AdaOrb: Adapting In-Orbit Analytics Models for Location-aware Earth Observation Tasks

Zhouyu Li, Pinxiang Wang, Xiaochun Liang, Xuanhao Luo, Yuchen Liu, Xiaojian Wang, Huayue Gu, Ruozhou Yu

***Abstract*—The rapid growth in low-Earth-orbit satellites enables providing Earth observation applications to public users via a shared platform. However, the limited satellite-ground communication resources present a major challenge in downloading and fully utilizing satellite-captured Earth observation data on the ground. As a new edge computing paradigm, *orbital edge computing* allows satellites to host deep learning models with on-board computing resources for in-orbit data analysis, reducing downlink data volume and response time. However, the limited generalizability of in-orbit models and data distribution shifts across geographical locations severely impact the accuracy of in-orbit analytics. In this work, we design a framework, AdaOrb, which dynamically schedules online model retraining for location-specific Earth observation tasks. Scheduling decisions are made with a model predictive control-based algorithm that allocates limited satellite downlink capacity among onboard tasks to download model retraining data. By developing and using a hardware-in-the-loop orbital edge computing testbed, we show that our method achieves superior overall accuracy of in-orbit analytics tasks compared to alternative methods.**

***Keywords*—Orbital edge computing, Earth observation, low Earth orbit satellite, model adaptation, downlink scheduling**

I. Introduction

The development of low-Earth-orbit (LEO) satellite services and standard nanosatellite products [1] has made Earth observation more accessible to the public. Nowadays, users can submit their customized analytics tasks to existing Earth observation constellations through platforms like Planet [2]. By specifying an area of interest and analytic goal, satellite images are downloaded periodically and analyzed on the ground. While these satellites can capture high-resolution aerial images covering large areas, limited ground-satellite communication resources and brief ground-satellite contact periods prevent users from downloading and analyzing all the captured images.

The concept of Orbital Edge Computing (OEC) [3] was proposed to address the downlink capacity constraint through lightweight in-orbit computation. Under this backdrop, a flexible framework for multi-task OEC was introduced [4]. In multi-task OEC, analytics tasks are performed promptly on the satellite using deep-learning models for edge devices. This approach allows the satellite to serve more users and deliver results in a timely manner. However, challenges arise from the distribution shift of satellite images among different locations [5]. In-orbit models with limited generalizability often struggle to perform well in areas unseen in their training datasets [4]. Moreover, the diversity of satellite constellations, each equipped with different sensors and using different image harmonization techniques, causes data distribution shifts among satellites and prevents data reusing from other constellations. This leads to a shortage of task-specific, annotated datasets for each constellation and makes it hard to train a global model offline that performs well on various areas. The location-wise data distribution shift, coupled with the difficulties of training multi-area models, significantly impacts in-orbit analytic accuracy and makes in-orbit processed results unreliable. We further explore this problem in Section II-B.

As the Earth observation service continuously incorporates unpredictable areas of interest, the data distribution shifts keep happening. A promising solution is to develop location-specific models through retraining with online data from each area due to the compact and stateless nature of in-orbit models that enable fast instantiation and flexible updates. While similar techniques have been applied in edge video analytics [6], [7], those applications typically assume abundant communication resources between cameras and edge servers. In contrast, communication resources remain a major limitation for OEC.

Existing OEC frameworks either neglect data distribution shifts [8], or employ simple *a priori* defined preprocessing (such as context-aware image filters [4]) to improve OEC accuracy. Such approaches cannot adapt well to dynamically submitted Earth observation tasks with unpredictable areas of interest. To address this issue, we exploit the downlink channel to download a portion of images to retrain in-orbit analytics models specific to user-specified areas of interest. We devise a framework, AdaOrb, that adapts in-orbit analytics deep learning models for location-aware Earth observation tasks by scheduled model retraining. AdaOrb schedules model retraining by allocating downlink capacity to download task-specific retraining data during each ground-satellite connection. For each satellite, the allocation decisions jointly consider its intermittent bandwidth-constraint connections to the ground and sporadic visits to each task's area of interest, and leverage the model predictive control (MPC) method to maximize the overall analytics accuracy for all onboard tasks. To evaluate our proposed framework, we implement a hardware-in-the-loop OEC testbed with the computation unit used on existing LEO satellites [9]. Evaluation results show that our methods significantly improve the overall accuracy of all the onboard analytics tasks under a limited downlink channel. Our major contributions are summarized as follows:

- We propose AdaOrb, a model retraining scheduling framework that adapts in-orbit deep learning models for dynamically submitted Earth observation tasks.
- We formulate the downlink capacity allocation problem for downloading retraining data for in-orbit models, and design an MPC-based solution for the problem.
- We implement an OEC testbed with real satellite computing devices and show that AdaOrb improves overall in-orbit analytics accuracy over alternative methods.

The rest of the paper is organized as follows: Section II presents preliminary experiments that provide motivations and insights for solution design. Section III presents the detailed design of the AdaOrb framework. Section IV evaluates the

Li, Wang, Liang, Luo, Liu, Wang, Gu, Yu ({zli5, pwang25 xliang8, xluo26, yliu322, xwang244, hgu5, ryu5}@ncsu.edu) are all with North Carolina State University, Raleigh, NC 27606, USA. This research was supported in part by NSF grant 2045539, 2433966, and 2312138. The information reported here does not reflect the position or the policy of the funding agency.

effectiveness of AdaOrb. Section V introduces related works. Section VI concludes the paper.

II. Preliminary Experiments

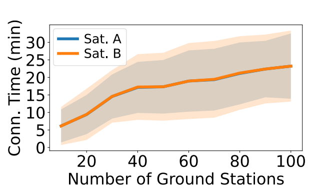
In this section, we inspect the constraint downlink channel and unique patterns in satellite traces. We also explore the distribution shifts of Earth observation data across different locations and the best practice for OEC model retraining.

A. Downlink Channel Inspection

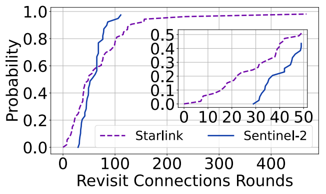
We inspect the downlink channel by simulating satellite traces with the Hypatia LEO simulator [10] using the orbit of Sentinel-2 Earth observation constellation [11]. For both satellites A and B in the Sentinel-2 constellation, we record the ground-satellite connection time for each orbital revolution around the Earth. In addition to the existing ground stations in Matera, Svalbard, and Maspalomas [12], we add more ground stations near highly populated cities [10] to check the average per-revolution connection time with different numbers of ground stations. We choose top-populated cities because that constellation operation and satellite data processing require domain-specific experts and reliable network connections, which are resources typically available near large urban centers. The results are presented in Fig. 1(a). Curves and shadow areas present the mean and standard deviations of the connection time in each revolution. We observe that the average connection time per revolution remains under 10 minutes when the number of ground station is less than 20. This brief and intermittent connection limits the amount of data to be downloaded from the satellite to the ground. Though the per-revolution connection time increases linearly as more ground stations being built at first, but converges to less than 25 minutes. The marginal benefits of building additional ground stations are decreasing. Moreover, we cannot increase the number of ground stations unrestrictedly due to significant costs and time overhead. Since building more ground stations is not a practical solution to mitigate the downlink capacity constraint. We need to efficiently use the downlink channel.

Proposition 1. *An efficient framework is in need to better leverage the downlink channel, given the limitations in existing and near-future ground-satellite communication resources.*

To improve downlink channel utilization, we ask a key question: *Do we need to download data for all locations from the last ground-satellite connection?* To answer this question, we simulate satellite traces and analyze the number of ground-satellite contacts between revisits to the same location. We simulate both the Sentinel-2 and the first shell of the Starlink constellations. For Sentinel-2, we use its three dedicated ground stations [12]. For Starlink, we use the ten most populated cities as ground station locations [10]. We record the timing of ground-satellite connections and satellite visits to the 61 locations from the SpaceNet7 dataset [13]. For each location, we count the number of ground-satellite connections between consecutive visits to that location, which we call *revisit connection rounds*. Fig. 1(b) shows the cumulative proportion of locations with revisit connection rounds above the x-axis value. Our analysis reveals that only in extremely rare cases does a satellite revisit the same location after just one ground-satellite connection, leading to Proposition 2.

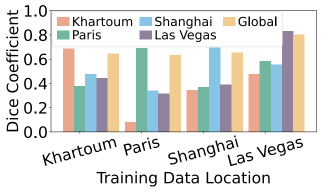


(a) Per-revolution connection time v.s. numbers of ground stations.

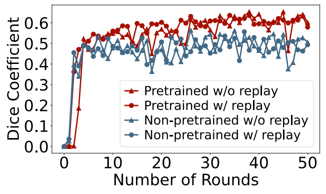


(b) Revisit connection rounds CDF

Fig. 1: (a). The per-revolution connection time of satellite A and B in the Sentinel-2 constellation for different numbers of ground stations. Lines and shadows are the mean and standard deviation of connection time. (b). CDF for revisit connection rounds in Starlink and Sentinel-2. We observe that most satellites will revisit the same location after several rounds of connections.



(a) Location data distribution shift.



(b) Model retraining accuracy curves.

Fig. 2: (a). Accuracy of applying analytics models trained on data from different locations to other locations for building detection and segmentation. Bars in the same color are accuracies of the same model for the location in the legend. Each cluster of bars represents models' accuracies when applied to the location indicated on the x-axis. We observe a significant accuracy drop when applying a model trained for one location to another location. (b). Accuracies v.s. model retraining rounds during model adaptation. We notice starting from pretrained models (red lines) leads to higher accuracy at convergence, while using the replay technique (circle marks) stabilizes the training process.

Proposition 2. *For any given area of interest, it is not necessary to download the data immediately after capturing.*

B. Data Distribution Shift and Model Retraining

In this experiment, we first inspect the data distribution shift in satellite data across different locations with a typical Earth observation application: building segmentation [14]. We use the SpaceNet2 dataset [15] with four different locations: Khartoum, Paris, Shanghai, and Las Vegas. Following existing work [16], we train a UNet [17] model for each location with data exclusively from that area. We then apply each of these models to all four locations in the dataset to evaluate their performance across different areas. Fig. 2(a) presents the DICE coefficient [18] of different models when applied to their training data location and other locations. The metric ranges from 0 to 1, with higher values indicating better performance. In Fig. 2(a), we observe a significant accuracy drop when models trained on one location are applied to other locations. From these results, we can notice that *in-orbit analytic models trained with data collected at one location cannot generalize well to data collected at other locations.* This finding substantiates the data distribution shifts across locations, making it impractical to directly apply a model trained on one location to another. While satellite orbits follow predictable paths, we cannot predict which locations will be specified in new Earth observation tasks. This makes it infeasible to train a comprehensive global model offline, as downloading data from all locations along the satellite's orbit would exceed downlink capacity constraints. To address these distribution shifts, researchers typically retrain the model for every location with online data [6], [7]. This raises an important question: *Should we build a global model using all available data, or should we train separate models for each location?* We train a global model with an equal amount of location-specific data evenly sampled from four locations in the dataset. Its performance is shown in Fig. 2(a). We notice that the global model accuracy is always lower than that of training a dedicated model for each location. Since the extreme case for training a global model is we only have one location, the model performance should be equal to the location-specific model; with the number of interested locations increasing, the global model performance will keep decreasing. Hence, we have the proposition:

Proposition 3. *Training separate models tailored to each area is a solution to cope with location-wise data distribution shifts.*

We demonstrate the effectiveness of model retraining with the accuracy curve of incrementally training an UNet model using data from Las Vegas. We retrain the model for 50 rounds with 20 images per round. Results in Fig. 2(b) reflect that:

Observation 1. *Model accuracy generally increases as more retraining data is used.*

We also explore how different training starting points affect performance. We use models pretrained on the SpaceNet1 dataset [15] as starting points. As shown in Fig. 2(b), starting with a pretrained model leads to higher accuracy at convergence. Additionally, Fig. 2(b) confirms the effectiveness of the *replay* technique from continuous learning research [19]. Training with accumulated data enables faster convergence and produces more stable models. So we have our proposition:

Proposition 4. *When retraining a model to a new location, start with pretrained weights and use accumulated data.*

III. AdaOrb Framework

In response to Proposition 1 and Proposition 3, we design AdaOrb, an intelligent OEC framework to efficiently exploit the downlink capacity to adapt deep learning models for location-specific analytics tasks via retraining.

A. Framework Design

We first present the design of the AdaOrb framework in Fig. 3. The framework includes three major components: the mission data center, the ground station, and the satellites. The mission data center manages satellites' onboard tasks, including task publication, data annotation, and model training. It is equipped with abundant computational resources and sufficient network bandwidth connecting all ground stations. Ground stations serve as interfaces for ground-satellite communication. Earth observation satellites are equipped with sensors to collect Earth observation data. Satellites also have some edge computational resources for onboard data analytics. However, these resources are too limited to handle intensive workloads like in-orbit model training. By close collaboration among the components, AdaOrb achieves the following functions for model adaptation.

Task management. The mission data center continuously receives Earth observation tasks from users, and periodically publishes these tasks to satellites. An Earth observation *task* is defined by an *area of interest* and an *analytics model*. The published tasks are first forwarded to all ground stations and sent to the satellite when a ground-satellite connection is established.

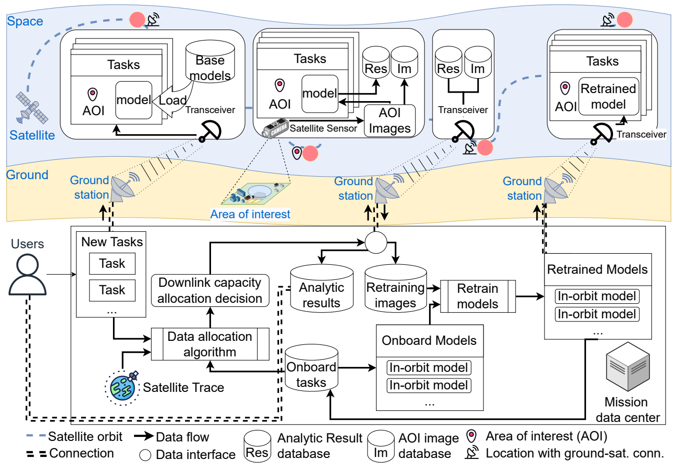


Fig. 3: Framework overview. The mission data center receives Earth observation tasks from users and publishes them to satellites via ground stations. It also makes downlink capacity allocation decisions and conducts model retraining. The satellite loads a pre-loaded base model for each task according to their type of analysis when receiving new tasks. When passing over a task's area of interest, the satellite collects Earth observation data with onboard sensors and performs in-orbit analysis with the in-orbit model. Analytic results and part of the collected data are downloaded in the following ground-satellite connection according to the downlink capacity allocation decision. Models retrained in the mission data center will be uploaded to the satellite using the next ground-satellite connection after downloading the retraining data.

For every published task, the mission data center maintains the latest copy of its in-orbit model in a local database for potential model retraining. Once a task is published, it is deemed as *onboard*. The dynamic and unpredictable onboard task set presents the challenge of making adaptive online decisions.

Downlink capacity allocation. The mission data center also monitors all satellite movements and tracks their traces. It forecasts the time and duration of future satellite-ground connections, as well as the number of images to be captured in areas of interest. In response to Proposition 2, AdaOrb uses a downlink capacity allocation algorithm to strategically allocate the downlink capacity of each connection to all the onboard tasks based on this forecast information. This algorithm is executed whenever there is an update to the onboard task set, such as when a new task is published or a task's model has been retrained. The algorithm decision is sent to all ground stations to instruct each satellite's data download during the ground-satellite connection. We formally formulate the downlink capacity allocation problem in Section III-B and solve it by our algorithm in Section III-C.

Location-specific in-orbit analytics. Each task consists of an area of interest and an in-orbit model. Following Proposition 3, we use a specific model for each task dedicated to its area of interest. When a task is onboarded to the satellite, its model is initiated by loading a copy of the base model. According to Proposition 4, the base model is pre-trained for the general type of analysis required by the task. Base models can be pre-uploaded on each satellite to minimize the time a user has to wait to start its task. However, the initiated model for each task is expected to be updated as new data becomes available. When the satellite passes over a task's area of interest, it captures images using its sensors and applies the onboard model for in-orbit analysis. The captured location-specific images are also stored for future downloading and retraining.

Model retraining and update. During a ground-satellite connection, besides downloading analytics results, the ground station downloads location-specific images from satellites instructed by the downlink capacity allocation decision to retrain each task's model. After the connection, the designated quantity of images for each task is downloaded and sent to the mission data center for annotation and subsequent retraining. The retrained model weights are uploaded to the satellite via the uplink during the next satellite-ground connection, potentially with a different ground station.

B. Downlink Capacity Allocation Problem

To efficiently leverage the opportunity in Proposition 2, we formulate an online decision problem as a Markov decision process whose parameters, control inputs, states, transition function, and cost function are specified below.

Tasks and connections. We define a satellite to have a set of onboard tasks $\mathbf{K} = \{k_i | i = 1, 2, \ldots, N_K\}$. Each time point refers to a round of ground-satellite connection, which is predictable and represented as a set $\mathbf{T} = \{t_j | j = 0, 1, 2, \ldots, N_T\}$.

Accuracies. Let the average accuracy of the model for task k_i on all the captured images at k_i's area of interest from time point t_j to t_{j+1} be acc_j^i. Based on Observation 1, for each task k_i at time point t_j, we represent the acc_j^i as a function of the number of downloaded images d_j^i, denoted as $\mathcal{F}_i$

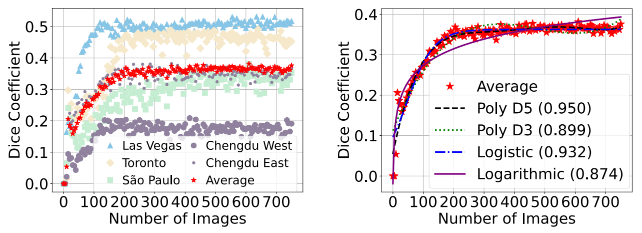
$$acc_j^i = \mathcal{F}_i\left(d_j^i\right), \quad \forall j \geq 1, k_i \in \mathbf{K}. \tag{1}$$

Since it takes time to annotate the downloaded images and retrain the model on the ground. The images downloaded at time t_j for task k_i will be used to retrain the model with current accuracy acc_j^i. The updated model will be uploaded to the satellite at time point t_{j+1}, assuming we always have enough computational power and communication resources on the ground. The updated model will be applied to images captured at the area of interest of task k_i from time point t_{j+1} to timepoint t_{j+2} with accuracy acc_{j+1}^i.

Time-varying parameters. The number of images captured for task k_i from time point t_j to t_{j+1} is a predictable external time-varying parameter ν_j^i. We let $\nu_0^i = 0, \forall k_i \in \mathbf{K}$ as there is no data available at the initial time point t_0. All the ν_j^i's from t_j to t_{j+1} forms a set $\boldsymbol{\nu}_j = \left\{\nu_j^i | i = 1, 2, \ldots, N_K\right\}$. Until the *beginning* of time point t_j, the accumulated number of captured images for task k_i is v_j^i, where $v_j^i = \sum_{j'=0}^{j-1} \nu_{j'}^i$. On connection at time point t_j, the number of downloadable images, referred to as *downlink capacity*, is also a predictable time-varying parameter denoted as b_j.

Control inputs. Let the number of images to be downloaded for the task k_i in connection t_j be δ_j^i. At time point t_j, δ_j^i from all the tasks in $\mathbf{K}$ form a set of *control inputs* $\boldsymbol{\Delta}_j = \{\delta_j^i | i = 1, 2, \ldots, N_K\}$. The amount of downloaded images for task k_i until (but not include) t_j is denoted as d_j^i, where $d_j^i = \sum_{j'=0}^{j-1} \delta_{j'}^i$, $j \geq 1$. Values of δ_j^i's are subject to the *downlink capacity constraint* $\sum_{i=1}^{N_K} \delta_j^i \leq b_j, \forall j \geq 1$ and *available online image constraint* $d_j^i + \delta_j^i \leq v_j^i, \forall j \geq 1, \forall k_i \in \mathbf{K}$.

States. We use set $\mathbf{X} = \{\boldsymbol{\chi}_j | j = 0, 1, 2, \ldots, N_T\}$ to represent the states in the Markov decision process. Each state $\boldsymbol{\chi}_j$ is defined by model accuracies, accumulated number of downloaded images, and accumulated number of captured images



(a) Data-accuracy sample points when incrementally training UNet models on randomly selected five locations, and their average values.

(b) Fit with different functions. In the parenthesis is the coefficient of determination of each curve.

Fig. 4: Profiling the accuracy function.

for all the onboard tasks at the beginning of time point t_j.

$$\boldsymbol{\chi}_j = \left\{ \left(acc_j^i, d_j^i, v_j^i\right) \middle| i = 1, 2, \ldots, N_K \right\}, \forall t_j \in \mathbf{T}. \tag{2}$$

Transition function. After taking action $\boldsymbol{\Delta}_j$, the state transition function is defined as

$$\boldsymbol{\chi}_{j+1} = \mathcal{P}\left(\boldsymbol{\chi}_j, \boldsymbol{\Delta}_j, \boldsymbol{\nu}_j\right), \quad \forall \left(N_T - 1\right) \geq j \geq 0. \tag{3}$$

This function comprises three parts: the accuracy change for each task $acc_{j+1}^i = \mathcal{F}_i\left(d_j^i\right)$, the aggregated number of downlinked images for each task $d_{j+1}^i = d_j^i + \delta_j^i$, and the aggregated number of captured images of each task $v_{j+1}^i = v_j^i + \nu_j^i$, for all the tasks $k_i \in \mathbf{K}$ and timepoints $t_j, j \geq 1$.

Cost function. We aim to maximize the sum of all the analytic accuracy acc_j^i for all the v_j^i online predictions over all the onboard tasks $k_i \in \mathbf{K}$ and time points $t_j \in \mathbf{T}$. Specifically, given a prediction horizon N_H, our cost function is defined as

$$J\left(\boldsymbol{\Delta}\right) = \sum_{i=1}^{N_K} \sum_{j'=j}^{j+N_H} \nu_{j'}^i (1 - acc_{j'}^i). \tag{4}$$

where $\boldsymbol{\Delta}$ is a sequence of the control inputs $\boldsymbol{\Delta}_j$ in the prediction horizon $\boldsymbol{\Delta} = [\boldsymbol{\Delta}_j, \boldsymbol{\Delta}_{j+1}, \ldots, \boldsymbol{\Delta}_{j+N_H-1}]$, and the *prediction horizon* N_H represents the number of future time points the algorithm can anticipate and plan for.

C. Downlink Capacity Allocation Algorithm

Due to limited online data availability, training a reinforcement learning (RL) model for online decision-making is impractical. Such models require extensive online data to explore and refine their decision-making policies through actual retraining [20]. This motivates us to design an MPC-based algorithm to solve the downlink capacity allocation problem. Unlike RL-based methods, MPC does not require extensive data to bootstrap. However, MPC requires a closed-form state transition function. In the state transition function Eq. (3), while the number of downloaded and captured images is predictable from satellite traces, the accuracy function remains unknown. Fortunately, we can obtain an approximated function through profiling.

Profiling the accuracy function. To profile the unknown accuracy function, we obtain several sample curves of the relationship between the number of retraining images (data) and model accuracy (accuracy) by incrementally retraining models on several location-specific datasets offline. We use known functions to fit the average curve for all profiled samples to get the approximation of the accuracy function in

Eq. (1). To illustrate the accuracy function profiling process, we incrementally train pruned UNet models [17], [21] on data from randomly selected five locations in the SpaceNet7 dataset [13] for building segmentation. Profiled data-accuracy curves are presented in Fig 4 (a). We observe that curves for all locations exhibit a nearly submodular shape. Candidates for fitting such curves include polynomial, logarithmic, and logistic functions. Additionally, we notice that curves for the same type of task at different locations, though converging to different accuracies, share the same pattern of convergence. To this end, it is reasonable to profile and construct a uniform function, denoted as $\tilde{\mathcal{F}}$, that represents the typical data-accuracy curves for each type of task and apply the function to new, unseen locations. In Fig. 4(b), we show the average curve fitting results of different functions among 3-degree polynomial, 5-degree polynomial, logarithmic, and logistic functions. Each function's coefficient of determination is also presented in the legend of Fig. 4(b). We find that the 5-degree polynomial function has the highest coefficient of determination and best fits the average curve.

Solving the problem with MPC. Using the profiled accuracy function, we can solve the online decision problem with MPC by substituting the approximated function

$$\tilde{acc}_j^i = \tilde{\mathcal{F}}\left(d_j^i\right), \forall j \geq 1, k_i \in \mathbf{K} \tag{5}$$

for the original accuracy function in Eq. (1), and use this approximated accuracy $\tilde{acc}_j^i$ to replace the accuracy terms in Eq. (2) and Eq. (4). Since $\tilde{\mathcal{F}}$ is non-linear, we adopt a non-linear optimization technique such as the Gauss-Newton algorithm [22] to solve the MPC problem iteratively by minimizing the cost function Eq. (4). Upon convergence, the control input is a downlink capacity allocation decision that maximizes the overall accuracy of all the onboard tasks.

IV. Evaluation

To evaluate the effectiveness and efficiency of AdaOrb, we implement a hardware-in-the-loop OEC emulator and use real constellation traces and Earth observation images for AdaOrb's performance assessment. We also explore the influence of key parameters on AdaOrb's performance.

A. *Evaluation Setup*

Dataset. We use the training set of the SpaceNet7 dataset [13], excluding the 5 locations used for accuracy function profiling in Section III-C. The evaluation set includes 54 locations, each representing an area of interest for a building segmentation task. We divide each image into 64 pixel $\times$ 64 pixel tiled images for analysis—a common technique to reduce computational overhead [4]. The tile size aligns with public datasets such as EuroSat [23], [24] and PASTIS [25]. In the following paragraphs, unless otherwise specified, all references to images pertain to these 64 pixel $\times$ 64 pixel tiled ones.

Constellations. We obtain one month's satellite movement traces from two active constellations, Sentinel-2 and Starlink, via the Hypatia LEO simulator [10], with the same setup in Section II-A. We record the times when the satellite passes over each location in our dataset and the duration of each connection with ground stations. In our discrete event emulation, we consider each ground-satellite connection as a time step tick, as it allows us to download data for retraining and update the retrained models during these connections and aligns with the time point in Section III. According to existing nanosatellite specifications [1], the typical downlink data rate is 125 megabits per second. Accounting for traffic generated by normal operations (such as delivering images and analytical results to users), we set aside 10% of the available bandwidth for downloading the retraining data. By default, we assume each satellite generates 100 images when visiting an area of interest. Given our small model size, we assume the uplink is always sufficient for model and decision updates.

Tasks. For our evaluation, we designate each location in our dataset as an area of interest for a task. Task initiation follows the order in which the satellite first passes over each area of interest, with tasks published to the satellite on a daily basis through ground stations. The number of published tasks per publication follows a Poisson distribution with a mean of 10.

Model and retraining. We use the UNet model [17] for building segmentation tasks. As per Proposition 4, we pretrain the model on the SpaceNet 1 dataset [15]. To enhance efficiency for in-orbit inferences, we prune half of the parameters and fine-tune the model before deployment [21]. The pruned UNet contains approximately 63,000 trainable parameters and occupies only 2.4MB of memory space. Upon each connection, we randomly download previously captured images for each location. We split these images into training and validation sets at an 8 : 2 ratio. Each model is trained until convergence or for at most 20 epochs. Following Proposition 4, we cumulatively add images to the dataset for retraining. We impose each task to download at most 300 images for retraining, as the model accuracy converges at this amount according to Fig. 4(b).

Metrics. DICE coefficient [18] is the metric for building segmentation accuracy. For each task, the sum of DICE coefficients of all in-orbit predictions is its inference *utility*.

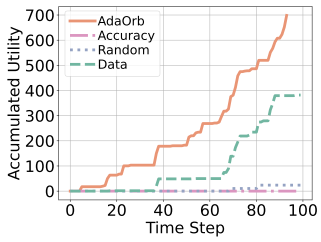
Compared algorithms. In our evaluation, we compare AdaOrb's MPC-based downlink capacity allocation algorithm with the following algorithms:

- **Random-allocation (Random)**: Randomly splits the downlink capacity among all onboard tasks for retraining.
- **Accuracy-oriented (Accuracy)**: Allocates downlink capacity to tasks weighted by one minus their accuracies.
- **Data-oriented (Data)**: Allocates downlink capacity to tasks weighted by their total numbers of incoming images within the prediction horizon.

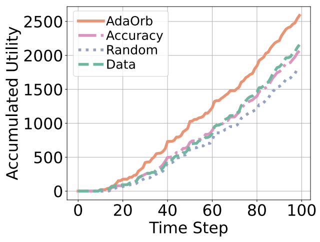
All algorithms work under the framework described in Section III. By default, we set the prediction horizon for AdaOrb's MPC-based algorithm and the data-oriented algorithm to 5 and 20 time steps for Sentinel-2 and Starlink, respectively. We also examine the impact of varying this prediction horizon.

B. *Evalutation results*

Utility over time. We first compare the cumulative utility of online inferences over 100 time steps. The evaluation results are presented in Fig. 5. Our proposed OEC scheduling scheme significantly outperforms the comparison algorithms after the initial 10-time-step bootstrap stage. This advantage stems from AdaOrb's ability to allocate more downlink capacity to tasks with suboptimal models while prioritizing models with higher per-image accuracy improvements. We observe that algorithms capable of anticipating and planning for incoming data volume (AdaOrb and data-oriented) generally perform better than data-agnostic algorithms (random-allocation and accuracy-oriented), highlighting the importance of leveraging

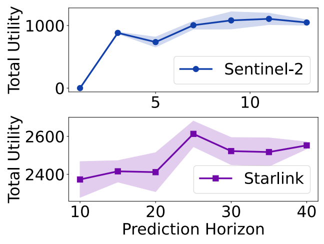


(a) Sentinel.

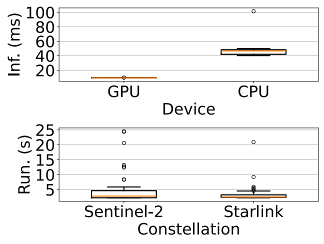


(b) Starlink.

Fig. 5: Accumulated utility over time. We observe that AdaOrb consistently outperforms the compared algorithms, with its advantage in accumulated utility increasing as the time step increases.



(a) Influence of prediction horizon.



(b) Model and algorithm efficiency.

Fig. 6: (a). Prediction horizon of Sentinel-2 and Starlink constellations. (Upper). Sentinel-2 constellation with task publication interval of 5 time steps. (Lower). Starlink constellation with task publication interval of 20 time steps. (b). Efficiency evaluation results. (Upper). Inference time for in-orbit models with different devices. (Lower). Algorithm running time for two constellations.

this unique feature of LEO satellites for data allocation. However, without a proper accuracy function approximation and optimal downlink capacity assignment, the performance of the data-oriented algorithm remains suboptimal compared to AdaOrb. This demonstrates the importance of leveraging predictable satellite movement patterns with carefully designed algorithms. The algorithms perform differently in the two constellations since Starlink's lower altitude provides longer contact times and more downloadable data in each connection, leading to higher inference utility for all algorithms compared to Sentinel-2. This difference highlights the importance of efficient scheduling algorithms for bandwidth-constrained high-orbit constellations and demonstrates the advantages of scheduling across varying downlink channel conditions.

Selection of prediction horizon. We examine how prediction horizon, a key parameter of the MPC algorithm, affects performance. Results are shown in Fig. 6(a). The onboard task set updates every 5 time steps for Sentinel-2 and 20 time steps for Starlink during daily task publication. We observe that utility peaks when the prediction horizon is slightly longer than these update intervals, after which it declines. This suggests that while a prediction horizon marginally longer than the update interval improves decision-making, excessively long horizons incorporate outdated information and reduce achievable utility.

In-orbit model efficiency. We examine the feasibility of using the location-specific model for in-orbit analytics by recording the per-image inference time of the in-orbit model on the Nvidia Jetson Orin Nano, an edge intelligent device that serves as the computation unit in real nanosatellites [26]. The benchmark results for using and not using the onboard GPU are presented in the upper part of Fig. 6(b). We observe that the median inference time for an image, with and without using an embedded GPU are 11.9 ms and 58.5 ms, respectively. A typical image capture interval of 30 seconds allows the processing of 500 to $2,500$ image tiles of 64 pixels $\times$ 64 pixels. With a 10-meter-per-pixel resolution, this efficiency enables analysis of 204.6 to $1,036.6$ square kilometers per 30 seconds on a median level, which aligns with the average size of areas of interest in existing Earth observation constellations [11], and validates the feasibility of using the selected model for location-specific in-orbit analytics.

Algorithm efficiency. We benchmarked the running time of the MPC-based downlink capacity scheduling algorithm for the two evaluated constellations on a desktop with 64GB memory and a 16-Core Intel i9-9900K CPU. Results are presented in the lower part of Fig. 6(b). Except for a few outliers, the highest running times for both constellations are around 5 seconds, with median times below 3 seconds. Given that scheduling decisions are made between each consecutive satellite-ground connections—an interval significantly longer than a few seconds—the MPC-based algorithm will incur negligible time and computational overhead.

V. Related Works

Downlink channel constraints and data distribution shifts have been two long-standing problems affecting Earth observation applications [5], [27], [28] To address communication constraints, Denby et al. first introduced the concept of OEC [27] for performing data analytics in orbit. Recent works have incorporated image compression and residual encoding techniques to further conserve downlink bandwidth after filtering [29]. Instead of downloading filtered and compressed images, recent studies propose obtaining analytics results directly in orbit [8]. To handle data distribution shifts, Ekya [6] schedules resources for retraining and inference using a theft algorithm to improve performance on end devices when real-time data diverges from the training distribution. Adainf [7] enables inference using partially trained models, allowing more flexible online training scheduling. However, these approaches do not adequately address the challenges in communication-constrained orbital environments. [4] observes performance degradation across different contexts and dynamically switches between context-specific models to improve inference accuracy. However, our work shows that model performance degrades even within the same context when applied to different locations.

VI. Conclusion

In this paper, we propose AdaOrb, a framework that adapts in-orbit analytics models through intelligently scheduled model retraining with an MPC-based algorithm. Extensive preliminary studies validate the downlink capacity constraint, data distribution shifts across locations, and the effectiveness of adapting in-orbit model to location-specific data by retraining. Satellites' revisit gaps among different locations provide opportunities for strategic model retraining scheduling. We design the AdaOrb with novel functions for model adaptation and formulate a downlink capacity allocation problem to maximize overall accuracy for all users' tasks. To solve this problem, we design an MPC-based algorithm with profiled model accuracy function. We develop a hardware-in-the-loop OEC emulator to evaluate our framework's effectiveness and feasibility. Results show that our downlink capacity allocation algorithm outperforms other compared algorithms and AdaOrb can run efficiently on off-the-shelf desktops.

References

[1] "6U XL Platform." URL: https://www.endurosat.com/products/6u-xl-platform/ Accessed date: 04/10/2024.

[2] P. L. PBC, "Planet application program interface: In space for life on earth," 2018/. URL: https://api.planet.com Accessed date: 03/10/2024.

[3] B. Denby and B. Lucia, "Orbital Edge Computing: Machine Inference in Space," *IEEE Comput. Arch. Lett.*, vol. 18, no. 1, pp. 59–62, 2019.

[4] B. Denby, K. Chintalapudi, R. Chandra, B. Lucia, and S. Noghabi, "Kodan: Addressing the Computational Bottleneck in Space," in *ACM ASPLOS*, 2023, pp. 392–403.

[5] E. Rolf, K. Klemmer, C. Robinson, and H. Kerner, "Mission Critical – Satellite Data is a Distinct Modality in Machine Learning," 2024, arXiv:2402.01444.

[6] R. Bhardwaj, Z. Xia, G. Ananthanarayanan, J. Jiang, Y. Shu, N. Karianakis, K. Hsieh, P. Bahl, and I. Stoica, "Ekya: Continuous learning of video analytics models on edge compute servers," in *USENIX NSDI*, 2022, pp. 119–135.

[7] S. S. Shubha and H. Shen, "AdaInf: Data Drift Adaptive Scheduling for Accurate and SLO-guaranteed Multiple-Model Inference Serving at Edge Servers," in *ACM SIGCOMM*, 2023, pp. 473–485.

[8] Q. Zhang, X. Yuan, R. Xing, Y. Zhang, Z. Zheng, X. Ma, M. Xu, S. Dustdar, and S. Wang, "Resource-efficient In-orbit Detection of Earth Objects," 2024, arXiv:2402.01675.

[9] T. Mahendrakar, R. T. White, M. Tiwari, and M. Wilde, "Unknown noncooperative spacecraft characterization with lightweight convolutional neural networks," *Journal of Aerospace Information Systems*, vol. 21, no. 5, pp. 455–460, 2024.

[10] S. Kassing, D. Bhattacherjee, A. B. Águas, J. E. Saethre, and A. Singla, "Exploring the "internet from space" with hypatia," in *Acm IMC*, 2020.

[11] S. Delwart, "ESA Standard Document," no. 1, accessed date: 04/10/2024. URL: https://sentinel.esa.int/documents/247904/685211/Sentinel-2_User_Handbook

[12] "Sentinel-2: So much data, so little time | Thales Group," 2016. URL: https://www.thalesgroup.com/en/critical-information-systems-and-cybersecurity/news/sentinel-2-so-much-data-so-little-time Accessed date: 01/09/2024.

[13] A. Van Etten, D. Hogan, J. Martinez-Manso, J. Shermeyer, N. Weir, and R. Lewis, "The Multi-Temporal Urban Development SpaceNet Dataset," 2021, arXiv:2102.04420.

[14] C. Robinson, A. Ortiz, H. Park, N. L. Gracia, J. K. Kaw, T. Sederholm, R. Dodhia, and J. M. L. Ferres, "Fast building segmentation from satellite imagery and few local labels," in *IEEE CVPR Workshop*, 2022, pp. 1462–1470.

[15] A. Van Etten, D. Lindenbaum, and T. M. Bacastow, "SpaceNet: A Remote Sensing Dataset and Challenge Series," 2019, arXiv:1807.01232.

[16] W. Liu, Z. Lai, Q. Wu, H. Li, Q. Zhang, Z. Li, Y. Li, and J. Liu, "In-Orbit Processing or Not? Sunlight-Aware Task Scheduling for Energy-Efficient Space Edge Computing Networks," in *IEEE INFOCOM*, 2024, pp. 881–890.

[17] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *MICCAI*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, vol. 9351, pp. 234–241.

[18] L. R. Dice, "Measures of the Amount of Ecologic Association Between Species," *Ecology*, vol. 26, no. 3, pp. 297–302, 1945.

[19] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: Incremental Classifier and Representation Learning," in *IEEE CVPR*, 2017, pp. 5533–5542.

[20] H. Ma, X. Luo, and D. Xu, "Intelligent queue management of open vSwitch in multi-tenant data center," *Future Generation Computer Systems*, vol. 144, pp. 50–62, 2023.

[21] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning Convolutional Neural Networks for Resource Efficient Inference," 2017, arXiv:1611.06440.

[22] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation," in *Nonlinear Model Predictive Control*, M. Morari, M. Thoma, L. Magni, D. M. Raimondo, and F. Allgöwer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, vol. 384, pp. 391–417.

[23] 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 JSTARS*, 2019.

[24] ——, "Introducing EuroSAT: A novel dataset and deep learning benchmark for land use and land cover classification," in *IEEE IGARSS*, 2018, pp. 204–207.

[25] V. Sainte Fare Garnot and L. Landrieu, "Panoptic segmentation of satellite image time series with convolutional temporal attention networks," *IEEE ICCV*, 2021.

[26] "Lockheed Martin and USC to Launch Jetson-Based Nanosatellite for Scientific Research Into Orbit," 2020. URL: https://developer.nvidia.com/blog/lockheed-martin-usc-jetson-nanosatellite/ Accessed date: 12/29/2024.

[27] B. Denby and B. Lucia, "Orbital Edge Computing: Nanosatellite Constellations as a New Class of Computer System," in *ASPLOS*, 2020, pp. 939–954.

[28] V. Singh, A. Prabhakara, D. Zhang, O. Yağan, and S. Kumar, "A community-driven approach to democratize access to satellite ground stations," in *ACM MobiCom*, 2021, pp. 1–14.

[29] K. Du, Y. Cheng, P. Olsen, S. Noghabi, R. Chandra, and J. Jiang, "Earth+: On-board satellite imagery compression leveraging historical earth observations," 2024, arXiv:2403.11434.