs with fMoW and the Spire constellation, where OrbitalBrain without MA overlaps with AsyncFL in the training trend. Due to the widely geo-distributed Spire constellation, the number and bandwidth of inter-satellite links decrease significantly, resulting in fewer images transferred across satellites and making their performance similar.

OrbitalBrain optimizes each satellite's data distribution by selectively substituting local computation with data transfer. OrbitalBrain without DT (data transfer) experiences a noticeable performance reduction. Table 5 shows that the degradation can reach up to 11.3% in final accuracy and 2.2× in time-to-accuracy for OrbitalBrain without DT on the Planet constellation.

An interesting observation is the Spire (CelesTrack, Retrieved in 2022b) constellation, in which OrbitalBrain w/o MA overlaps with the AsyncFL for the training trend. Given the widely geo-distributed Spire constellation, each satellite can collect more balanced data samples. As a result, they can collaboratively train a good global model by relying on the inter-satellite model aggregation only.

5.4 Robustness and Sensitivity

We use four distinct scenarios to evaluate OrbitalBrain's robustness and sensitivity. Appendix G provides a comprehensive description. The main takeaways are:

- **Collaboration between satellites and ground stations.** Figure 10(a) shows that OrbitalBrain, AsyncFL, and FedSpace all benefit from cooperating with ground stations for ML model training. This suggests a potential future direction for enhancing in-space ML training.
- **Participation rate.** As Figure 10(b) shows, OrbitalBrain attains robust training performance with moderate participation rates (50%) but struggles with a low participation rate (30% and 10%). This occurs because a low participation rate leads to increased data skewness and reduces the overall training contributions to the global model.
- **Cloudy images.** We evaluate OrbitalBrain and AsyncFL with different cloud cover thresholds. We find that OrbitalBrain achieves a much more consistent training curve, while AsyncFL fluctuates over time. This can be attributed to the data-sharing strategy that mitigates the local deviation of cloudy images.
- **Image size and resolution.** We evaluate OrbitalBrain, BentPipe, and AsyncFL with various image sizes. We discover that while larger images considerably hinder the training of BentPipe and AsyncFL, OrbitalBrain demonstrates a lesser impact from larger images. This is because OrbitalBrain adaptively determines the utility of local training and data transfer.

5.5 Comparison with the Ideal Cases

In this section, we propose two ideal baseline scenarios to showcase the effectiveness of OrbitalBrain: (1) BentPipe-ideal assumes i.i.d data for each satellite and downloads all gathered data to the ground without any link constraints. It consumes the same total energy as all satellites while training the global model with 10,000 data samples[3]. (2) AsyncFL-ideal also assigns i.i.d data to each satellite using random data assignment. Each satellite con-

[3]On average, an entire satellite constellation processes 105,019 data samples per window in space.

sistently communicates with the ground for global model distribution and aggregation.

We draw three conclusions from Figure 11: (1) Both `BentPipe` and `AsyncFL` experience significant difficulties due to their non-i.i.d data and intermittent connections with the ground for model aggregation, resulting in a substantial performance gap compared to their ideal scenarios. (2) A centralized training method with sufficient data, such as `BentPipe`-ideal, achieves the best performance (purple). However, it performs poorly when limited to non-i.i.d data samples under link constraints (orange). (3) By facilitating inter-satellite model aggregation and data transfer, OrbitalBrain optimizes data distribution and model staleness for each satellite, bringing it closer to the `AsyncFL`-ideal.

6 Related Work

Model Aggregation in Federated Learning. There is a large body of work studying different aspects of FL (McMahan et al., 2017; Kairouz et al., 2021). FL model aggregation can be broadly classified into Synchronous FL (`SyncFL`) (Bonawitz et al., 2019) and Asynchronous FL (`AsyncFL`) (Nguyen et al., 2021). `SyncFL` ensures the convergence of the global model (McMahan et al., 2017; Li et al.) by waiting for slower clients (Kairouz et al., 2021; Zhang et al., 2021), but this approach inevitably results in increased latency. In contrast, `AsyncFL` (Wang et al., 2019; Xie et al., 2019; Xu et al., 2021) avoids stragglers to improve time efficiency but must consider convergence analysis (Chen et al., 2020c;b) and model staleness for biased aggregation (Sprague et al., 2018; Chai et al., 2020). Our solution also takes into account the trade-offs between stragglers and model staleness. However, our problem formulation and solution are more comprehensive, as they also consider data sharing, local training, and adherence to interdependent physical constraints.

Federated Learning in Space. FL has been investigated for use in space-based training (Chen et al., 2022; Razmi et al., 2022; So et al., 2022), where each satellite only sends its local model updates to the ground, avoiding the need to download high-resolution data. However, these studies either adopt simplistic orbital assumptions or overlook certain constraints. For instance, `FedSpace` (So et al., 2022) only considers satellite-to-ground station connectivity for communication, neglecting energy, storage, or data streaming constraints. In contrast, we develop a first-of-its-kind ML training simulator using real satellite traces while adhering to the physical constraints of satellites (Tesat-Spaceco, Retrieved in 2022; Planet, 2022a;b), and we conduct extensive comparisons with existing solutions in our evaluation. Our approach also differs from traditional FL, as we do not require data isolation for privacy concerns among satellites (So et al., 2021; Nguyen et al., 2021; Lu et al., 2019). Consequently, OrbitalBrain thoroughly leverages inter-satellite data transfer and model aggregation for efficient ML training in space.

Resource-Constrained Federated Learning. A growing body of research in FL focuses on its application in resource-constrained environments, such as the Internet of Things (IoT), robots, and drones (Wang et al., 2019; Imteaj et al., 2022). Existing work typically targets a specific limited resource (e.g., computation, memory, and energy) and proposes solutions to improve the efficiency of that particular resource. For example, pruned or quantized models have been utilized to reduce computation and communication overheads (Jiang et al., 2019; Horváth et al., 2021; Li et al., 2021; Alam et al., 2022). Our work is largely complementary to these approaches, as we examine the *trade-offs* of various operational decisions to optimize the use of limited resources in a previously unexplored context. By combining our findings with these existing solutions, we can achieve even better performance in our problem setting.

Orbital Edge Computing for Satellites. The concept of orbital edge computing (Denby & Lucia, 2020; Denby et al., 2023) involves equipping satellites with computing devices to derive insights in space, which has been implemented in real satellite missions (Giuffrida et al., 2022). There are several simulators designed for this domain, and `CosmicBeats` (Microsoft Research, 2023) is publicly available and validated with actual nanosatellite deployments. Existing orbital edge computing research mostly focuses on running imagery filtering or ML inference on satellites, while our work targets more complex ML training tasks to keep these edge models up-to-date. Our simulator incorporates `CosmicBeats`'s components to provide a comprehensive research platform for ML training in space that respects real-world constraints.

7 Conclusion

Training ML models in space offers a promising alternative to conventional ground-based training, especially in the era of rapidly growing LEO nanosatellites. In this work, we develop a simulator for distributed ML training in space, demonstrating that the physical constraints of satellites pose significant challenges. We introduce OrbitalBrain, an efficient solution that intelligently balances trade-offs among data transfer, model aggregation, and local training. OrbitalBrain estimates the *utility* of various operations using predictable physical constraints and local training statistics, employing a two-stage decision process to allocate satellite resources. Extensive evaluation across two space ML tasks and two satellite constellations reveals that OrbitalBrain achieves a $1.52\times$-$12.4\times$ speed-up in time-to-accuracy and a 1.9%-49.5% final model accuracy improvement over existing centralized and federated learning baselines. We hope our findings, simulator, and datasets encourage further research to address this emerging problem.

REFERENCES

Spire global nanosatellite constellation. https://www.eoportal.org/satellite-missions/spire-global, 2017.

High-resolution imagery with Planet satellite tasking. https://www.planet.com/products/hi-res-monitoring/, 2020.

Planetscope - Dove Satellite Constellation (3m). https://www.satimagingcorp.com/satellite-sensors/other-satellite-sensors/dove-3m/, 2021.

Agency, E. S. Sentinel-2 msi user guide. In *https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi*, Retrieved in 2022.

Alain, G., Lamb, A., Sankar, C., Courville, A., and Bengio, Y. Variance Reduction in SGD by Distributed Importance Sampling. *arXiv:1511.06481 [cs, stat]*, 2016.

Alam, S., Liu, L., Yan, M., and Zhang, M. FedRolex: Model-heterogeneous federated learning with rolling sub-model extraction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Aragon, B., Houborg, R., Tu, K., Fisher, J. B., and McCabe, M. Cubesats enable high spatiotemporal retrievals of crop-water use for precision agriculture. *Remote Sensing*, 2018.

Barbulescu, L., Whitley, L. D., and Howe, A. E. Leap before you look: An effective strategy in an oversubscribed scheduling problem. In *Proceedings of AAAI*, 2004.

Barlow, H. B. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.

Barmpoutis, P., Papaioannou, P., Dimitropoulos, K., and Grammalidis, N. A review on early forest fire detection systems using optical remote sensing. *Sensors*, 2020.

Bhardwaj, R., Xia, Z., Ananthanarayanan, G., Jiang, J., Shu, Y., Karianakis, N., Hsieh, K., Bahl, P., and Stoica, I. Ekya: Continuous learning of video analytics models on edge compute servers. In *Proceedings of USENIX NSDI*, 2022.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., McMahan, B., et al. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, 2019.

Burke, M., Driscoll, A., Lobell, D. B., and Ermon, S. Using satellite imagery to understand and promote sustainable development. *Science*, 371(6535), 2021.

Cangialosi, F., Agarwal, N., Arun, V., Narayana, S., Sarwate, A., and Netravali, R. Privid: Practical,{Privacy-Preserving} video analytics queries. In *Proceedings of USENIX NSDI*, 2022.

CelesTrack. Planet. In *http://celestrak.org/NORAD/elements/table.php?GROUP=planet&FORMAT=tle*, Retrieved in 2022a.

CelesTrack. Spire. In *http://celestrak.org/NORAD/elements/table.php?GROUP=spire&FORMAT=tle*, Retrieved in 2022b.

Chai, Z., Chen, Y., Zhao, L., Cheng, Y., and Rangwala, H. Fedat: A communication-efficient federated learning method with asynchronous tiers under non-iid data. *ArXivorg*, 2020.

Chen, H., Xiao, M., and Pang, Z. Satellite-based computing networks with federated learning. *IEEE Wireless Communications*, 2022.

Chen, K., Avouac, J.-P., Aati, S., Milliner, C., Zheng, F., and Shi, C. Cascading and pulse-like ruptures during the 2019 ridgecrest earthquakes in the eastern california shear zone. *Nature communications*, 2020a.

Chen, T., Jin, X., Sun, Y., and Yin, W. Vafl: a method of vertical asynchronous federated learning. *arXiv preprint arXiv:2007.06081*, 2020b.

Chen, Y., Ning, Y., Slawski, M., and Rangwala, H. Asynchronous online federated learning for edge devices with non-iid data. In *Proceedings of IEEE International Conference on Big Data (Big Data)*, 2020c.

Cho, H., Mathur, A., and Kawsar, F. Flame: Federated learning across multi-device environments. *arXiv preprint arXiv:2202.08922*, 2022.

Christie, G., Fendley, N., Wilson, J., and Mukherjee, R. Functional map of the world. In *Proceedings of IEEE CVPR*, 2018.

Denby, B. and Lucia, B. Orbital edge computing: Nanosatellite constellations as a new class of computer system. In *Proceedings of ACM ASPLOS*, 2020.

Denby, B., Chintalapudi, K., Chandra, R., Lucia, B., and Noghabi, S. A. Kodan: Addressing the computational bottleneck in space. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2023.

Devaraj, K. Planet's high speed downlink network: Employing agile aerospace to download 20 TB daily imagery from the Dove constellation. *LEOCONN*, 2021.

Devaraj, K., Ligon, M., Blossom, E., Breu, J., Klofas, B., Colton, K., and Kingsbury, R. Planet High Speed Radio: Crossing Gbps from a 3U Cubesat. In *Small Satellite Conference*, 2019.

Dimitriadis, D., Garcia, M. H., Diaz, D. M., Manoel, A., and Sim, R. Flute: A scalable, extensible framework for high-performance federated learning simulations. *arXiv preprint arXiv:2203.13789*, 2022.

Escher, A. https://techcrunch.com/2018/09/14/inside-planet-labs-new-satellite-manufacturing-site/, 2018.

Franch-Pardo, I., Napoletano, B. M., Rosete-Verges, F., and Billa, L. Spatial analysis and gis in the study of covid-19. a review. *Science of the total environment*, 2020.

Franklin, D. Nvidia jetson tx2 delivers twice the intelligence to the edge. In *https://developer.nvidia.com/blog/jetson-tx2-delivers-twice-intelligence-edge/*, Retrieved in 2022.

Gabriel, E., Fagg, G. E., Bosilca, G., Angskun, T., Dongarra, J. J., Squyres, J. M., Sahay, V., Kambadur, P., Barrett, B., Lumsdaine, A., et al. Open mpi: Goals, concept, and design of a next generation mpi implementation. In *European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting*. Springer, 2004.

Giuffrida, G., Fanucci, L., Meoni, G., Batic, M., Buckley, L., Dunne, A., van Dijk, C., Esposito, M., Hefele, J., Vercruyssen, N., Furano, G., Pastena, M., and Aschbacher, J. The Φ-sat-1 mission: The first on-board deep neural network demonstrator for satellite earth observation. *IEEE Trans. Geosci. Remote. Sens.*, 60:1–14, 2022.

Handley, M. Delay is not an option: Low latency routing in space. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, 2018.

Horváth, S., Laskaridis, S., Almeida, M., Leontiadis, I., Venieris, S. I., and Lane, N. D. FjORD: Fair and accurate federated learning under heterogeneous targets with ordered dropout. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 12876–12889, 2021.

Hsieh, K., Phanishayee, A., Mutlu, O., and Gibbons, P. B. The Non-IID data quagmire of decentralized machine learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

Imteaj, A., Thakker, U., Wang, S., Li, J., and Amini, M. H. A survey on federated learning for resource-constrained iot devices. *IEEE Internet Things J.*, 9(1):1–24, 2022.

in-the-sky.org. Flock 3p-49. In *https://in-the-sky.org/spacecraft.php?id=42001*, 2022.

Jewett, R. Next starlink satellites will have inter-satellite links, shotwell says. In *https://www.satellitetoday.com/broadband/2021/08/26/next-starlink-satellites-will-have-inter-satellite-links-shotwell-says/*, Retrieved in 2022.

Jiang, Y., Wang, S., Ko, B. J., Lee, W., and Tassiulas, L. Model pruning enables efficient federated learning on edge devices. *CoRR*, abs/1909.12326, 2019.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 2021.

Katharopoulos, A. and Fleuret, F. Not all samples are created equal: Deep learning with importance sampling. In *Proceedings of ICML*. PMLR, 2018.

Lai, F., Zhu, X., Madhyastha, H. V., and Chowdhury, M. Oort: Efficient Federated Learning via Guided Participant Selection. In *Proceedings of USENIX OSDI*, 2021.

Li, A., Sun, J., Zeng, X., Zhang, M., Li, H., and Chen, Y. FedMask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking. In *SenSys '21: The 19th ACM Conference on Embedded Networked Sensor Systems*, pp. 42–55. ACM, 2021.

Li, C., Zeng, X., Zhang, M., and Cao, Z. Pyramidfl: Fine-grained data and system heterogeneity-aware client selection for efficient federated learning. In *Proceedings of ACM MobiCom*, 2022.

Li, Q., Wen, Z., and He, B. Federated learning systems: Vision, hype and reality for data privacy and protection. *arXiv preprint arXiv:1907.09693*.

Lu, Y., Huang, X., Dai, Y., Maharjan, S., and Zhang, Y. Differentially private asynchronous federated learning for mobile edge computing in urban informatics. *IEEE Transactions on Industrial Informatics*, 2019.

Magazine, M. J. and Chern, M.-S. A note on approximation schemes for multidimensional knapsack problems. *Mathematics of Operations Research*, 1984.

Maimaitijiang, M., Sagan, V., Sidike, P., Daloye, A. M., Erkbol, H., and Fritschi, F. B. Crop monitoring using satellite/uav data fusion and machine learning. *Remote. Sens.*, 12(9):1357, 2020.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the International Conference on Artificial Intelligence and Statistics, PMLR*, 2017.

Microsoft Research. CosmicBeats-Simulator: A space simulation platform that caters to individuals with diverse research interests, including networking, AI, computing, and more. Unlike traditional simulators tied to specific research applications, our design allows for seamless integration of various space-related research verticals. https://github.com/microsoft/CosmicBeats-Simulator, 2023.

Moision, B. and Hamkins, J. Deep-space optical communications downlink budget: modulation and coding. *IPN Progress Report*, 2003.

Mundhenk, T. N., Konjevod, G., Sakla, W. A., and Boakye, K. A large contextual dataset for classification, detection and counting of cars with deep learning. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III*, volume 9907, pp. 785–800, 2016.

NASA-IMPACT. Etci 2021 competition on flood detection. In *https://nasa-impact.github.io/etci2021/*, 2021.

Nguyen, J., Malik, K., Zhan, H., Yousefpour, A., Rabbat, M., Malek, M., and Huba, D. Federated learning with buffered asynchronous aggregation. *arXiv preprint arXiv:2106.06639*, 2021.

NVIDIA. Technical specifications for jetson-tx2. In *https://developer.nvidia.com/embedded/jetson-tx2*, Retrieved in 2022.

Padwick, C., Deskevich, M., Pacifici, F., and Smallwood, S. Worldview-2 pan-sharpening. In *Proceedings of the American Society for Photogrammetry and Remote Sensing (ASPRS)*, 2010.

Planet. Planet imagery and archive. In *https://www.planet.com/products/planet-imagery/*, 2022a.

Planet. Planet imagery product specifications. In *https://assets.planet.com/docs/combined-imagery-product-spec-final-may-2019.pdf*, 2022b.

Razmi, N., Matthiesen, B., Dekorsy, A., and Popovski, P. Ground-assisted federated learning in leo satellite constellations. *IEEE Wireless Communications Letters*, 2022.

Safyan, M. Planet's dove satellite constellation. *Handbook of Small Satellites: Technology, Design, Manufacture, Applications, Economics and Regulation*, 2020.

Shukla, S., Macharia, D., Husak, G. J., Landsfeld, M., Nakalembe, C. L., Blakeley, S. L., Adams, E. C., and Way-Henthorne, J. Enhancing access and usage of earth observations in environmental decision-making in eastern and southern africa through capacity building. *Frontiers in Sustainable Food Systems*, 2021.

Singh, V., Chakraborty, T., Jog, S., Chabra, O., Vasisht, D., and Chandra, R. Spectrumize: Spectrum-efficient satellite networks for the internet of things. In *Proceedings of USENIX NSDI*, 2024.

So, J., Ali, R. E., Güler, B., and Avestimehr, A. S. Secure Aggregation for Buffered Asynchronous Federated Learning. *arXiv:2110.02177 [cs, math, stat]*, 2021.

So, J., Hsieh, K., Arzani, B., Noghabi, S., Avestimehr, S., and Chandra, R. Fedspace: An efficient federated learning framework at satellites and ground stations. *arXiv preprint arXiv:2202.01267*, 2022.

Sprague, M. R., Jalalirad, A., Scavuzzo, M., Capota, C., Neun, M., Do, L., and Kopp, M. Asynchronous federated learning for geospatial applications. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018.

Sumbul, G., De Wall, A., Kreuziger, T., Marcelino, F., Costa, H., Benevides, P., Caetano, M., Demir, B., and Markl, V. Bigearthnet-mm: A large-scale, multimodal, multilabel benchmark archive for remote sensing image classification and retrieval [software and data sets]. *IEEE Geoscience and Remote Sensing Magazine*, 2021.

Tesat-Spaceco. Cubelct - cubesat laser communication transmitter. In *https://satsearch.co/products/tesat-cubelct*, Retrieved in 2022.

Vasisht, D., Shenoy, J., and Chandra, R. L2d2: Low latency distributed downlink for leo satellites. In *Proceedings of ACM SIGCOMM*, 2021.

Wang, P., Reinelt, G., Gao, P., and Tan, Y. A model, a heuristic and a decision support system to solve the scheduling problem of an earth observing satellite constellation. *Computers & Industrial Engineering*, 2011.

Wang, S., Tuor, T., Salonidis, T., Leung, K. K., Makaya, C., He, T., and Chan, K. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, 2019.

Werner, D. Planet pelicans to offer high resolution and revisit rates. In *https://spacenews.com/planet-pelican-details/*, Retrieved in 2022.

Wertz, J. R., Larson, W. J., Kirkpatrick, D., and Klungle, D. *Space mission analysis and design*. Springer, 1999.

WorldVu Satellites Limited. WorldVu Satellites Limited (d/b/a OneWeb) proposes to expand its previously authorized 720-satellite LEO constellation in Ku and Ka-band to 1,980 satellites. *FCC Fixed Satellite Service Filing SAT-MOD-20180319-0002. Federal Communication Commission*, 2018.

Xie, C., Koyejo, S., and Gupta, I. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*, 2019.

Xu, C., Qu, Y., Xiang, Y., and Gao, L. Asynchronous federated learning on heterogeneous devices: A survey. *arXiv preprint arXiv:2109.04269*, 2021.

Zech, H., Biller, P., Heine, F., and Motzigemba, M. Optical intersatellite links for navigation constellations. In *Proceedings of the International Conference on Space Optics (ICSO)*. SPIE, 2019.

Zhang, T., Gao, L., He, C., Zhang, M., Krishnamachari, B., and Avestimehr, S. Federated learning for internet of things: Applications, challenges, and opportunities, 2021.

Zhao, P. and Zhang, T. Stochastic optimization with importance sampling for regularized loss minimization. In *Proceedings of ICML*. PMLR, 2015.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. Federated learning with non-iid data. *CoRR*, abs/1806.00582, 2018.

Zhu, X. J. Semi-supervised learning literature survey. 2005.

Zhu, X. X., Hu, J., Qiu, C., Shi, Y., Kang, J., Mou, L., Bagheri, H., Häberle, M., Hua, Y., Huang, R., et al. So2sat lcz42: A benchmark dataset for global local climate zones classification. *arXiv preprint arXiv:1912.12171*, 2019.