SatCooper: Enhancing Cooperative Inference Analytics for Satellite Service via Multi-Exit DNNs

Qiyang Zhang, Shangguang Wang, *Senior Member, IEEE*, Jinglong Guan, Praveen Kumar Donta, *Senior Member, IEEE*, Xiao Ma, RangaRao Venkatesha Prasad, *Senior Member, IEEE*, Schahram Dustdar, *Fellow, IEEE*, and Xuanzhe Liu, *Senior Member, IEEE*

***Abstract*—As a key technology of intelligent satellite-enabled services in B5G or 6G networks, deploying Deep Neural Networks (DNN) models on satellites has been a notable trend, catering to the daily demand for extensive computing-intensive and latency-sensitive tasks. The computing resources are strategically deployed on satellites where sensor data is generated or collected, facilitating the fine-grained computational inference of DNN-based tasks. However, no prior study has comprehensively explored the crucial inference challenges – e.g., the trade-off between the number of tasks completed and accuracy and partitioning models in multi-exit models – in the resource-constrained space environment. Effective scheduling frameworks cater to various streams of inference tasks are scarce because inference performance may deviate from the ideal situation due to changes in task system status, such as task profiles and network state. To this end, we first formulate a gain-aware in-orbit computing inference problem to strike a proper trade-off between inference latency and the number of tasks completed by dynamically selecting optimal early exit points and model partitioning points. We propose an offline dynamic programming-based algorithm that provides an effective solution when comprehensive system details are to be predicted. We have developed an online learning-based method to schedule inference tasks with uncertain and dynamic system statuses in real-world situations. Our evaluation shows that, compared to baseline methods, the online learning-based algorithm can improve task gain by an average of 87.3% across various tasks.**

***Index Terms*—Satellite service, satellite networking, task inference, multi-exit DNN.**

Received 22 January 2025; revised 17 March 2025; accepted 26 March 2025. Date of publication 31 March 2025; date of current version 6 August 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62325201, Grant 62032003, and Grant 62372061, and in part by Young Elite Scientists Sponsorship Program by CAST under Grant 2023QNRC001. Recommended for acceptance by L. Zhu. *(Corresponding author: Xuanzhe Liu.)*

Qiyang Zhang and Xuanzhe Liu are with the School of Computer Science, Peking University, Beijing 100871, China (e-mail: qiyangzhang@pku.edu.cn; liuxuanzhe@pku.edu.cn).

Shangguang Wang, Jinglong Guan, and Xiao Ma are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100088, China (e-mail: sgwang@bupt.edu.cn; mr_guanjl@bupt.edu.cn; maxiao18@bupt.edu.cn).

Praveen Kumar Donta is with the Department of Computer and Systems Sciences, Stockholm University, 16455 Kista, Sweden (e-mail: praveen@dsv.su.se).

RangaRao Venkatesha Prasad is with the Networked Systems (NS), Delft University of Technology (TU Delft), 2628 Delft, CD, Netherlands (e-mail: r.r.venkateshaprasad@tudelft.nl).

Schahram Dustdar is with the Distributed System Group, TU Wien, 1040 Wien, Austria, also with UPF, 08002 Barcelona, Spain (e-mail: dustdar@dsg.tuwien.ac.at).

Digital Object Identifier 10.1109/TMC.2025.3556457

I. Introduction

CONTINUOUS advancements in space exploration have fueled the rapid expansion of satellite constellations. Many popular companies, such as Telesat, Amazon, and SpaceX [1], have built satellite constellations with a dedicated focus on achieving goals like Earth Observation (EO) [2]. The applications periodically collect EO data and provide ubiquitous coverage during disaster management [3], [4]. In this scenario, EO applications can employ in-orbit computation to process sensor-acquired data for real-time processing and reducing the downlink data volume. For example, a disaster response department captures EO images and requires rapid image results from a wildfire to mitigate risks to human lives and property. Recently, the trend has been to launch nano- or micro-satellites that use Commercial Off-The-Shelf (COTS) components, which are limited in power and computation but have a fast turnaround. The emerging machine learning (ML) algorithms facilitate EO applications, especially Deep Neural Network (DNN) models. As such, the framework of in-orbit capture and computational inference is expected to play a crucial role in the B5G or 6G networks by providing diverse, efficient inference services.

Due to the characteristics of low orbit, a single Low Earth Orbit (LEO) satellite can only capture a limited area within each image and does not cover consecutive regions due to its high-speed movement relative to the ground (around 8 kmps). Current LEO satellites, such as Planet's Dove satellites,[1] rely heavily on multiple satellite collaboration to achieve wide-area observation coverage. Moreover, from a computing perspective, the limited computing capability available for each satellite presents significant challenges when handling highly demanding yet complex inference tasks. Previous efforts have primarily focused on offloading onboard tasks to the ground, which is considered a promising solution [5], [6], [7], [8], [9], [10], [11]. However, this solution often faces significant latency (ranging from a few hours to days even) due to intermittent connection with the ground station. These methods can significantly impact performance when inference services exceed a predefined deadline. For example, degraded performance may lead to decision errors and economic losses for services requiring in-orbit results analysis for decision-making. Some efforts on satellite computing [12], [13] demonstrate the feasibility of scheduling

[1] https://www.planet.com/our-constellations/

computing workloads on LEO satellite constellations, and such architectural optimizations can enhance inference performance. However, these methods cannot effectively manage diverse and computationally intensive in-orbit inference services. Therefore, achieving ubiquitous coverage and efficient computing with individual satellites is challenging.

To deal with this issue, the promising solution involves a cooperative paradigm for satellite image capture and computing, such as DNN partitioning in multi-exit models between Medium Earth Orbit (MEO) and LEO satellites. Automatic switching between MEO and LEO satellites occurs based on service requirements, ensuring efficient in-orbit service continuity. These satellites in distinct orbits offer unique advantages: MEO satellites excel at achieving extensive imaging coverage, whereas LEO satellites specialize in capturing high-resolution images. Compared to collaborations among multiple LEO satellites, MEO satellites could achieve similar objectives while reducing image size and conserving computing resources. Additionally, Starlink demonstrated the feasibility of laser-based Inter-Satellite Links (ISLs). Planet and Telesat have announced their intentions to investigate radiofrequency ISL spanning various orbits, from LEO to MEO satellites [14], [15]. Motivated by integrating computing capabilities and image capture across satellites in different orbits, we argue that a cooperative image capture and computing mechanism is needed to achieve high-performance inference services. This paper introduces an efficient framework for cooperative inference involving multiple orbital satellites in a resource-constrained space environment. To the best of our knowledge, this is the first work to thoroughly investigate inference in dynamic, task-driven satellite imaging services. The approach includes: (a) Partitioning DNN models between one MEO satellite and multiple LEO satellites; (b) To meet the demand of current EO tasks, which range from ubiquitous coverage to detailed observation due to specific task requirements, multiple satellites must collectively perform EO observations, each handling specific computational tasks; and (c) Due to the limited computing capabilities of each satellite, the task owner distributes inference tasks across multiple satellites, each equipped with COTS hardware.

For the inference task, the task owner processes the initial portion of the DNNs and delegates the remaining part to other task executors. Addressing these issues in a demanding computing environment requires advanced artificial intelligence mechanisms, specifically multi-exit models. Multi-exit mechanisms are introduced to reduce significant computational workloads by selectively activating optimal early-exit (EE) points and Model Partitioning (MP) points [16], [17]; otherwise, it leads to performance degradation of inference services. Determining each task's optimal EE and MP points based on current system conditions remains challenging. The simultaneous management of EE and MP points adds complexity to the inference system, especially when dealing with evolving task profiles. We aim to consider the trade-off between maintaining satisfactory task accuracy and task completion, which is not trivial. When prioritizing high inference performance for each task, some tasks may fail to meet their deadlines due to prolonged waiting times. When we focus solely on increasing system throughput, overall inference performance may not meet time constraints. We propose an exponential-based task gain function that balances latency and inference accuracy to address this challenge. The optimal solution to the problem can maximize task gain by dynamically selecting the optimal EE points and MP points based on each model's characteristics and considering the tasks' profiling.

On the other hand, analyzing streaming tasks under tight time constraints poses significant challenges, especially under extreme conditions [18] such as poor network conditions and excessive task arrivals. When information on dynamic tasks (e.g., task arrivals and network state) is known in advance, a Dynamic Programming (DP)-based cooperative inference strategy can efficiently address the optimization problem. However, there are limitations in real-world applications with uncertain task information, meaning decision-makers cannot obtain complete task information. In response to this challenge, we developed an online algorithm based on Deep Reinforcement Learning (DRL). This algorithm adjusts adaptive strategies to environmental interactions, including uncertainty and dynamics.

We evaluate the proposed algorithm using a combination of task-driven simulations and hardware emulation on a general-purpose EO classification application. The results demonstrate that, compared to the baseline methods, the proposed method can significantly improve the defined task gain and task completion rates while reducing the average task latency. Moreover, despite the absence of uncertain task information, the proposed method closely approaches the performance of the DP-based method with theoretical upper bound performance. The primary contributions of this work are summarized below:

Problem Formulation: We formulate an optimization problem for satellite inference services by considering inference task requirements and model modification, i.e., ubiquitous coverage and high-resolution satellite imaging. This formulation aims to balance task latency and accuracy before the deadlines.

Algorithm: We extend the introduced dynamic programming-based task gain-aware algorithm to obtain an efficient solution when system dynamics can be predicted accurately [19]. We propose a Deep Reinforcement Learning (DRL)–based online algorithm designed to handle dynamic and uncertain system statuses.

Evaluation: Extensive evaluations through simulations show the effectiveness of the proposed algorithm in meeting inference time constraints. Our approach improves the defined task gain by 87.3% across various task numbers compared to baseline algorithms. The rest of the work is structured as follows: After reviewing the background and related work in Section II, Section III presents the system model and problem formulation. Section IV introduces the two algorithms, while Section V discusses the experimental results. Finally, Section VI concludes the paper and outlines its limitations.

II. Background and Related Work

A. Background

MEO satellites, positioned at 8,000 km altitude, offer extensive coverage capabilities due to their geostationary positions. These satellites can monitor large-scale geographical areas continuously, but their distance from the Earth's surface results in lower-resolution imagery [20], [21]. Therefore, MEO satellites

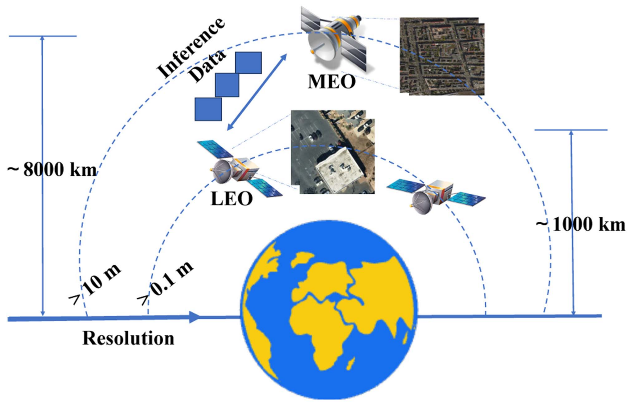


Fig. 1. Example of satellite service for ubiquitous coverage and high-resolution satellite imaging.

are commonly used for general monitoring but are less suitable for high-resolution imaging services. In contrast, LEO satellites, positioned in orbits ranging from hundreds to thousand kilometers, provide high-resolution images. Their proximity to Earth enables them to capture fine-grained details, making them valuable for detailed monitoring. Therefore, as shown in Fig. 1, the choice of different orbits depends on specific application requirements for high coverage and resolution of satellite images.

Leveraging MEO satellites establishes a stable and efficient link between MEO and LEO satellites, resulting in expedited data delivery suitable for time-sensitive applications. The utilization of modern MEO satellites, equipped with long-distance laser communication components capable of providing data rates as high as gigabits per second (Gbps), is the fundamental reason for achieving these elevated data rates [22]. Furthermore, MEO and LEO satellites can maintain connectivity owing to their intrinsic characteristics, such as extended-duration vision and communication capabilities [23], [24]. Communication between these two satellites can be facilitated through co-frequency band operation, reducing frequency band conversion and enhancing cross-system compatibility. Moreover, their high-altitude co-orbiting trajectory enables prolonged mutual visibility, providing extended communication windows [25]. On the other hand, the rapid advancement of onboard computing capability is evidenced by significant progress in integrating increasingly powerful computing capabilities into satellites, such as platforms like Raspberry Pi and NVIDIA Jetson. For example, in 2020, the European Space Agency deployed a neural network-based cloud detector as part of their ϕ-Sat-1 mission.

This paper considers a potential application as the case study, where the dynamic utilization of MEO and LEO satellites. Each of these satellites has unique coverage and imaging capabilities, ensuring an effective response to the ever-changing demands of disaster management. In this case, MEO and LEO satellites have distinct roles during different stages: (1) Getting Ready and Warning. To prepare for disasters, monitoring large areas for early signs is essential. MEO satellites are well-suited for this task due to their broad coverage capabilities; (2) Responding and Checking the Damage. After a disaster, LEO satellites, positioned close to Earth, capture high-resolution images that play a vital role in understanding the extent of damage; (3) Recovery and Reconstruction. Once the immediate issues are addressed, the focus shifts to large areas to evaluate the progress of recovery efforts and their impact on broader regions.

B. Related Work

This work facilitates a cooperative inference stream system involving multiple satellites, utilizing DNN models by partitioning multi-exit models. In this approach, task owners distribute subsequent inference tasks across different satellites to enhance inference accuracy and overall system throughput. This section reviews the related works.

Mega-constellations: Most LEO satellites typically employ a traditional architecture, where their primary function is transmitting data rather than processing data. However, data transmission between satellites and ground stations faces notable impediments due to unreliable communication links and intermittent connection. This leads to a significantly reduced transmission rate compared to the onboard data generation rate, contributing to a substantial increase in the overall latency of all tasks. Furthermore, mega-constellations utilize inter-satellite data processing, enabling data exchange and processing between satellites [26]. This method facilitates satellite computing, where onboard data processing is vital in enhancing satellite services.

In-orbit Computing: In-orbit computing proposes shifting the processing tasks to satellites, reducing bandwidth consumption by introducing emerging COTS hardware. This approach leverages onboard computing resources to provide satellite services [12], [13], [27]. For instance, both orbital edge computing [12] and Kodan [13] decrease the number of images transmitted to Earth by filtering and prioritizing valuable ones. Orbital edge computing utilizes satellites within a constellation as computational pipelines that seamlessly distribute computing workloads. Kodan seeks to train the optimal model for specific applications by concentrating on the computation of each satellite. However, these approaches fail to address the challenges posed by diverse and computing-intensive satellite applications. Recent mega-constellations, such as Starlink and Kuiper, consist of thousands of satellites with laser ISLs, providing broadband Internet service with low latency and high-speed transmission. However, the constrained computing capabilities of satellites impose a substantial burden on their operations, involving significant challenges in satisfying the constraints of satellite service. Therefore, we investigate a novel strategy based on a multi-exit model and model partitioning by allocating fine-grained computational resources cooperatively among different satellites.

Model Partitioning for Cooperative Task Inference: Collaborative inference is developing rapidly, with most existing research focusing on deep neural network operator acceleration [28], cloud or server-based computing [29], [30], and multi-device collaboration on the ground [31], [32], [33]. However, this area of work is orthogonal to the contributions of this work. Furthermore, previous efforts suggest that model partitioning can delegate partial computational tasks from task owners to other edge devices [4], [34], [35]. Notably, Neurosurgeon [36] presented layer-wise partition strategies to offload computation and brought additional speedup for inference. The collaborative framework JointDNN [37] was designed to optimize resources on the edges by incorporating both model training and inference. The synergy between these two aspects contributes to enhanced efficiency and performance in edge computing environments. DDNN [38] proposed a distributed computing continuum system to minimize the communication data size. Other works address an optimization problem by selecting the most suitable DNN depth for model partitioning. Edgent [39] and SPINN [40] guarantee inference accuracy, while the optimization in [41] saves energy. In contrast, aiming at the efficient inference of AI-enabled satellite services, our work pursues a distinct objective to achieve a trade-off between task accuracy and the number of completed tasks.

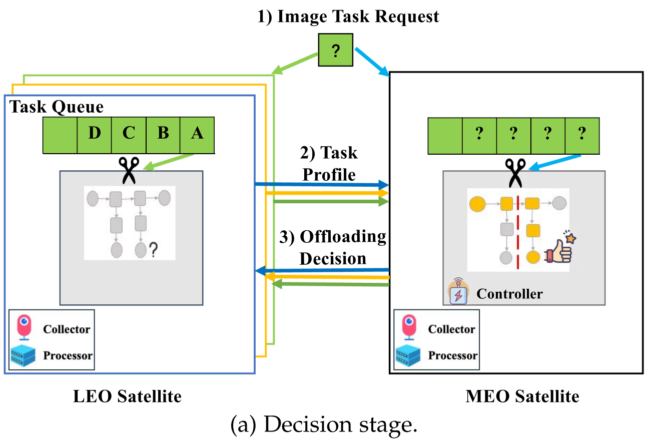
Multi-exit Models for Boosting Task Inference: The models with multiple branches are denoted as multi-exit models. Multi-exit models facilitate adaptive inference in conjunction with model partitioning, achieved by modifying the structure of pre-trained models. There has been a growing interest in partitioning multi-exit models as a strategy to enhance DNN performance. BranchyNet [42] as the first to introduce an inference engine that enables accurate predictions to exit the backbone network early through EE points when a certain confidence level is achieved. eDeepSave [43] aimed to enhance inference performance using the EE mechanism to prevent interruptions when transitioning to the mobile network. Edgent [39] introduced a DNN partitioning model that aims to maximize inference accuracy while adhering to latency constraints. While multi-exit partitioning models for satellite services seem promising, they are yet to be fully explored. Therefore, in a more realistic setting with uncertain task arrivals and the current state of the task, we design an efficient solution to facilitate decision-making through adaptive learning-based scheduling in response to dynamic system states.

III. Problem Definition

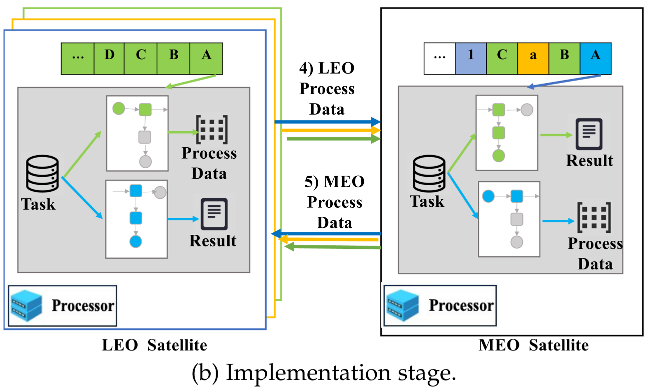
Fig. 2 depicts a time-slotted stream model comprising multiple LEO satellites and an MEO satellite. LEO satellites fulfill the roles of capturing and processing data. In addition to these functions, the MEO satellite also serves as the controller for executing decision-making tasks. MEO satellites capture and queue MEO imaging tasks for processing, while LEO satellites relay information about their imaging tasks to the MEO satellite for further handling. Pre-trained multi-exit models are deployed on each satellite to support subsequent inference tasks on each satellite.

The workflow of the whole inference system involves two main stages, described as follows:

(1) The satellites receive the task requests, including the current task requirements, such as task type, data size, and time constraints. Specifically, MEO imaging tasks are captured by MEO satellites and queued for processing. LEO satellites transmit information about LEO imaging tasks to an MEO satellite. Based on the satellite imaging requirements, an MEO satellite evaluates inference solutions and transmits their decisions to LEO satellites.



(a) Decision stage.



(b) Implementation stage.

Fig. 2. The workflow of collaborative task stream inference system.

(2) LEO satellites perform imaging tasks by gathering inference information, processing the initial portion, and forwarding the remainder to the MEO satellite. MEO satellite manages these tasks according to the First-Come-First-Served (FCFS) principle. In MEO imaging tasks, the initial part is processed on the MEO satellite before the remaining portion needs to be transmitted to the LEO satellites. When the MEO satellite finishes processing, LEO satellites receive their results. We ignore the MEO satellite's time to deliver the final results for the LEO imaging tasks [44].

A. Satellite Inference Task Model

Let the n^{th} task in the task-slotted stream be denoted by $T_n = (S_n, D_n, L_n, t_n^A)$, where task requirement type is denoted by S_n, the task size is represented by D_n, the maximum allowable time constraint is denoted by L_n, and the arrival time is represented using t_n^A. The relation between L_n and D_n is proportional, expressed as $L_n = k \times D_n$, $k \in N$. In addition, M indicates the total number of EE points and the number of layers of i^{th} EE point is denoted as M_i. For the n^{th} task, P_n and E_n donate the MP and EE points, respectively, with the constraints $0 \leq E_n \leq M$, $0 \leq P_n \leq M_{E_n}$. Table I summarizes the parameters and their meanings.

According to [37], we tailor the transmission data size and processing time during handover to specific applications through profiling. The prediction model demonstrated high accuracy in

TABLE I
SUMMARY OF SYSTEM PARAMETERS

Parameters	Descriptions
N	the number of tasks
p	the task arrival rate in the system
D_n	the data size
L_n	the warning task deadline
t_n^A	the arriving time of task
M	the number of EE points
E_n	the EE point of n^{th} task
P_n	the MP point of n^{th} task
α, β	the hyperparameters of optimization function
$f_{i,j}^d(D_n)$	the predicted inference of the j^{th} layer in i^{th} EE point of the model
$f_{i,j}^D(D_n)$	the predicted output data size of the j^{th} layer in i^{th} EE point of the model
T_n	the n^{th} task
A_n	the accuracy of the P_n^{th} point for n^{th} task

practical scenarios, ensuring reliability while minimizing execution costs. Thus, we use a linear regression-based prediction model for each layer in the multi-exit model, considering both LEO and MEO satellites, to accurately estimate output data size and inference time. For this discussion, we focus on the LEO satellite. Inference time is recorded using a *timeline* and modeled by fitting linear functions to the inference times associated with each model layer. The predicted inference time, $f_{i,j}^d(D_n)$, corresponds to the input data size D_n at the j^{th} layer of the i^{th} EE point. This is executed in LEO satellites if $d = 0$ or MEO satellites if $d = 1$. Similarly, the predicted output data size at j^{th} layer of the i^{th} EE point is denoted as $f_{i,j}^D(D_n)$, where D is 2 by default.

B. Inter-Satellite Communication Model

Communication of an ISL depends on satisfying the following three criteria simultaneously [6]: (1) There exists no physical obstruction between two satellites, and links between satellites must be positioned in the same direction; (2) Satellites are visible to each other. In this case, MEO satellites consistently maintain proximity to LEO satellites; (3) The cooperative satellites remain connected if their antennas are within a specified pointing angle. The pitch angle θ of MEO satellite and LEO satellites must fall within the antenna scanning of LEO satellites, i.e., $\sigma_{min} < \theta < \sigma_{max}$, where σ_{min} and σ_{max} represent the lower and upper bounds of the antenna scanning range for LEO satellites. Similarly, the same applies to MEO satellites.

The transmission time of inter-satellite links $L_{latency}^{ISL}$ consists of the following two parts: (1) Transmission delay, which is the time cost to transmit all the data from the first bit, can be calculated by the size *len(D)* divided by ISL data rate $w(S_L, S_M)$; (2) Propagation delay, which is the time required for electromagnetic radiation to travel the distance $Q_{(S_L,S_M)}$, can be expressed as the ratio of the distance to the propagation rate of the electrical signal, typically approximated as the speed of light c). Here, the transmission latency of ISL is,

$$t_n^{tr} = \frac{len(D)}{w(S_L, S_M)} + \frac{Q_{(S_L,S_M)}}{c}, \tag{1}$$

where S_L and S_M denote the LEO satellite and MEO satellite, respectively. D and $Q_{(S_L,S_M)}$ represent the size of data transmitted and the length of the transmission link between MEO and LEO satellites, respectively. The data transmission rate ($w(S_L, S_M)$) is calculated as follows:

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

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

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

where transmit power is written as P_t, G_t, and G_r are the receive gain and the transmit gain, respectively. f_c represents the carrier frequency.

C. Computational Model

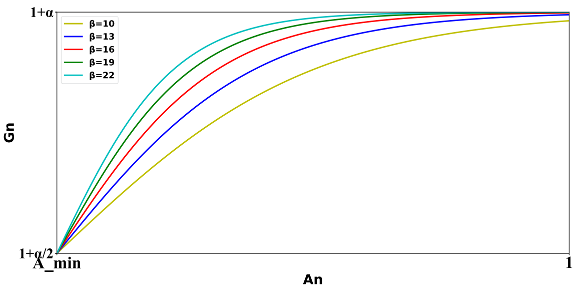
We aim to select the optimal points for EE and MP to meet the deadline constraints for transmitting task data between satellite nodes. The entire system can be divided into three distinct stages based on how E_n and P_n are chosen: First, when the imaging task is processed on the LEO satellites, the first stage processing time for task T_n is $t_n^{P_1} = \sum_{i=1}^{P_n} f_{E_n,i}^0(D_n)$, or if no processing occurs, $t_n^{P_1} = 0$; Second, when imaging task is processed on the LEO satellites, the second stage processing time for task T_n 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 processed on the MEO satellite, the third stage processing time for task T_n is $t_n^{P_3} = 0$, or $t_n^{P_3} = \sum_{i=P_n+1}^{M_{E_n}} f_{E_n,i}^0(D_n)$. Notably, the transmission data size between LEO and MEO satellites is denoted by

$$len(D) = f_{E_n,P_n}^D(D_n), \tag{4}$$

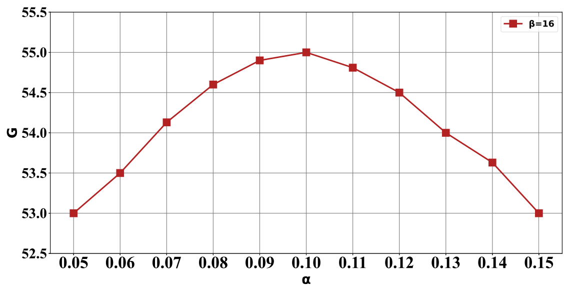
Since the adjacent tasks may come from distinct satellites, each satellite maintains at most one task queue. T_b and T_n denote the sequential tasks generated on the same satellite, where T_b precedes T_n. In this case, the time at the LEO satellite starts processing the task T_n is $t_n^{S_1} = \max(t_n^A, t_b^{O_1})$, the time when LEO satellite starts to transmit the task T_n is $t_n^{O_1} = t_n^{S_1} + t_n^{P_1}$, the time when MEO satellite starts inference the task T_n is $t_n^{S_2} = \max(t_n^{O_1} + t_n^{tr_1}, t_b^{O_2})$, the time when the task T_n is completed is $t_n^{O_2} = t_n^{S_2} + t_n^{P_2}$. Furthermore, the time when the LEO satellite finally starts processing the task T_n is $t_n^{S_3} = \max(t_n^{O_2} + t_n^{tr_2}, t_b^{O_3})$, the time when T_n is completed is $t_n^{O_3} = t_n^{S_3} + t_n^{P_3}$. Finally, we compute the total inference time for T_n is $T_n^P = t_n^{O_3} - t_n^A$.

D. Problem Formulation

In the multi-exit DNN model, each EE point corresponds to a specific level of accuracy associated with different computational loads. Therefore, the accuracy of EE point E_n is represented by A_{E_n}, where $0 \le A_{E_n} < 1$. In this scenario, $A_{E_n} = 0$ if and only if $E_n = 0$. Feng et al. [45] proposed a



(a) The function G_n under various β.



(b) The function G_n under various α.

Fig. 3. Parameter analysis.

function associated with computing cost, introducing a novel weight parameter to balance inference performance and latency. This facilitates the selection of a descendant DNN model that meets the resource-constrained requirements of the satellites. Similarly, striving to enhance task accuracy while completing as many tasks as possible, we introduce a novel exponential function, and the task gain G_n of the task n is expressed as

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

where the first component of the gain function quantifies the tasks completed within the task stream by approximating the inference accuracy. The second component is a sigmoid function associated with the inference accuracy, which expresses the user's satisfaction with the model's performance. Specifically, A_{min} represents the minimum inference accuracy. A_n is the accuracy of the inference result, where $A_n \in [A_{min}, 1) \cup 0$, 0 indicates that the task has not been completed successfully. We introduce two hyperparameters, α and β, both greater than 0, serving as weighting parameters to assess task inference performance. The first parameter α balances the task completion outcome against the inference accuracy level. On the other hand, the second parameter β is employed to regulate the impact of accuracy variations on the extent of the function value change. Theoretically, increasing the hyperparameter α magnifies the influence of the second component on the task gain value. Similarly, increasing the hyperparameter β makes the effect of the accuracy difference on the function value more apparent.

Fig. 3(a) illustrates the impact of inference accuracy A_n on the function G_n across different β, where $1 + \alpha$ represents the maximum gain for performing the task, while $1 + \frac{\alpha}{2}$ denotes the gain when the task is executed with the minimum inference accuracy. Fig. 3(b) illustrates the function G_n on different α when β is optimal. Consistent with theoretical analysis, our results highlight the critical importance of hyperparameter tuning: A low β requires significant improvements in accuracy to enhance task performance, while a high β enables substantial task gains with only marginal accuracy improvements; A low α prioritizes task completion over achieving higher accuracy, while a high α excessively emphasizes accuracy at the expense of task completion. Consequently, achieving optimal performance requires a careful balance between these parameters.

In this paper, we define the task stream, comprising N tasks within the system, as an optimization problem:

$$\begin{aligned} \max_{E_n, P_n} \; & \sum_{n=1}^{N} \lceil A_n \rceil + \frac{\alpha}{1 + e^{-\beta(A_n - A_{min})}}, \\ s.t. \; & t_1^{S_1} = t_1^A, n \in N, \\ & t_n^{O_3} \leqslant t_n^A + L_n, \\ & 0 \leqslant E_n \leqslant M, \\ & 0 \leqslant P_n \leqslant M_{E_n}. \end{aligned} \tag{6}$$

where the initial constraint initializes the first task, while the subsequent constraint imposes a time limitation on task inference, the final set of constraints delineates the feasible ranges for E_n and P_n.

E. Problem Analysis

This optimization problem aims to efficiently address the challenge posed by the subsequent task stream, involving the trade-off between the completed inference task and task latency. This is achieved by implementing efficient partitioning and cooperative inference techniques for branch DNN models throughout the inference process. In this scenario, the total number of combinations of EE and MP points is denoted as H $(h \in H)$.

When selecting the h^{th} combination of these points, we represent the overall inference time and task gain for the m^{th} task as P_h^m and G_h^m, respectively. We thus reformulate the problem as a group knapsack problem involving H items, where each item must be efficiently accommodated within the knapsack. The h^{th} item's volume and value within the m^{th} group are represented by P_h^m and G_h^m, respectively. The overall capacity of the backpack is determined by the sum of t_M^A and L_M. Consequently, the formulated problem seeks to maximize the cumulative value of items that can be accommodated in the backpack. As a result, the problem addressed in this work is classified as *NP-hard*.

IV. Solution

A. DP-Based Cooperative Inference Algorithm

Aiming to achieve the maximum task gain by identifying the optimal EE and MP points for the tasks, this section introduces a task gain-aware algorithm based on DP. This algorithm consists of two main parts: task gain-aware decision and optimal point selection.

As shown in Algorithm 1, the algorithm begins by calculating the maximum gain for each task at each time slot and subsequently determines the overall maximum gain. In this scenario, we can anticipate task streaming information, including input data, arrival time, time limitation, and total task number. Consequently, we redefine the maximum task gain $\mathcal{G}(i,j)$ considering the i^{th} task and the limitation of the j^{th} time slot:

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

where $S(D,t)$ represents the highest accuracy of EE points for data size D and latency constraint t. Algorithm 1 initializes an array $\mathcal{G}(0,t)$, $(1 \leq t \leq t_N^A + L_N)$. Upon the arrival of the i^{th} task, if the sum of the previous round's arrival time and the latency constraint is less than the current task's arrival time, we update $\mathcal{G}(i-1, t_i^A)$ to $\mathcal{G}(i-1, t_{i-1}^A + L_{i-1})$. Simultaneously, for each j^{th} time point, we recursively iterate from t_i^A to $t_i^A + L_i$, comparing the maximum task gain of the previous round $\mathcal{G}(i-1,j)$. This process enables the determination of the maximum $\mathcal{G}(i,j)$.

Following the decision-making process considering task gain, Algorithm 1 undertakes an exhaustive search of the EE and MP points to select optimal points simultaneously, meeting both accuracy and time constraints. Upon the arrival of task T_n, for each branch in the multi-exit model, we sequentially determine the optimal MP points and calculate the overall processing time. Subsequently, we select the MP point associated with the minimum total processing time as the appropriate choice for the current task and branch. Additionally, each branch in the multi-exit model corresponds to a task accuracy A_n, resulting in a series of A_n values for the current task and its corresponding branches. Among these, the maximum A_n is selected. Finally, the corresponding EE and MP points, denoted as E_n and P_n, are the optimal associations for the current task.

B. Learning-Based Cooperative Inference Algorithm

The proposed cooperative inference strategy based on DP addresses the problem of when task information can be predicted. However, real-world applications face certain task information, such as the decision-maker's inability to obtain task information fully. This challenge demands the use of an online algorithm. Although task stream information is challenging to obtain in advance, task arrival times follow a *Bernoulli* process, where task arrival rate p obeys the normal distribution of data size.

Therefore, the DRL algorithm can be applied to solve it using a Markov Decision Process (MDP). The MDP can be modeled into a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$, where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, and $\mathcal{R}$ are considered as a set of states, actions, transition function, and reward functions, respectively. In the MDP, based on the decision policy $\mathcal{P}(\theta, s)$, the agent will take action $a \in \mathcal{A}$ for each $s \in \mathcal{S}$. To maximize the agent's long-term reward, constantly adjusting the parameter θ is crucial.

Fig. 4 illustrates the workflow of the proposed online cooperative inference strategy for decision-making in the MEO satellite, which acts as the system controller. System information is the starting point for system input, including warning deadlines, data size, and task queues, and serves as the starting point for system input. Online algorithms utilize this information to determine the system's state and integrate neural networks and decision components for decision-making. The algorithm's output is subsequently transmitted to the environment, where model inference occurs, followed by further processing based on both data and algorithmic decisions. The workflow is continuous

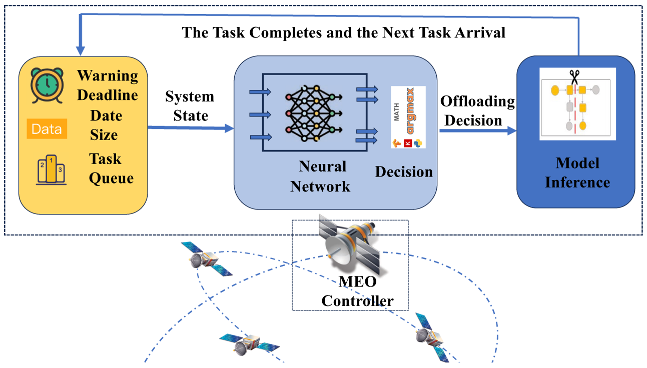


Fig. 4. The workflow for learning-based online decision-making strategy.

Algorithm 1: DP-Based Cooperative Inference Algorithm.

```
Input: $N$: the task number;
  $M$: the number of EE points;
  $\{M_k | k = 1,\ldots,M\}$: the number of layers of the EE points $k$;
  $D_n$: $T_n$'s input data size;
  $L_n$: $T_n$'s time deadline;
  $t_n^A$: $T_n$'s arriving time;
  $L_{latency}^{ISL}$: the inter-satellite transmission rate;
  $f_{i,j}^d$: the predictive model in the work;
Output: Optimal $E_n$, $P_n$
 1: 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:       $\mathcal{G}(i-1, t_i^A) = \mathcal{G}(i-1, L_{i-1} + t_{i-1}^A)$;
   // Update the recursion task gain value.
 5:   end if
 6:   for $j = t_i^A$ to $L_i + t_i^A$ do
 7:     Update $\mathcal{G}(i,j)$ according to (7);
 8:   end for
 9: end for
10: for $k = M$ to 1 do
11:   $E_n = k$;
12:   $C_n = \min_{P_n=1,\ldots,M_k} \{t_n^{O_3} - t_n^A\}$;
   // Divide the model EE points in sequence and select the smallest computing time.
13:   if $C_n \leq L_n$ and $A_n \geq A_n^*$ then
14:     record $E_n, P_n, A_n$;
15:   end if
16: end for
17: return $E_n, P_n$ when $A_n$ is maximum;
```

and iterative, with the system promptly addressing the next task upon completion.

Before making task decisions with the online cooperative inference strategy, we must first define MDP elements. This involves determining whether the task can be completed by considering its arrival time and total processing time, which includes satellite processing time and task waiting time. The state information should consist of data size, latency constraints, and task waiting time, which are essential for determining processing time. As mentioned in Section III, three factors influence the task waiting time for each task T_n: (i) the time when the LEO satellite starts to transmit T_{n-1}, i.e., $t^{O_1}_{n-1}$, (ii) the MEO satellite starts to transmit T_{n-1}, i.e., $t^{O_2}_{n-1}$ and (iii) the time when the task stream of T_{n-1} finishes, i.e., $t^{O_3}_{n-1}$. Consequently, the state is defined as shown in (8).

$$\begin{aligned} s_n = (D_n, L_n - max\{t^{O_1}_{n-1} - t^A_n, 0\}, \\ max\{t^{O_2}_{n-1} - t^A_n, 0\}, max\{t^{O_3}_{n-1} - t^A_n, 0\}), \end{aligned} \quad (8)$$

where D_n is the data size of the task, and $L_n - max\{t^{O_1}_{n-1} - t^A_n, 0\}$ denotes the remaining latency limitation for T_n after subtracting the waiting time for a previous task on the satellites. $max\{t^{O_2}_{n-1} - t^A_n, 0\}$ and $max\{t^{O_3}_{n-1} - t^A_n, 0\})$ represents LEO and MEO satellite overlap processing time for adjacent tasks, respectively. Next, the EE and MP points are selected from the model according to the system's performance, denoted as $a = (E_n, P_n)$. Simultaneously, we define the reward as G_n to maximize the system's gain, which can be increased by increasing the reward.

As a DRL algorithm capable of outputting discrete actions for decision-making, Double-Deep-Q-Network (DDQN) [46] has demonstrated superior performance in various scenarios and is utilized in the proposed learning-based algorithm. The highly complex and dynamic environment of satellite inference services – combined with multitasking services from LEO and MEO satellites – presents significant challenges. Therefore, we aim to improve the performance and adaptability of the DRL algorithm in such a space environment. To this end, we leverage experience from similar environments to initialize this algorithm. When training our learning-based online algorithms, the trained model parameters are associated with environmental parameters, such as task arrival rate, communications bandwidth, etc. In a new environment, we search for model parameters trained in an environment similar to the current one. These parameters are then loaded into the neural network for initialization, maximizing the training efficiency of the model. The environmental parameters form a multidimensional European space, where we use the nearest neighbor algorithm to search and identify the trained model parameters most similar to the current environment.

Furthermore, the balanced experience replay strategy is proposed to effectively accelerate DDQN convergence by integrating prioritized sampling with random sampling, thereby mitigating the risk of overfitting to noisy data. During the initial stages of training, the prioritized experience replay strategy focuses on samples with significant temporal difference errors. These samples, which often correspond to critical state-action pairs, significantly impact Q-value updates, thereby expediting the learning process. As training progresses, the model stabilizes. This adjustment prevents overfitting to noisy data in the experience pool, encourages better state space exploration, and ultimately enhances the model's generalization capability. The transition from prioritized to random sampling is governed by a predefined scheduling strategy, which adjusts the weights of the two sampling methods based on the number of training episodes.

Algorithm 2 describes the details of an online learning-based algorithm. First, we initialize the weight parameters of the target action-value function Q' and action-value function Q, with θ' and θ, respectively. The Q function is approximated using a DNN model, consisting of an input layer that takes the current state s_t as input, multiple hidden layers employing structures such as convolution and fully connected layers to extract and learn features, and an output layer that produces the final value for each possible action a_t. The Q' function follows a similar architecture. We utilize a probability ε-greedy strategy to select an action a for the state s_t, interacting with the environment to receive a reward r_t and the next state s_{t+1}. The tuple (s_t, a_t, r_t, s_{t+1}) is stored as a data sample in the replay buffer, using prioritized experience replay based on a binary tree search [17]. When updating the weight parameters of the action-value function Q, we input s_{t+1} into Q to obtain the Q-values for different actions. Then, we choose the action a' corresponding to the maximum Q-value. Based on s_{t+1} and a', we obtain the Q-value $Q'(s_{t+1}, a')$ from the target action-value function Q'. The actual value y_j is calculated according to (9) in the following.

$$y_j = \begin{cases} r_j, & \text{when episode} = \text{e}, \\ r_j + \gamma Q'(s_{j+1}, & argmax_a Q(s_{j+1},\, a;\, \theta),\, \theta'), \text{otherwise}. \end{cases} \quad (9)$$

Next, we calculate the loss based on the difference between the actual and estimated value $Q(s_t, a)$ and perform backpropagation to update the weight parameters. Finally, we synchronize θ' with θ after every N^- steps.

C. Complexity Analysis of Proposed Algorithms

The time complexity analysis of the proposed algorithm is discussed in this section. For the DP-based cooperative inference algorithm, the computational complexity of the algorithm can be simplified and expressed as $O(Nt^2)$, where t is the average delay constraint of tasks. Therefore, the proposed strategy's decision-making performance depends on the task system's average delay constraint and the total number of tasks.

To analyze the computational complexity of the online learning-based algorithm, we examine two modules: training and decision-making. Let P denote the maximum number of episodes during the training process, and Z represent the number of decision-making processes for each episode. According to [47], the complexity of the training stage is $O(PZ(\epsilon_1 + \epsilon_2))$, when ϵ_1 is the complexity of parameter updates and gradient decent and ϵ_2 is the complexity of action selection. During training, the algorithm makes inference decisions due to implementing reinforcement learning. While the agents execute, they upload

Algorithm 2: Online Learning-Based Cooperative Inference Algorithm.

```
Input: $D$: empty replay buffer;
  $N_r$: capacity of replay memory;
  $e$: the number of episodes;
  $p$: the number of steps;
  $N_b$: training batch size;
  $N^-$ steps: target network replacement freq;
Output: target weight parameters $\theta'$;
 1: Initialize action-value function $Q$ with random weights $\theta$;
 2: Initialize target action-value function $Q'$ with weights $\theta' = \theta$ with empirical knowledge from similar environment;
 3: Initialize the capacity of replay buffer $D$;
 4: for episode $= 1$ to $e$ do
 5:   Initialize $s_1$;
 6:   for $t = 1$ to $p$ do
 7:     With probability $\varepsilon$ select a random action $a_t$, Otherwise select $a_t = argmax_a Q(s_t, a; \theta)$;
 8:     Executing action $a_t$, observe reward $r_t$ and next state $s_{t+1}$;
 9:     Store tuple $(s_t, a_t, r_t, s_{t+1})$ in replay buffer $D$ with balanced experience replay strategy;
10:     Set $s_t \leftarrow s_{t+1}$;
11:     Sample a minibatch of tuples $(s_j, a_j, r_j, s_{j+1})$ in replay buffer $D$ with the same strategy;
12:     Executing a gradient descent step with loss $(y_j - Q(s_j, a_j; \theta))^2$ according to (9);
13:     Replace $Q' \leftarrow Q$ after every $N^-$ steps;
14:     Reset $\theta' \leftarrow \theta$;
15:   end for
16: end for
```

historical information, such as the states, actions, and rewards. Once training is complete, the agent obtains the weights of the actor network and imports them into the actor network for computational inference decision. According to [48], the complexity for decision-making is $O(PZ\epsilon_2)$, as DDQN-based algorithms make decisions only with actor network after training.

V. Evaluation

In this section, we initially describe the experimental setup and subsequently conduct a comprehensive analysis of the experimental results.

A. Evaluation Setup

Simulation Setting: We utilize the StarLink satellite constellation [49] and MEO satellite as benchmarks to simulate inter-satellite connectivity across different orbits. In the task-slot workflow model, each MEO satellite coordinates with three LEO satellites by default. The simulation is implemented in Python, utilizing the Python package Networkx [50] to establish the ISL network. The Ka-band, known for its high data transmission rates and compact device size, is increasingly utilized in

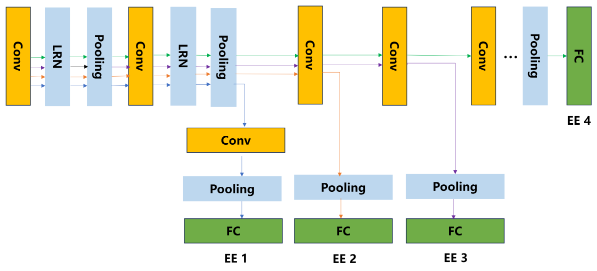


Fig. 5. The AlexNet model comprises 4 EE points in general. However, when task failure is regarded as a distinct scenario from the outset, the AlexNet model encompasses 5 EE points.

inter-satellite communications. For instance, NASA's ISARA and MarCO missions employed the Ka-band to achieve high-bandwidth communications [51], demonstrating its potential for future missions requiring substantial data throughput. Consequently, this work also uses the Ka-band for communications between MEO and LEO satellites. Additionally, we leverage LEO satellites' diverse computing resource capabilities by adjusting parameters such as the number and frequency of CPUs.

Moreover, we assume that the input data size is directly proportional to the maximum tolerable latency for each task, simplifying the discussion. Specifically, we express this relationship as $L_n = k \times D_n$, where $k \in \mathbb{N}$. The default value assigned to the constant k is 5. The results presented in each figure are based on 100 consecutive tasks processed by the proposed algorithm and baseline algorithms. If not specified, the setting above will be the default in our simulation. The simulation employs the CIFAR-10 dataset, classifying each image into one of ten classes. This dataset serves multiple purposes, including model training, simulating tasks, and evaluating the proposed algorithm. The evaluated DNN-based inference application focuses on general-purpose classification, utilizing an 8-layer AlexNet model with 4 EE points. The overview of the considered AlexNet model is depicted in Fig. 5. In cases where task failure is considered, the model includes 5 EE points at the beginning. Consequently, after training, the corresponding inference accuracy for the 5 EE points is [0, 0.527, 0.623, 0.697, 0.743].

Proposed Algorithms: The task generation is modeled by a *Bernoulli distribution* with a rate of $p = 0.1$, ensuring that at most one task is generated during each time slot. The input consists of images to be inferred in the task system, where the number of images follows a normal distribution within the range [1, 10]. Each time slot is fixed to be 3 seconds. The hyperparameters α and β are assigned values of 0.1 and 16, respectively. The proposed online learning-based algorithm uses two layers of 20 neurons per layer, and the *tanh* activation function is applied. Training is conducted with a discount factor of 0.9, and the replay memory capacity is set to 10^6. Table II shows the detailed setting of algorithm parameters.

Baseline Algorithms for comparison: We assess the performance of our proposed algorithm by comparing it with the following algorithms:

Greedy: The algorithm makes cooperative inference decisions based on the current information of the task system without considering the interaction between adjacent tasks.

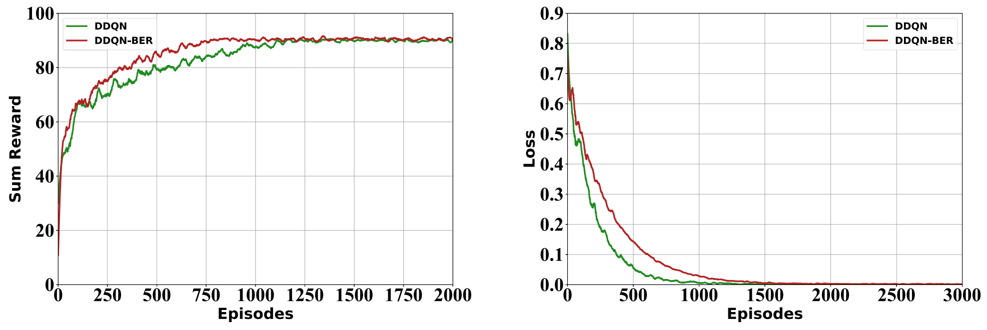


Fig. 6. The values of reward and loss during the training of our algorithm and basic DDQN algorithm.

TABLE II
SETTINGS OF ALGORITHM PARAMETERS

Parameters	Values
The proportion k	5
The discount factor for model training	0.9
The capacity of replay memory	10^6
The arrival rate of task	0.1
Hyperparameter α	0.1
Hyperparameter β	16
The number of consecutive tasks	100

Random: The algorithm randomly makes cooperative inference decisions without incorporating any system information regarding the task stream.

LEO-all: The algorithm processes all computational tasks in LEO satellites and selects the optimal EE point for each task without considering MP points.

MEO-all: Similar to LEO-all, the algorithm sends all computational tasks to MEO satellites for processing.

DP: The algorithm makes the cooperative inference decisions by selecting the optimal strategies, which also set the theoretically upper-performance limit.

Performance Metrics: We consider three key metrics to evaluate the proposed algorithm:

Overall Task Gain: The metric, defined by (5), donates the sum of task gain for all consecutive tasks in the system.

Task Completion Rate: From the user's perspective, this metric illustrates the effectiveness of decision-making in task completion. It is determined as the ratio of the number of successfully completed tasks to the total number of tasks received.

Average Task Latency: This metric reflects the impact of decision-making on the average task latency. It is calculated by summing the differences between the completion time of completed tasks and their arrival time and then dividing this sum by the number of tasks completed.

B. Convergence Behavior

Fig. 6 illustrates the cumulative reward and loss of the online approach during training, highlighting the convergence behavior of the improved DDQN-based algorithm and basic DDQN algorithm. The experimental results demonstrate that our algorithm converges rapidly and performs better than the basic DDQN algorithm. This improvement is attributed to the balanced experience replay strategy, which allows our algorithm to learn action patterns through interactions with the environment more efficiently. While the basic DDQN algorithm initially makes decisions randomly, our approach accelerates learning and ultimately achieves stable cumulative reward and loss values.

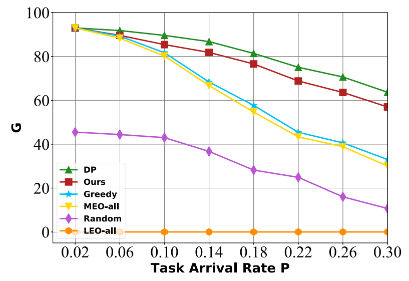
C. Impact of Task Arrival Rate

We investigate the effectiveness of each algorithm across varying task arrival rates, where task arrivals follow *Bernoulli* distribution with parameter p. Fig. 7(a) and (b) show that as the task arrival rate p increases, the over-task gain and the task completion rate of all algorithms continuously decrease. In Fig. 7(a), the proposed approach mitigated a 16.5% decline in the reward sum compared with the Greedy algorithm. Besides, Fig. 7(b) also exhibits that our algorithm mitigates the 25.3The gap between our algorithm and Greedy algorithm's task completion rate was small but gradually widened. That is because, as the task arrival rate p increases, representing a higher task occurrence, the interaction between adjacent tasks becomes increasingly vital. This results in a notable rise in the number of decision failures in the Greedy algorithm. Fig. 7(c) shows that the average task latency of our algorithm outperforms other algorithms except the DP-based algorithm. Additionally, the LEO-all algorithm exhibits the poorest performance, as the algorithm cannot complete most tasks when all tasks are assigned to LEO satellites.

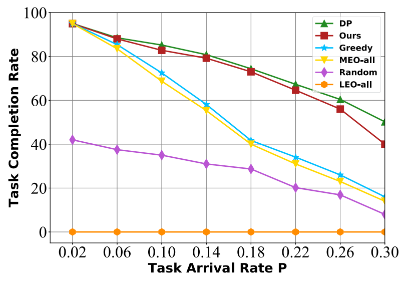
D. Impact of Different Task Numbers

Fig. 8 illustrates the performance of each algorithm across different task numbers, highlighting the operational scalability of the proposed algorithm. The task number has a negligible impact on our algorithms under the dynamic state of the task stream. Our algorithms outperform the other baselines in the overall task gain (i.e., cumulative reward), task completion rate, and average latency. The online learning-based algorithm ranks second, only behind the DP-based algorithm, which sets a theoretically upper bound by obtaining all task system information.

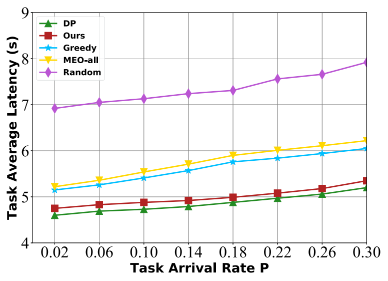
Specifically, as depicted in Fig. 8(a), the cumulative rewards of all algorithms continue to improve with the number of tasks, except for the LEO-all baseline algorithm. Compared with the



(a) The overall task gain G.

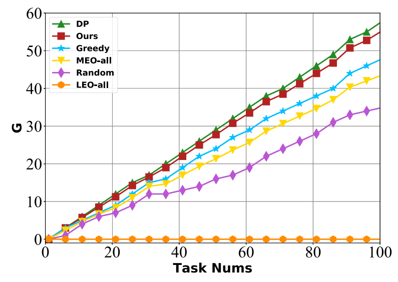


(b) The task completion rate.

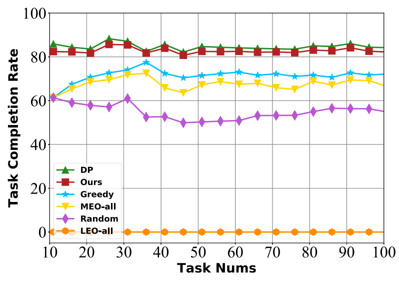


(c) The task average latency.

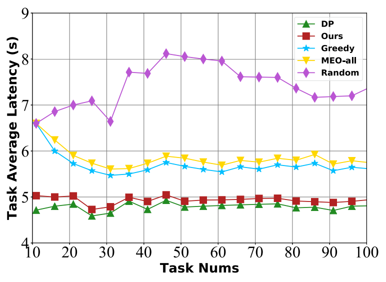
Fig. 7. The performance of all algorithms under various task arrival rates p.



(a) The overall task gain G.

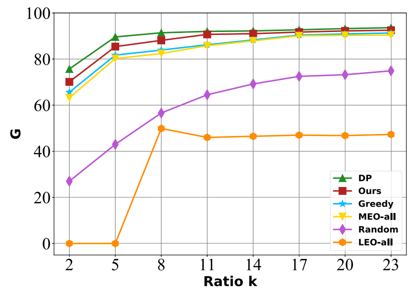


(b) The task completion rate.

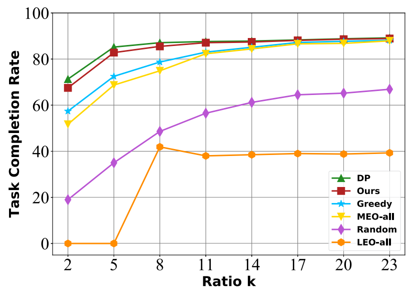


(c) The task average latency.

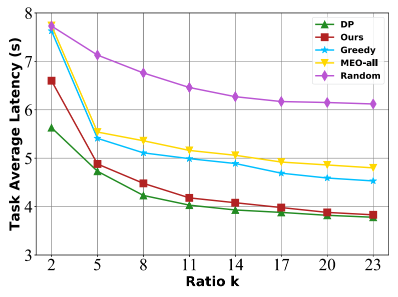
Fig. 8. The performance of all algorithms under various task numbers.



(a) Over task gain.



(b) The task completion rate.



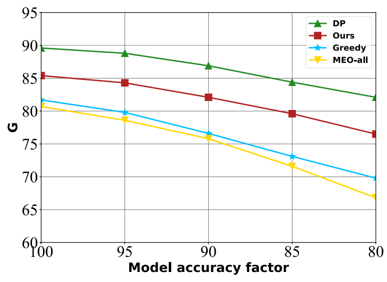
(c) The task average latency.

Fig. 9. The performance of all algorithms under various ratio k (i.e., various deadline constraint).

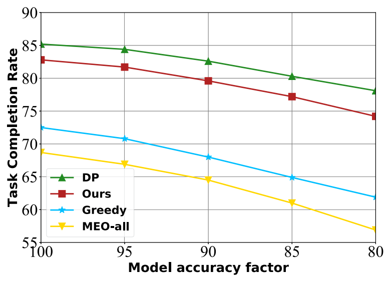
Greedy algorithm, Random algorithm, and MEO-all algorithm, the proposed method enhances the task gain by 47.4% , 151.6% , and 62.9% , respectively. Our online learning-based algorithm exhibits a significant improvement over the Greedy algorithm in cumulative reward despite the algorithm adhering to the optimal strategy based on the current task system. In Fig. 8(b), our algorithms show better performance and minimal fluctuations across different task numbers. Notably, the LEO-all algorithm exhibits the poorest performance in task completion rate. This is because when all tasks are executed in LEO satellites, many tasks experience prolonged waiting times, failing to meet specific performance requirements. Additionally, in Fig. 8(c), the learning-based approach performs similarly to the DP-based approach, remaining well below the maximum tolerance time for task completion under the default simulation setting. Our method consistently achieves the proper trade-off between task accuracy and completion time, adapting to the current task state.

E. Impact of Maximum Deadline

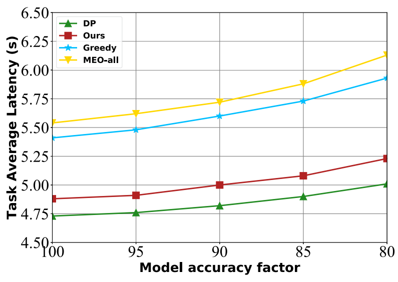
We investigate the effect of different deadlines on the performance of all algorithms based on the simulation assumption that the maximum time deadline for each task is k times the input data size. Fig. 9 illustrates the superior performance of our algorithms in terms of sum reward, task completion, and task average latency. For all algorithms, as the proportion k increases, both the cumulative reward and task completion consistently improve while the average task latency steadily decreases. Compared to the Greedy algorithm, our learning-based algorithm achieves an average of 3.6% enhancement in sum reward and an average 6.3% enhancement in the task completion rate. As the proportion k increases, the Greedy algorithm gradually approaches the performance of our algorithm. This curve implies that the interaction between adjacent tasks diminishes as the task latency constraint is relaxed. The LEO-all algorithm fails to complete



(a) Over task gain.



(b) The task completion rate.



(c) The task average latency.

Fig. 10. The performance of all algorithms under different model accuracy prediction.

any tasks when the proportion k is lower, and the improvement remains limited as the proportion k increases because of the constrained resources of LEO satellites. Fig. 9(c) depicts that the average latency of our algorithm is lower than that of the Greedy algorithm. However, the MEO-all algorithm performs similarly to the Greedy algorithm and significantly outperforms the Random algorithm. This highlights the advantage of collaboration among space satellites in improving overall performance.

F. Impact of Inference Model

As an extension to the above experiment, we further investigate the impact of the inference model across various algorithms. To this end, we conduct experiments to evaluate system performance under different model accuracy conditions [52], [53]. We precisely adjust the original accuracy of the model branches on LEO and MEO satellites by introducing a modeling accuracy factor, which modifies the model's accuracy. As illustrated in Fig. 10, the results clearly show that a decrease in the factor leads to a reduction in overall task gain and completion rate and an increase in average task latency across all algorithms. Despite these changes, our proposed method demonstrates advantages when model accuracy is affected. On the other hand, the experimental results indicate that, with a smaller factor, each algorithm requires lower accuracy and additional time for each algorithm to address tasks due to inaccuracies in the model, finally reducing overall system performance. This find also highlights the importance of selecting higher-accuracy models for improved system performance.

G. Impact of Run Time Performances

As illustrated in Fig. 11, we analyze the runtime of the proposed algorithm across varying time slots. Our findings indicate a significant decrease in run time as the duration of time slot increases, which is consistent with our previous analysis. Therefore, exploring the most optimal computational allocation for each task could be a promising solution to maintain runtime within a reasonable range, warranting further investigation.

H. Discussion

To address the requirements of ubiquitous coverage and high-resolution satellite imaging, this work leverages collaborative inference across different orbital satellites, utilizing emerging

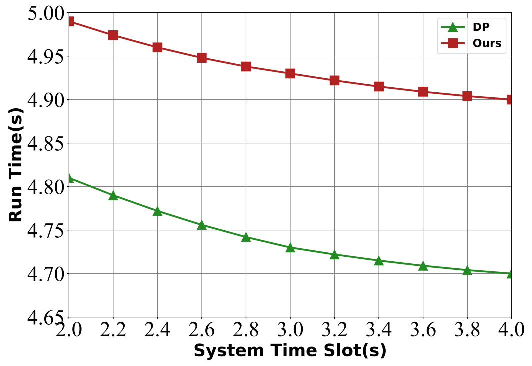


Fig. 11. Run time of proposed algorithms across different time slots.

computational capabilities. While computational bottlenecks in space systems present significant challenges, ongoing advancements in onboard computing have made these objectives increasingly attainable. After two decades or more of operation, the processing capabilities of onboard processors may approach twice that of the current state-of-the-art technology. Hence, this work can be easily extended to support advancements in future satellite hardware.

When initiated from scratch, the online learning-based cooperative inference algorithm involves exploration steps, which may result in performance fluctuations during inference. In practical scenarios, extensive exploration is considered unnecessary, as a multitude of samples representing typical cases can be efficiently pre-populated by satellites. Additionally, using a pre-trained neural network significantly decreases the overall training time. The meta-learning approach [54], [55] highlights the capability to construct a generalized neural network adaptable to diverse environments with minimal input data, notably streamlining the retraining process. After accumulating sufficient samples from satellites, the ground station employs meta-learning to generate a base Q-network transmitted to the satellite. This neural network seamlessly integrates into the satellite, providing a robust foundation for swift adaptation. Thus, learning time is substantially reduced compared to initiating the process from scratch.

VI. Conclusion and Future Work

This paper introduced an in-orbit task gain-aware computing cooperative inference problem in a satellite edge computing network via multi-exit models. This achieved an adaptive DNN-based inference strategy for multiple-task streaming services. The formulated problem aimed to make a trade-off between the completed inference task and DNN inference accuracy. This problem is solved by a dynamic programming-based algorithm when system information is known. We also proposed an online algorithm for the real-world case with unknown task arrivals and uncertain network states. We conducted extensive simulations, showing that the proposed algorithms outperform baseline algorithms regarding vital performance metrics.

In-orbit computing has an exciting future with many open research questions. In future work, we intend to validate our experiments on the onboard satellite constellation testbed and explore the potential for optimizing DNN inference. Moreover, computational and communication challenges will remain significant for satellite-based computer systems. These constraints imposed by energy limitations and satellite-ground bandwidth dynamics also present unique limitations on satellite computing. We will also investigate the energy consumption of satellites on computing inference performance when satellite resources are available and propose energy-aware computing inference algorithms.

Acknowledgment

A preliminary version of this paper appears as a conference paper in proceedings of the 59th IEEE International Conference on Communications [19].

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] UCS satellite database, 2023. [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,” *Proc. IEEE*, vol. 111, no. 3, pp. 257–276, Mar. 2023.
[4] Y. Chen, Q. Zhang, Y. Zhang, X. Ma, and A. Zhou, “Energy and time-aware inference offloading for DNN-based applications in LEO satellites,” in *Proc. 31st Int. Conf. Netw. Protoc.*, 2023, pp. 1–6.
[5] M. Tian, G. Ma, P. Huang, B. Cheng, and W. Li, “Optimizing satellite ground station facilities scheduling for RSGS: A novel model and algorithm,” *Int. J. Digit. Earth*, vol. 16, no. 1, pp. 3949–3972, 2023.
[6] C. Chi, Y. Zhang, Q. Sun, and S. Wang, “SABM: Adaptive backup mechanism for satellite services,” in *Proc. 1st ACM MobiCom Workshop Satell. Netw. Comput.*, 2023, pp. 1–6.
[7] J. Chen et al., “A remote sensing data transmission strategy based on the combination of satellite-ground link and GEO relay under dynamic topology,” *Future Gener. Comput. Syst.*, vol. 145, pp. 337–353, 2023.
[8] S. Wang, Q. Zhang, R. Xing, F. Qi, and M. Xu, “The first verification test of space-ground collaborative intelligence via cloud-native satellites,” *China Commun.*, vol. 21, no. 4, pp. 207–218, 2024.
[9] P. K. Donta, B. Sedlak, V. Casamayor Pujol, and S. Dustdar, “Governance and sustainability of distributed continuum systems: A Big Data approach,” *J. Big Data*, vol. 10, no. 1, pp. 1–31, 2023.
[10] S. Dustdar and I. Murturi, “Towards IoT processes on the edge,” in *Next-Gen Digital Services. A Retrospective and Roadmap for Service Computing of the Future: Essays Dedicated to Michael Papazoglou on the Occasion of His 65th Birthday and His Retirement*, Berlin, Germany: Springer, 2021, pp. 167–178.
[11] Q. Zhang et al., “Resource-efficient in-orbit detection of earth objects,” in *Proc. 43rd Int. Conf. Commun.*, 2024, pp. 1–9.
[12] B. Denby and B. Lucia, “Orbital edge computing: Nanosatellite constellations as a new class of computer system,” in *Proc. 35th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2020, pp. 939–954.
[13] B. Denby, K. Chintalapudi, R. Chandra, B. Lucia, and S. Noghabi, “Kodan: Addressing the computational bottleneck in space,” in *Proc. 28th ACM Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2023, pp. 392–403.
[14] Planet to support NASA relay networks for telesat, ses. satellite today, 2022. [Online]. Available: https://www.satellitetoday.com/government-military/2022/08/23/planet-to-support-nasa-relay-networks-for-telesat-ses/
[15] B. Tao, O. Chabra, I. Janveja, I. Gupta, and D. Vasisht, “Known knowns and unknowns: Near-real time earth observation via query bifurcation in serval,” in *Proc. 21st USENIX Symp. Netw. Syst. Des. Implementation*, 2024, pp. 1–16.
[16] Q. Zhang et al., “A comprehensive deep learning library benchmark and optimal library selection,” *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 5069–5082, May 2024.
[17] Z. Liu et al., “Hastening stream offloading of inference via multi-exit DNNs in mobile edge computing,” *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 535–548, Jan. 2024.
[18] Z. Lai, Q. Wu, H. Li, M. Lv, and J. Wu, “OrbitCast: Exploiting mega-constellations for low-latency earth observation,” in *Proc. 29th Int. Conf. Netw. Protoc.*, 2021, pp. 1–12.
[19] J. Guan, Q. Zhang, I. Murturi, P. K. Donta, S. Dustdar, and S. Wang, “Collaborative inference in DNN-based satellite systems with dynamic task streams,” in *Proc. 59th Int. Conf. Commun.*, 2024, pp. 3803–3808.
[20] D. Mengmeng, N. Noboru, I. Atsushi, and S. Yukinori, “Multi-temporal monitoring of wheat growth by using images from satellite and unmanned aerial vehicle,” *Int. J. Agricultural Biol. Eng.*, vol. 10, no. 5, pp. 1–13, 2017.
[21] J. Verbesselt, R. Hyndman, G. Newnham, and D. Culvenor, “Detecting trend and seasonal changes in satellite image time series,” *Remote Sens. Environ.*, vol. 114, no. 1, pp. 106–115, 2010.
[22] O. Laux, D. Poncet, R. Mager, and K. Schoenherr, “Status of the European data relay satellite system,” in *Proc. Int. Conf. Space Opt. Syst. Appl.*, 2012, pp. 9–12.
[23] G. K. Kurt et al., “A vision and framework for the high altitude platform station (HAPS) networks of the future,” *IEEE Commun. Surv. Tuts.*, vol. 23, no. 2, pp. 729–779, Second Quarter 2021.
[24] J. Pelton, “Overview of high altitude platform systems (HAPS) and their future relation to satellite networks,” in *Proc. Int. Commun. Satell. Syst. Conf. Exhibit.*, 2000, pp. 1130–1130.
[25] C. E. Kement et al., “Sustaining dynamic traffic in dense urban areas with high altitude platform stations (HAPS),” *IEEE Commun. Mag.*, vol. 61, no. 7, pp. 150–156, Jul. 2023.
[26] D. Bhattacherjee and A. Singla, “Network topology design at 27,000 km/hour,” in *Proc. 15th Int. Conf. Emerg. Netw. Exp. Technol.*, 2019, pp. 341–354.
[27] A. Chen et al., “Opara: Exploiting operator parallelism for expediting DNN inference on GPUs,” *IEEE Trans. Comput.*, vol. 74, no. 1, pp. 325–333, Jan. 2025.
[28] J. Wu, L. Wang, Q. Jin, and F. Liu, “Graft: Efficient inference serving for hybrid deep learning with SLO guarantees via DNN re-alignment,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 35, no. 2, pp. 280–296, Feb. 2024.
[29] F. Xu et al., “iGniter: Interference-aware GPU resource provisioning for predictable DNN inference in the cloud,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 3, pp. 812–827, Mar. 2023.
[30] Z. Sun, X. Guan, J. Wang, F. Liu, and H. Cui, “New problems in distributed inference for DNN models on robotic IoT,” in *Proc. 2024 Workshop Adv. Tools Program. Lang. PLatforms Implementing Eval. Algorithms Distrib. Syst.*, 2024, pp. 1–9.
[31] Y. Zhang, W. Liang, Z. Xu, W. Xu, and M. Chen, “AoI-aware inference services in edge computing via digital twin network slicing,” *IEEE Trans. Serv. Comput.*, vol. 17, no. 6, pp. 3154–3170, Nov./Dec. 2024.
[32] Z. Liu, M. Tian, M. Dong, X. Wang, C. Qiu, and C. Zhang, “MoEI: Mobility-aware edge inference based on model partition and service migration,” *IEEE Trans. Mobile Comput.*, vol. 23, no. 10, pp. 9437–9450, Oct. 2024.
[33] K. Liu, C. Liu, G. Yan, V. C. Lee, and J. Cao, “Accelerating DNN inference with reliability guarantee in vehicular edge computing,” *IEEE/ACM Trans. Netw.*, vol. 31, no. 6, pp. 3238–3253, Dec. 2023.
[34] 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 Things J.*, vol. 7, no. 10, pp. 9241–9254, Oct. 2020.
[35] W. Miao, Z. Zeng, L. Wei, S. Li, C. Jiang, and Z. Zhang, “Adaptive DNN partition in edge computing environments,” in *Proc. 26th Int. Conf. Parallel Distrib. Syst.*, 2020, pp. 685–690.
[36] Y. Kang et al., “Neurosurgeon: Collaborative intelligence between the cloud and mobile edge,” *ACM SIGARCH Comput. Archit. News*, vol. 45, no. 1, pp. 615–629, 2017.

[37] A. E. Eshratifar, M. S. Abrishami, and M. Pedram, “JointDNN: An efficient training and inference engine for intelligent mobile cloud computing services,” *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 565–576, Feb. 2021.
[38] S. Teerapittayanon, B. McDanel, and H.-T. Kung, “Distributed deep neural networks over the cloud, the edge and end devices,” in *Proc. 37th Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 328–339.
[39] E. Li, L. Zeng, Z. Zhou, and X. Chen, “Edge AI: On-demand accelerating deep neural network inference via edge computing,” *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 447–457, Jan. 2020.
[40] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, and N. D. Lane, “SPINN: Synergistic progressive inference of neural networks over device and cloud,” in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, 2020, pp. 1–15.
[41] D. Stamoulis et al., “Designing adaptive neural networks for energy-constrained image classification,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2018, pp. 1–8.
[42] S. Teerapittayanon, B. McDanel, and H.-T. Kung, “BranchyNet: Fast inference via early exiting from deep neural networks,” in *Proc. 23rd Int. Conf. Pattern Recognit.*, 2016, pp. 2464–2469.
[43] 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 Trans. Sensor Netw.*, vol. 17, no. 3, pp. 1–28, 2021.
[44] J. Song, Z. Liu, X. Wang, C. Qiu, and X. Chen, “Adaptive and collaborative edge inference in task stream with latency constraint,” in *Proc. Int. Conf. Commun.*, 2021, pp. 1–6.
[45] B. Fang, X. Zeng, and M. Zhang, “NestDNN: Resource-aware multi-tenant on-device deep learning for continuous mobile vision,” in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw.*, 2018, pp. 115–127.
[46] F. Zhao, Q. Wang, and L. Wang, “An inverse reinforcement learning framework with the Q-learning mechanism for the metaheuristic algorithm,” *Knowl.-Based Syst.*, vol. 265, 2023, Art. no. 110368.
[47] Z. Ning et al., “Deep reinforcement learning for intelligent internet of vehicles: An energy-efficient computational offloading scheme,” *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 4, pp. 1060–1072, Dec. 2019.
[48] Y. Sun, S. Zhou, and J. Xu, “EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks,” *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017.
[49] V. L. Foreman, A. Siddiqi, and O. De Weck, “Large satellite constellation orbital debris impacts: Case studies of OneWeb and SpaceX proposals,” in *Proc. AIAA SPACE Astronaut. Forum Expo.*, 2017, pp. 5200–5200.
[50] Networkx, 2023. [Online]. Available: https://networkx.org/documentation/stable/tutorial.html
[51] Nasa task, 2024. [Online]. Available: https://www.nasa.gov/smallsat-institute/sst-soa/soa-communications/
[52] B. Jacob et al., “Quantization and training of neural networks for efficient integer-arithmetic-only inference,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2704–2713.
[53] G. Fontanesi et al., “Artificial intelligence for satellite communication: A survey,” *IEEE Commun. Surveys Tut.*, early access, Feb. 2025, doi: 10.1109/COMST.2025.3534617.
[54] T. Gong, Y. Kim, J. Shin, and S.-J. Lee, “MetaSense: Few-shot adaptation to untrained conditions in deep mobile sensing,” in *Proc. 17th Conf. Embedded Netw. Sensor Syst.*, 2019, pp. 110–123.
[55] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

Qiyang Zhang received the PhD degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2024. He is currently a research assistant with the School of Computer Science, Peking University, Beijing, China. He is also a visiting student with the Distributed Systems Group, TU Wien from December 2022 to December 2023. He has published papers in WWW, INFOCOM, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Services Computing*, *IEEE Network*, etc. His research interests include satellite computing, edge intelligence.

Shangguang Wang (Senior Member, IEEE) is a professor with the School of Computer Science, Beijing University of Posts and Telecommunications, China. His research interests include service computing, mobile edge computing, cloud computing, and satellite computing. He is currently chair of the IEEE Technical Community on Services Computing (TCSVC) and vice chair of the IEEE Technical Community on Cloud Computing. He also served as general chair or program chair of more than 10 IEEE conferences and advisor/associate editor of journals such as the *Journal of Cloud Computing*, *Journal of Software: Practice and Experience*, *International Journal of Web and Grid Services*, *China Communications*, and so on. He is a fellow of the IET. For further information, please visit http://www.sguangwang.com.

Jinglong Guan received the BS degree from the Faculty of Information Technology, Beijing University of Technology, in 2018. He is currently working toward the master’s degree in computer science with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. His research interests include satellite edge computing and edge intelligence.

Praveen Kumar Donta (Senior Member, IEEE) received the BTech and MTech degrees with distinction from JNTU Anantapur, India, in 2012 and 2014, respectively, and the PhD degree from the Department of Computer Science and Engineering, IIT (ISM), Dhanbad, India, in June 2021. He is currently associate professor with Stockholm University. Before, he was a postdoctoral researcher with the Distributed Systems Group, TU Wien, Austria. His current research is distributed continuum systems, learning in IoT, and fog/edge computing.

Xiao Ma received the PhD degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2018. She is currently a lecturer with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing. From October 2016 to April 2017, she visited the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. Her research interests include mobile-cloud computing and mobile-edge computing.

RangaRao Venkatesha Prasad (Senior Member, IEEE) is an associate professor with ENSys Group of TU Delft. His research interest is in the areas of Space-IoT, Tactile Internet, and 60 GHz mmWave networks. He has supervised 20 PhD students and more than 65 MSc students. He has more than 300 publications in peer-reviewed international journals, conferences, standards, and book chapters. He has served on the editorial board of *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Mobile Computing*, *IEEE Communications Surveys and Tutorials*, *IEEE Transactions on Green Communications and Networking*, and many other IEEE Transactions. He was the vice-chair of the IEEE Tactile Internet standardization workgroup and is now a mentor. For more information, please refer to http://homepage.tudelft.nl/w5p50.

Schahram Dustdar (Fellow, IEEE) is full professor of computer science heading the research division of distributed systems with the TU Wien, Austria. He was the founding co-editor-in-chief of the new *ACM Transactions on Internet of Things (ACM IoT)* and editor-in-chief of *Computing (Springer)*. He is an associate editor of *IEEE Transactions on Services Computing*, *IEEE Transactions on Cloud Computing*, *ACM Transactions on the Web*, and *ACM Transactions on Internet Technology*, as well as on the editorial board of *IEEE Internet Computing* and *IEEE Computer*. In 2021, he was elected president of the Asia-Pacific Artificial Intelligence Association (AAIA).

Xuanzhe Liu (Senior Member, IEEE) is a full professor with the School of Computer Science, Peking University, Beijing, China. His research interests mainly fall in service-based software engineering and systems. Most of his recent efforts have been published at prestigious conferences, including WWW, ICSE, FSE, SOSP, SIGCOMM, NSDI, MobiCom, MobiSys, and in journals including the *ACM Transactions on Software Engineering and Methodology*, *ACM Transactions on Information Systems*, *IEEE Transactions on Software Engineering*, *IEEE Transactions on Mobile Computing*, and *IEEE Transactions on Services Computing*. He is a distinguished member of the ACM and the CCF. Web page: http://www.liuxuanzhe.com/.