Collaborative Inference in DNN-based Satellite Systems with Dynamic Task Streams

Jinglong Guan[1], Qiyang Zhang[1,2], Ilir Murturi[2], Praveen Kumar Donta[2], Schahram Dustdar[2], Shangguang Wang[1]

[1]*State Key Laboratory of Networking and Switching Technology,*
Beijing University of Posts and Telecommunicatons, Beijing, China
[2]*Distributed System Group, TU Wien*, Vienna, Austria
{mr_guanjl;qyzhang;sgwang}@bupt.edu.cn;{imurturi;pdonta;dustdar}@dsg.tuwien.ac.at

Abstract—As a driving force in the advancement of intelligent in-orbit applications, DNN models have been gradually integrated into satellites, producing daily latency-constraint and computation-intensive tasks. However, the substantial computation capability of DNN models, coupled with the instability of the satellite-ground link, pose significant challenges, hindering timely completion of tasks. It becomes necessary to adapt to task stream changes when dealing with tasks requiring latency guarantees, such as dynamic observation tasks on the satellites. To this end, we consider a system model for a collaborative inference system with latency constraints, leveraging the multi-exit and model partition technology. To address this, we propose an algorithm, which is tailored to effectively address the trade-off between task completion and maintaining satisfactory task accuracy by dynamically choosing early-exit and partition points. Simulation evaluations show that our proposed algorithm significantly outperforms baseline algorithms across the task stream with strict latency constraints.

Index Terms—Satellite inference, Task offloading, Multi-exit DNNs, Model partition

I. INTRODUCTION

With the continuous technological advancements and increasing drive for space exploration, recent years have witnessed the proliferation of Low Earth Orbit (LEO) satellites such as Telesat and SpaceX [1]. Today, around 44% of in-orbit LEO satellites are dedicated to Earth Observation (EO) purposes [2]. Many machine learning algorithms, especially Deep Neural Networks (DNN), are employed for inference raw image observations such as forest fires and geomorphic changes [3].

The prevailing approach suggests transmitting all observations to the ground for image analysis, utilizing a traditional method referred to as the "bent pipe" architecture [4]. However, due to the typically unreliable and intermittent satellite-ground environment, a significant amount of raw observations remains untransmitted in a limited time. An alternative solution involves executing DNN models directly on the satellite where the data is generated [5]. However, the majority of LEO satellites are equipped with limited computing power and small physical size, which pose challenges in handling the high computational demands associated with complex DNN models. Much attention has been given to offloading in-orbit computing tasks to the ground as a promising solution. Typically, existing efforts focus on offloading tasks to ground stations for processing, provided that the satellites are within the coverage of specific ground stations. While straightforward and practical, this solution can suffer from a computational bottleneck due to the limited constraints of each satellite.

To achieve efficient inference offloading, many state-of-the-art systems like Tracking and Data Relay Satellite (TDRS) and European Data Relay System (EDRS) involve satellite relays in High Earth Orbit (HEO) satellites for task offloading [6]. It is reported that even a free-space laser link achieves up to 1.8 Gbps data rate [7]. When faced with extensive and diverse computational demands, it is necessary to collaborate the efforts of multiple satellites. The task owner distributes inference tasks across multiple satellites, with each satellite equipped with high-performance commercial, off-the-shelf (COTS) computational hardware.

Despite these efforts, it remains challenging for the inference task streams within the limited time constraints in potentially extreme factors [6] such as poor network conditions and excessive task arrivals. To meet the requirement of current EO tasks, which can range from demanding precision to partial precision at times, multiple satellites are required to collectively perform observations, with different satellites handling specific computational tasks. In such a computing environment characterized by diversity and dynamics, multi-exit mechanisms are crucial to address these issues. Multi-exit mechanisms are introduced as a solution to reduce a significant computational load by selectively activating an early-exit (EE) point [8] within a multi-layer network, even if it leads to inference degradation. Partitioning multi-exit models for multiple task stream applications remains to be fully considered. However, the simultaneous management of EE points and partition points adds complexity to the inference system, especially when dealing with evolving task profiles.

This paper proposes an efficient framework for inference offloading involving multiple satellites by partitioning multi-exit DNN models, which consist of one HEO satellite and multiple LEO satellites. We divide the inference task into two parts, with the task owner executing the initial portion of the DNNs and offloading the remaining part to other task executors. The key challenge lies in determining the optimal EE and partitioning points for each task, based on specific task demands and the current system status. We need to consider the trade-off between task completion and maintaining satisfactory task accuracy. When prioritizing high

inference performance for each task, some tasks may fail to meet their deadlines due to prolonged waiting times. In turn, if we solely focus on increasing system throughput, overall inference performance may not meet expectations. Therefore, this paper needs to formulate this issue and design efficient algorithms to address it.

The main contributions of our paper include the following aspects.

- We formulate an optimization problem by considering inference task and computing models to trade-off between task completion and maintaining satisfactory task accuracy under the time limitations.
- We introduce a task gain-aware method based on task completion and task accuracy, aiming to derive an efficient solution to the formulated problem.
- Simulation evaluations show that our proposed algorithm significantly outperforms the baseline algorithms across the task stream.

II. BACKGROUND AND RELATED WORK

"Bent pipe" architecture. Most LEO satellites have recently employed a "bent pipe" architecture, where satellites function as data-transmitting relays rather than data-processing capabilities. For example, Giuliari et al. [9] introduced an inter-networking approach that involves ground stations serving as gateways for satellite-ground communications, aimed at enhancing satellite data transmission efficiency. Cheng et al. [10] proposed an integrated satellite-ground computing architecture that utilizes satellites to enable access to cloud resources on the ground. However, data transmission between satellites and the ground is significantly hindered by an unstable communication link and intermittent availability. Consequently, this results in a substantially lower transmission rate compared to the rate at which data is generated onboard, leading to a noteworthy increase in the overall latency of all tasks.

In-orbit computing. This paradigm has attracted much attention in the satellite computing domain. Denby et al. [4] proposed the orbital edge computing framework that performs data filtering on the satellites, which aims to enhance downlinking efficiency in saturated satellites while considering limited computational capacity. Denby et al. [11] recently presented a novel computing architecture that maximizes the utility of saturated satellite-ground downlinking while alleviating computational bottlenecks. Therefore, for existing computing schemes, the entire inference process is executed on satellites. The limited computing resources and low energy acquisition capabilities of this approach also place a heavy burden on satellites. To address these issues, we investigate a model partitioning strategy for fine-grained computational resource optimization based on AI-oriented tasks.

Model partitioning for offloading inference. Some efforts suggest that model partitioning can be used to offload inference tasks from task owners [12], [13]. Neurosurgeon [14] proposed layer-wise partition strategies to adaptively offload model computation. JointDNN [15] was a collaboration framework that employs both model training and inference to achieve resource optimization. Nonetheless, these works ensure model partitions to guarantee inference accuracy under various constraints.

Multi-exit models for boosting task inference. There has been a growing interest in partitioning multi-exit models to enhance the performance of DNNs. For example, EDeepSave [16] aimed to improve the inference performance by the EE mechanism to avoid interruption when handing over to the mobile network. Edgent [17] proposed a model of DNN partitioning to maximize inference accuracy while satisfying latency constraints. Different from the above, we consider multiple inference task streams to balance task latency and accuracy.

III. SYSTEM MODEL AND PROBLEM FORMULATION

Fig. 1 illustrates the time-slotted task stream system, which includes one HEO satellite and multiple LEO satellites. For each satellite, we deployed the pre-trained models for subsequent adaptive inference tasks in advance. HEO satellite images typically feature high spatial resolution and extensive coverage, while LEO satellite images offer even greater resolution and more frequent observations. The selection of the appropriate satellite depends on the specific task requirements. We describe the workflow of an inference system involving two main stages as follows: 1) A task request includes current task requirements when it arrives. Specifically, for HEO imaging tasks, the HEO satellite captures the image first and stores it in the task queue; for LEO imaging tasks, the LEO satellite transmits the task information (e.g., task type, data size, time constraint) to the HEO satellite. In the case of imaging tasks, the HEO satellite determines offloading solutions and transmits the offloading decision back to the LEO satellites; 2) When LEO satellites perform imaging tasks, they receive offloading information, process the front part, and send the remaining part to HEO satellites. HEO satellites handle these tasks on the First-Come-First-Served (FCFS) principle. During HEO imaging tasks, HEO satellites also process the front part and then send the rest to LEO satellites; LEO satellites receive the processing result once HEO satellites complete the task. For ease of discussion, we do not consider the time it takes for the final results from the HEO satellite for LEO imaging tasks [18].

A. Inference Task Model

Let the task be denoted by $T_n = (S_n, D_n, L_n, t_n^A)$, where T_n indicates the n-th task information in the task stream, S_n indicates task type, D_n indicates task size, L_n indicates the maximum allowable task time constraint, and t_n^A indicates the arrival time. Given that there is a proportional relationship between L_n and D_n, we have $L_n = k \times D_n, k \in N$. Meanwhile, the number of EE points is M and the layer number in the i-th EE point is M_i. Further, the EE and the partition points of the n-th task are E_n and P_n respectively, with the constraints $0 \leq E_n \leq M, 0 \leq P_n \leq M_{E_n}$.

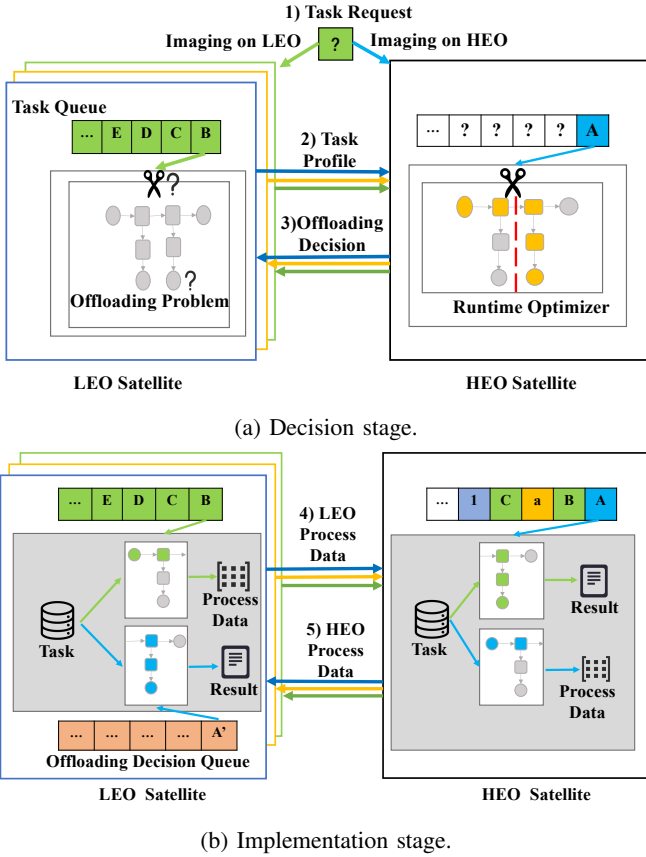(a) Decision stage.

(b) Implementation stage.

Fig. 1: The workflow of collaborative task stream offloading system.

Inspired by [15], processing time and transmission data size during handover are application-specific based on profiling. The prediction model can provide accurate predictions in practical applications, characterized by high reliability and minimal implementation costs. Consequently, we use a linear regression-based prediction model, which can accurately estimate the inference time and output data size of each layer in the multi-exit model, considering both LEO and HEO satellites. For example, we consider the case of LEO satellites. We utilize *timeline* to record the inference time and fit the inference time of each model layer with a linear function. For the input data size D_n, $f_{i,j}^d(D_n)$ represents the predicted inference time of the jth layer in the ith EE point, which is executed on the LEO satellite if $d = 0$ or the HEO satellite if $d = 1$. $f_{i,j}^d(D_n)$ represents the predicted output data size of the jth layer in the ith EE point of the model.

B. Computing Model

When selecting E_n and P_n, we divide the entire system into three stages: First, when the imaging task is executed on the LEO, we compute task T_n's first stage processing time as $t_n^{P_1} = \sum_{i=1}^{P_n} f_{E_n,i}^0(D_n)$, or $t_n^{P_1} = 0$; Second, when imaging task is executed on the LEO, we can compute task T_n's second stage processing time as $t_n^{P_2} = \sum_{i=P_n+1}^{M_{E_n}} f_{E_n,i}^1(D_n)$, or $t_n^{P_2} =$

$\sum_{i=1}^{P_n} f_{E_n,i}^1(D_n)$; Third, when imaging task is executed on the HEO, we can compute task T_n's third stage processing time as $t_n^{P_3} = 0$, or $t_n^{P_3} = \sum_{i=P_n+1}^{M_{E_n}} f_{E_n,i}^0(D_n)$.

We denote the transmission time of inter-satellite links as $L_{latency}^{ISL}$, which is denoted as follows:

$$L_{latency}^{ISL} = \frac{len(D)}{w(S_L, S_H)} + \frac{Q_{(S_L,S_H)}}{c} \qquad (1)$$

where S_L denotes the LEO satellite, S_H denotes the HEO satellite, D denotes the transmitted data size between LEO and HEO, $Q_{(S_L,S_H)}$ denotes the length of the transmission link between HEO and LEO, c denotes the speed of laser communication. We denote the transmission rate of data $w(S_L, S_H)$ as follows:

$$w(S_L, S_H) = B \log_2 \left(1 + \frac{P_r(S_L, S_H)}{k_B T_s B \gamma}\right) \qquad (2)$$

where B denotes the bandwidth, k_B denotes the *Boltzmann's constant*, T_s denotes the system noise temperature, γ denotes the Signal-to-Noise ratio, $P_r(S_L, S_H)$ denotes the received signal strength. The calculation formula is as follows:

$$P_r(S_L, S_H) = P_t G_t G_r \left(\frac{4\pi Q_{(S_L,S_H)} f}{c}\right)^{-2} \qquad (3)$$

where P_t denotes the transmit power, G_t denotes the receive gain, G_r denotes the transmit gain, f denotes the carrier frequency. So the transmission time between LEO and HEO satellites is

$$t_n^{tr} = \frac{f_{E_n,P_n}^2(D_n)}{L_{latency}^{ISL}} \qquad (4)$$

Given that adjacent tasks may be generated from different satellites, there exists at most one task queue on each satellite. we denote that task T_b and T_n are generated adjacently on the same satellite, one after another i.e., T_b followed by T_n. Further, we can compute that the time point when LEO satellite begins to process T_n is $t_n^{S_1} = \max(t_n^A, t_b^{O_1})$, the time point when LEO satellite gets ready to transmit T_n is $t_n^{O_1} = t_n^{S_1} + t_n^{P_1}$, the time point when HEO satellite starts to inference T_n is $t_n^{S_2} = \max(t_n^{O_1} + t_n^{tr_1}, t_n^{O_2})$, the time point when T_n is finished is $t_n^{O_2} = t_n^{S_2} + t_n^{P_2}$. Then, the time point when the LEO processes the task T_n finally is $t_n^{S_3} = \max(t_n^{O_2} + t_n^{tr_2}, t_b^{O_3})$, the time point when T_n is finished is $t_n^{O_3} = t_n^{S_3} + t_n^{P_3}$. Finally we compute the total inference time for T_n is $T_n^P = t_n^{O_3} - t_n^A$.

C. Problem Formulation

For the multi-exit model, each EE point has its specific level of accuracy due to different computing amounts. Thus the accuracy of EE point E_n is donated by A_{E_n}, where $0 \leq A_{E_n} < 1$. In this case, $A_{E_n} = 0$ if and only if $E_n = 0$. Feng et al. designed a cost computing function that tradeoff inference performance and latency by introducing a novel weight parameter [19]. This enables the selection of a descendant model that satisfies the specifications of resource-constrained satellites. Similarly, striving to enhance

task accuracy while completing as many tasks as feasible, we introduce an exponential function and define task gain G_n of the n-th task as

$$G_n = \lceil A_n \rceil + \frac{\alpha}{1 + e^{-\beta(A_n - A_{min})}} \quad (5)$$

where rounding up the inference accuracy $\lceil A_n \rceil$ quantifies the number of tasks completed in the system, while the *Sigmoid* function associated with the inference accuracy represents the user's satisfaction with task performance. Specifically, A_{min} is the minimum branchy model inference accuracy. We introduce two parameters $\alpha > 0$ and $\beta > 0$, where the performance improvement factor α controls the significance of task inference performance. And β is used to finely adjust the influence of changes in accuracy.

Finally, we formulate the task stream with N tasks in the system as an optimization problem:

$$\max_{E_n, P_n} \sum_{n=1}^{M} G_n$$

$$s.t. \quad t_1^{S_1} = t_1^A, \quad (6)$$
$$t_n^{O_3} \leqslant t_n^A + L_n,$$
$$0 \leqslant E_n \leqslant M,$$
$$0 \leqslant P_n \leqslant M_{E_n}.$$

where $n \in N$, and the first constraint initializes the first task, the second constraint offers the inference time limitation of the task, the last constraints provide the feasible range of E_n and P_n, respectively.

D. Problem Analysis

The objective of this problem is to efficiently tackle the adaptive task stream challenge, involving the tradeoff between inference performance and task latency. This is achieved by implementing efficient partitioning and offloading techniques for branch DNN models across the entire inference process. In this case, we consider the total number of combinations of EE and partition points as H. When selecting the hth combination of these points, the total inference time and task gain of the mth task are P_h^m and G_h^m, respectively. Here we transform the formulated problem to a group knapsack problem with H items, where each item need to be packed into the knapsack. The volume and value of the hth item from the mth groups correspond to P_h^m and G_h^m, respectively. The total capacity of the entire backpack is $t_M^A + L_M$. Therefore, the formulated problem maximizes the total value of items that can be placed into a backpack. Hence, the problem addressed in this work is classified as NP-hard.

IV. TASK GAIN-AWARE INFERENCE OFFLOADING ALGORITHM

To achieve the maximum task gain, this work introduces a task gain-aware method designed to identify the optimal EE and partition points for the task.

Algorithm 1: Task Gain-aware Decision

Input: N: the task number;
    D_i: the input data size of task T_i;
    L_i: the time constraint of task T_i;
    t_n^A: the arriving time of task T_n;
Output: Maximum task gain $\mathcal{G}$
1: Initialize $\mathcal{G}(0,t) = 0$ // Initialization
2: **for** $i = 1$ to N **do**
3:    **if** $t_{i-1}^A + L_{i-1} < t_i^A$ **then**
4:       $\mathcal{G}(i-1, t_i^A) = \mathcal{G}(i-1, t_{i-1}^A + L_{i-1})$
      // Update the recursion task gain value.
5:    **end if**
6:    **for** $j = t_i^A$ to $t_i^A + L_i$ **do**
7:       Update $\mathcal{G}(i,j)$ according to Eq. (7)
8:    **end for**
9: **end for**

Algorithm 2: Optimal Point Selection

Input: M: the number of EE points;
    $\{M_k | k = 1, ..., M\}$: the number of layers of the EE points k;
    D_n: the input data size of T_n;
    L_n: the time limitation of T_n;
    $L_{latency}^{ISL}$: the inter-satellite transmission rate;
    $f_{i,j}^d$: the predictive model in the work
Output: Optimal E_n, P_n
1: **for** $k = M$ to 1 **do**
2:    $E_n = k$
3:    $C_n = \min_{P_n = 1, ..., M_k} \{t_n^{O_3} - t_n^A\}$
    // Divide the model EE points in sequence and select the smallest computing time.
4:    **if** $C_n \leq L_n$ **and** $A_n \geq A_n^*$ **then**
5:       record E_n, P_n, A_n
6:    **end if**
7:    **return** E_n, P_n when A_n is maximum
8: **end for**
9: **return** NULL

We begin by calculating the maximum gain of each task at each time slot, and ultimately obtain the overall maximum gain using Algorithm 1. In this case, we can forecast task stream information including the input data, arrival time, time limitation, and total task number. We define the maximum gain $\mathcal{G}(i,j)$ with the ith tasks and the limitation of jth time slots:

$$\mathcal{G}(i,j) = \max_{t_i^A \leq t < j} \{\mathcal{G}(i-1,j), \mathcal{G}(i-1,t) + \lceil S(D_i, j-t) \rceil + \alpha \frac{1}{1 + e^{-\beta(S(D_i, j-t) - A_i^*)}}\} \quad (7)$$

Meanwhile, $S(D,t)$ is used to denote the highest accuracy of EE points with the data size D and latency constraint t. Algorithm 1 first initializes the array $\mathcal{G}(0,t)$ ($1 \leq t \leq t_M^A +$
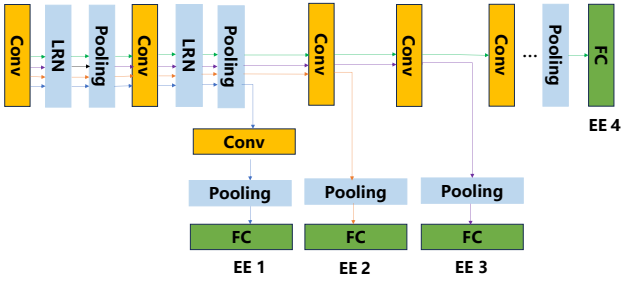
Fig. 2: AlexNet model consists of 4 EE points. When we consider task failure as another particular case when starting, AlexNet model has 5 EE points.

L_M). Then, when the task i arrives, if the sum of the arrival time of the previous round and the latency constraint is less than the current task's arrival time, we update $\mathcal{G}(i-1, t_i^A)$ to $\mathcal{G}(i-1, t_{i-1}^A + L_{i-1})$. At the same time, for each time point j, we loop from t_i^A to $t_i^A + L_i$ recursively, comparing the maximum gain of the previous round $\mathcal{G}(i-1, j)$. Thus we can obtain the maximum $\mathcal{G}(i, j)$.

Algorithm 2 selects the optimal points with both the highest accuracy and time constraints through an exhaustive search of the EE and partition points. When the task T_n arrives, for each branch in multi-exit model, we select the partition point in sequence and calculate the total processing time. Then, we select the minimum total processing time and the corresponding partition point is the best choice for the current task and branch. Additionally, since each branch of the multi-exit model corresponds to a task accuracy A_n, we can obtain a series of A_n for the current task and the corresponding branch. Among those, we select the maximum A_n. Finally, the corresponding EE and partition points are the E_n and P_n that the current task should be associated with.

V. SIMULATION EVALUATIONS

In this section, we evaluate the performance of our proposed algorithm with the following two algorithms:

- Greedy: The algorithm makes the offloading decisions according to the information of the current tasks system, disregarding the interaction between adjacent tasks.
- Random: The algorithm makes randomly the offloading decisions without task system information.

A. Experiment Setup

We use the satellite constellation StarLink and the satellite GPS system as references to simulate inter-satellite connectivity in orbit, and use various computing capabilities of LEO satellites by adjusting the number of CPUs, operating peaks, etc. The dataset used in the simulation is CIFAR-10 and each image is categorized into one of ten classes. The dataset is also used for model training, simulating the tasks, and evaluating the proposed algorithm. Additionally, we utilize widely-used classification as our evaluated DNN-based inference application and adopt an 8-layer AlexNet model

including 4 EE points. AlexNet model we consider is shown in Fig. 2. When we regard task failure as another particular case when starting, this model consists of 5 EE points. Therefore, after training, the corresponding inference accuracy of 5 EE points is [0, 0.527, 0.623, 0.697, 0.743].

In the evaluation, task generation follows *Bernoulli distribution* with $p = 0.1$, ensuring that at most one task is generated during each time slot. The input is the images to be inferenced in the task system, where the number of images follows a normal distribution within [1,10]. Every time slot is set to 3 seconds. In addition, the hyperparameters α and β are set to 0.1 and 16, respectively.

We consider three vital metrics to evaluate the proposed algorithm: the overall task gain, task completion rate, and average task latency. The overall task gain refers to Eq. (6), which considers the two task performance in the entire system. Task completion rate is calculated by dividing the number of tasks completed by the number of tasks arrived, which allows us to study the impact of system decisions on task completion from the user's perspective. The average task latency is calculated by summing the differences between the completion time of completed tasks and their arrival time, and then dividing it by the number of tasks completed, which indicates the impact of system decisions on the average task latency.

B. Experiment Results

As illustrated in Fig. 3, our algorithm consistently demonstrates superior performance compared to the baseline algorithms. As expected, with the intensive number of task arrivals, our algorithm demonstrates approximately linear growth in the overall system objective. Notably, when the system's task arrival number is 300, our algorithm outperforms the greedy and random algorithms by 19.39% and 61.48%, respectively. This improvement can be attributed to our algorithm's consideration of the interdependence between adjacent tasks in the task system, which incorporates both the average accuracy of tasks and the number of completed tasks. Furthermore, Fig. 4 demonstrates that our algorithm consistently outperforms the Greedy and Random algorithms, regardless of variations in the task arrival rate. Additionally, the excellence of our algorithm becomes progressively more apparent as task arrival rates increase, highlighting its remarkable performance in complex task scenarios.

Next, we investigate the task completion rate of all algorithms under varying numbers of task arrivals, as shown in Fig. 5, and find our algorithm's superiority over the other algorithms. When the task arrival number is 300, our algorithm achieves an 18.1% and 55.5% improvement compared to the Greedy and Random algorithms, respectively. This advantage originates from our algorithm's ability to determine the trade-off between task accuracy and latency constraints. By avoiding excessive time dedicated to individual tasks, our algorithm achieves a significantly higher task completion rate.

Finally, we investigate the average task latency of all algorithms under varying numbers of task arrivals, as illustrated
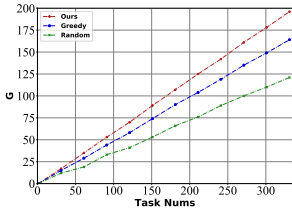
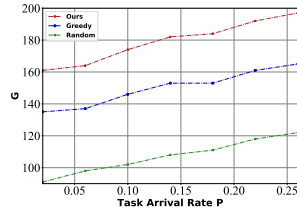Fig. 3: The total gain in different the number of tasks.

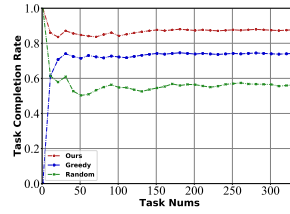Fig. 4: The total gain under the different task arrive rate p.

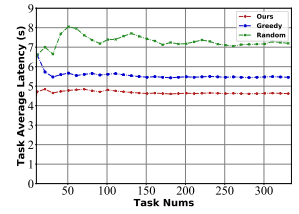Fig. 5: The task completion rate in different task numbers.

Fig. 6: The task average latency in different task numbers.

in Fig. 6. Our findings reveal that our algorithm consistently demonstrates lower latency than other algorithms. When the task arrival number is 300, our algorithm reduces latency by 35.6% and 15.3% compared to the greedy and random algorithms, respectively. This reduction can be attributed to our algorithm's comprehensive consideration of both average task accuracy and the number of completed tasks. By prioritizing efficient task completion while maintaining accuracy requirements, our algorithm effectively minimizes average task latency.

VI. CONCLUSION

The paper proposes a framework for collaborative intelligent inference for various satellite inference tasks based on multi-exit model and model partition. The framework aims to make an efficient trade-off between task completion and the average inference accuracy of DNN-based applications. We analyze the problem theoretically and design a task gain-aware algorithm to solve this issue. Simulation evaluations show that the proposed algorithm outperforms that of baseline algorithms, especially with intensive tasks and strict time constraints.

REFERENCES

[1] I. Del Portillo, B. G. Cameron, and E. F. Crawley, "A technical comparison of three low earth orbit satellite constellation systems to provide global broadband," *Acta astronautica*, vol. 159, pp. 123–135, 2019.

[2] (2023) Ucs satellite database. [Online]. Available: https://www.ucsusa.org/resources/satellitedatabase

[3] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, 2023.

[4] B. Denby and B. Lucia, "Orbital edge computing: Nanosatellite constellations as a new class of computer system," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 939–954.

[5] C. Wang, Y. Zhang, Q. Li, A. Zhou, and S. Wang, "Satellite computing: A case study of cloud-native satellites," *arXiv preprint arXiv:2307.08530*, 2023.

[6] Z. Lai, Q. Wu, H. Li, M. Lv, and J. Wu, "Orbitcast: Exploiting mega-constellations for low-latency earth observation," in *2021 IEEE 29th International Conference on Network Protocols (ICNP)*. IEEE, 2021, pp. 1–12.

[7] O. Laux, D. Poncet, R. Mager, and K. Schoenherr, "Status of the european data relay satellite system," in *2012 International Conference on Space Optical Systems and Applications (ICSOS)*, 2012, pp. 9–12.

[8] Q. Zhang, X. Che, Y. Chen, X. Ma, M. Xu, S. Dustdar, X. Liu, and S. Wang, "A comprehensive deep learning library benchmark and optimal library selection," *IEEE Transactions on Mobile Computing*, 2023.

[9] G. Giuliari, T. Klenze, M. Legner, D. Basin, A. Perrig, and A. Singla, "Internet backbones in space," *ACM SIGCOMM Computer Communication Review*, vol. 50, no. 1, pp. 25–37, 2020.

[10] N. Cheng, F. Lyu, W. Quan, C. Zhou, H. He, W. Shi, and X. Shen, "Space/aerial-assisted computing offloading for iot applications: A learning-based approach," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1117–1129, 2019.

[11] B. Denby, K. Chintalapudi, R. Chandra, B. Lucia, and S. Noghabi, "Kodan: Addressing the computational bottleneck in space," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, 2023, pp. 392–403.

[12] W. He, S. Guo, S. Guo, X. Qiu, and F. Qi, "Joint dnn partition deployment and resource allocation for delay-sensitive deep learning inference in iot," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9241–9254, 2020.

[13] W. Miao, Z. Zeng, L. Wei, S. Li, C. Jiang, and Z. Zhang, "Adaptive dnn partition in edge computing environments," in *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2020, pp. 685–690.

[14] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 1, pp. 615–629, 2017.

[15] A. E. Eshratifar, M. S. Abrishami, and M. Pedram, "Jointdnn: An efficient training and inference engine for intelligent mobile cloud computing services," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 565–576, 2019.

[16] W. Ju, D. Yuan, W. Bao, L. Ge, and B. B. Zhou, "Edeepsave: Saving dnn inference using early exit during handovers in mobile edge environment," *ACM Transactions on Sensor Networks (TOSN)*, vol. 17, no. 3, pp. 1–28, 2021.

[17] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge ai: On-demand accelerating deep neural network inference via edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 447–457, 2019.

[18] J. Song, Z. Liu, X. Wang, C. Qiu, and X. Chen, "Adaptive and collaborative edge inference in task stream with latency constraint," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.

[19] B. Fang, X. Zeng, and M. Zhang, "Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 115–127.