# Digital Twin-Driven Semantic Offloading for LEO-MEC-Enabled Remote IoT Networks

Ayalneh Bitew Wondmagegn, Dongwook Won, Donghyun Lee, Jaemin Kim, Juyoung Kim, Sungrae Cho

[a]*School of Computer Science and Engineering, Chung-Ang University, Seoul, 06974, Republic of Korea*

**Abstract**

Efficient task offloading to low-Earth orbit (LEO) satellite–assisted mobile edge computing (MEC) servers is vital for addressing the computational delay and energy constraints of remote Internet of Things (IoT) devices. However, inherent challenges in LEO satellite networks—including dynamic topology, intermittent connectivity, and bandwidth scarcity undermine the performance of conventional offloading schemes. Moreover, raw data transmission inflates latency and energy consumption, hindering real-time and resource-aware decision-making. To overcome these limitations, this paper presents a digital twin (DT)-driven, semantic-aware offloading framework tailored for LEO-integrated MEC systems. The proposed framework aims to minimize end-to-end DT synchronization delay while adhering to strict energy, semantic fidelity, and computational constraints. Semantic encoders extract task-relevant information to compress payloads, and a binary offloading strategy ensures each task is either fully executed locally or offloaded entirely. The problem is formulated as a joint optimization of offloading decisions, edge server selection, semantic compression ratio, and transmission power allocation. A proximal policy optimization (PPO)-based deep reinforcement learning algorithm is developed to solve the problem, leveraging real-time DT feedback for adaptive, context-aware control under dynamic network conditions. Simulation results demonstrate that the proposed framework reduces DT synchronization delay by up to 45% vs. local execution, by 26–35% vs. non-semantic offloading, and by 5–10% vs. DRL alternative at high load, with statistically significant gaps. Additionally, the system maintains energy and semantic fidelity requirements while scaling efficiently with data volume and device density. These findings offer a practical and scalable solution for enabling reliable, low-latency DT services in 6G satellite—ground integrated IoT networks.

*Keywords:* Digital twin (DT), Semantic communication (SemCom), Offloading, Low-Earth orbit mobile edge computing (LEO-MEC)

## 1. Introduction

### 1.1. Background

The widespread deployment of IoT devices has enabled real-time sensing and intelligent services in various domains, including smart agriculture, environmental monitoring, and disaster response. However, operating in remote or infrastructure-limited areas remains challenging due to the limited computational, energy, and storage resources on IoT devices.

*Corresponding author: Sungrae Cho

*Email addresses:* `ayalneh@uclab.re.kr` (Ayalneh Bitew Wondmagegn), `dwwon@uclab.re.kr` (Dongwook Won), `dhlee@uclab.re.kr` (Donghyun Lee), `jmkim@uclab.re.kr` (Jaemin Kim), `jykim@uclab.re.kr` (Juyoung Kim), `srcho@cau.ac.kr` (Sungrae Cho)

Although mobile edge computing (MEC) offers a viable solution by enabling task offloading to nearby edge servers, terrestrial MEC infrastructure is often unavailable or unreliable in such regions due to geographical limitations or disruptions caused by natural disasters [1, 2].

Low-Earth orbit (LEO) MEC systems, where MEC servers are directly embedded in LEO satellites, have emerged as a promising architecture for enabling low-latency and location-independent task processing to address this problem [3]. This approach allows IoT devices to offload tasks directly to satellite-edge servers, making it suitable for time-critical applications, including autonomous navigation and real-time surveillance [4–6]. However, challenges remain, including dynamic satellite topologies, intermittent connectivity, and limited onboard resources. The fast movement of LEO satellites causes frequent handovers and short coverage windows, which complicates server selection and stable connectivity. Moreover, bandwidth limitations and high transmission energy costs pose significant barriers, especially for remote IoT devices. Propagation delays and the need for real-time decision-making increase the complexity of managing delay-sensitive tasks.

A critical bottleneck in LEO-MEC-enabled task offloading is the transmission of extensive raw sensory data over bandwidth-constrained satellite links with short coverage durations. Semantic communication (SemCom) has emerged as a promising solution to address this challenge [7, 8]. Unlike conventional communication paradigms focusing on delivering the entire set of raw data, SemCom uses machine learning-based encoders and decoders to extract and transmit only task-relevant semantic content. This method considerably reduces redundant information, enhancing spectral and energy efficiency. The benefits of SemCom are notable for high-dimensional data types, such as text, images, and video, where only the essential meaning must be preserved for effective task execution [9–11].

Integrating SemCom into LEO-MEC systems is advantageous in bandwidth-scarce, intermittently connected environments with high-frequency impairments, which are typical conditions in satellite communication [12, 13]. Semantic-aware offloading enhances spectral efficiency and shifts part of the communication burden to the processing unit by transmitting compressed but semantically meaningful representations, balancing the trade-off between communication and computation. Moreover, SemCom is robust to radio link instability; even partially received semantic data can yield beneficial results, ensuring greater reliability in variable channel conditions [14].

In this context, the fusion of SemCom and LEO-MEC forms a robust foundation for next-generation offloading systems. With lightweight deep neural network-based encoders, semantic-aware offloading allows task-critical feature extraction, context-aware prioritization, and meaning-preserving compression. This approach reduces transmission loads while preserving mission-critical content, reducing delays, and improving the offload speed and user experience in remote and resource-constrained environments.

Complementing the capabilities of SemCom, DT technology provides an intelligent and dynamic framework for real-time system awareness, prediction, and decision making [15]. Moreover, DTs enable fine-grained observation and predictive modeling of IoT devices, LEO satellites, and communication links by maintaining a continuously synchronized virtual replica of physical entities [16]. In LEO-MEC environments, DTs enable context-aware optimization of task offloading, resource scheduling, and network coordination. Using historical data and real-time feedback, DTs can support adaptive semantic compression, delay-sensitive scheduling, and dynamic satellite cooperation [17].

However, realizing a robust DT-integrated semantic offloading framework presents challenges. First, the high dynamics of satellite topologies, variable link quality, and fluctuating task demands require offloading decisions that can adapt in real time, demanding low-delay feedback and predictive control loops, which DTs are suitable to provide [18]. Second, deep learning tasks vary in data modality, priority, and resource requirements, and treating these tasks uniformly can cause inefficient resource utilization and suboptimal offloading decisions [19]. Third, semantic compression presents a trade-off between reduced communication overhead and potential semantic distortion. Excessive compression may degrade task accuracy, whereas minimal compression increases delays and energy usage [9, 20]. Therefore, dynamic, task-aware semantic encoding is crucial.

Furthermore, optimizing multiple decision variables, such as offloading strategy, semantic compression level, transmission power, and edge server selection, requires joint consideration under strict delay and energy constraints, which are typical of remote IoT applications. Often, existing approaches independently address these variables, ignoring the interdependencies between semantic fidelity, system dynamics, and DT synchronization overhead. These challenges are amplified in integrated satellite-ground networks, where high mobility, intermittent links, and limited satellite resources substantially complicate real-time optimization compared to traditional terrestrial MEC systems.

### *1.2. Related Work*

#### *1.2.1. Conventional Offloading in LEO-MEC*

Task offloading has been explored as a strategy to shift computational workloads from resource-constrained user devices (UDs) to more capable infrastructure, such as mobile cloud computing [21], MEC [22], and fog computing [23] devices. Offloading to LEO-MEC systems has gained attention due to the improved performance in remote or infrastructure-limited regions. Recent studies have proposed various offloading schemes designed for LEO-MEC systems to reduce the delay and energy consumption [24–26]. For example, Cui et al. [24] presented a hybrid approach that combines the Lagrange multiplier method with deep reinforcement learning (DRL) to minimize the weighted sum of the delay and energy cost. However, their work neglects the connectivity duration constraints imposed by the high orbital mobility of LEO satellites. Similarly, Zhou et al. [26] formulated a constrained MDP (Markov decision process) to minimize task delay in satellite—ground integrated networks but considered only a single satellite and did not account for intermittent link connectivity. In contrast, the present work explicitly models dynamic connectivity with multiple LEO-MEC satellites and remote UDs.

Other studies have employed LEO satellites as backhaul relays to cloud data centers, bypassing their onboard processing potential. For instance, Cao et al. [25] and Zhu et al. [26] proposed offloading schemes that transmit tasks to terrestrial MEC servers or remote cloud platforms via satellite relays. Although these methods consider energy and delay constraints, they underutilize the computational capabilities of LEO satellites. The present work differs by assuming the absence of a ground MEC infrastructure, instead proposing a direct offloading model where LEO satellites perform edge computing. This architecture mitigates satellite-to-cloud propagation delays and considerably improves task execution delay and user experience [11, 27].

#### *1.2.2. Semantic-Aware Offloading in LEO-MEC*

By transmitting only task-relevant semantic features, rather than the full raw data, SemCom offers a novel paradigm to reduce communication overhead [18, 28]. This concept has recently been extended to satellite-borne edge computing networks, where bandwidth and energy constraints are severe [14]. Further, SemCom introduces a semantic transmission rate, focusing on information meaning rather than bit fidelity, allowing highly efficient data offloading for various modalities, including images, video, and text [9–11].

Several studies have investigated joint optimization strategies that concurrently consider offloading decisions, resource allocation, and semantic compression [18, 29, 30]. For example, Zheng et al. [29] suggested a framework that optimizes compression and offloading under task delay constraints using convex approximation methods. In addition, Hoa et al. [30] investigated an uncrewed aerial vehicle-assisted semantic offloading system, where tasks are semantically compressed using scene graphs before being transmitted to a Metaverse platform. Their solution, based on proximal policy optimization (PPO) and advantage actor-critic algorithms, dynamically assigns computing resources to balance the compression cost and transmission load.

Although these studies have highlighted the benefits of SemCom, many of them have neglected the system-level awareness and predictive capabilities offered by DT technology. Recent advances [14] have incorporated federated learning and scheduling into SemCom satellite-born edge computing frameworks but omit DT integration, which is necessary for dynamic, context-aware optimization in LEO-MEC networks.

### *1.2.3. DT-Assisted Semantic-Aware Offloading*

Due to its ability to model real-time system states and enable proactive decision-making, DT-assisted offloading has gained traction in edge computing [15][31–33]. For instance, Do-Duy et al. [31] proposed a DT-enabled MEC architecture to minimize the delay in industrial IoT environments, optimizing transmit power, edge server selection, and offloading strategies under quality of service constraints. Further, Ji et al. [32] introduced a DT-assisted multiagent cooperative framework for satellite–terrestrial edge computing, solving a complex optimization problem with a distributed actor–critic method.

Masaracchia et al. [33] envisioned the convergence of DT, SemCom, and Open radio access network for sixth-generation systems, emphasizing architectural collaborations to meet ultra-reliable low-latency communication demands. Moreover, Li et al. [15] explored DT synchronization strategies using SemCom in terrestrial MEC environments, applying a soft actor–critic-based DRL algorithm to manage synchronization, resource allocation, and transmission power adaptively.

Unlike these studies, this work addresses the integration of DT, SemCom, and task offloading in a highly dynamic and resource-constrained satellite-edge environment. The proposed DT-driven semantic task offloading framework employs multiple LEO-MEC satellites to deliver low-latency, energy-efficient task execution for remote IoT devices, explicitly considering short contact windows, intermittent connectivity, and heterogeneous task demands in such systems.

## *1.3. Motivation, Contribution, and Organization*

This work is motivated by the pressing need for intelligent, low-latency, and energy-efficient data offloading in remote IoT environments, where connectivity is intermittent and infrastructure is restricted. Traditional offloading strategies often fail to adapt to dynamic and resource-constrained satellite—ground integrated networks, particularly in the presence of LEO satellite mobility and heterogeneous workload demands. The integration of SemCom with LEO-MEC systems offers a promising solution by reducing the communication overhead via task-relevant feature transmission. However, realizing this potential requires a carefully designed architecture supporting local processing on UDs and remote processing on LEO-MEC servers. Goal-oriented semantic coders must adapt in real time to changing transmission content, posing challenges due to the limited energy, mobility, and service continuity requirements.

Existing distributed learning frameworks for SemCom, designed for terrestrial networks [34, 35], often ignore the participation of LEO-MEC servers and are unsuitable for short-duration satellite links. Moreover, SemCom shifts the offloading paradigm by increasing the computational load while reducing the bandwidth usage, necessitating adaptive offloading strategies that consider the semantic fidelity, energy constraints, and connectivity windows, which are often ignored in conventional LEO-MEC research.

This work proposes a novel DT-driven semantic task-offloading framework for LEO-MEC-enabled remote IoT networks to bridge these gaps. The proposed framework dynamically adapts to network conditions, resource availability, and task requirements by combining the predictive capabilities of DTs with the communication efficiency of SemCom, ensuring robust and efficient offloading in highly dynamic LEO-MEC environments. The primary contributions of this work are summarized as follows:

- **DT Integration:** This method instantiates a DT reference architecture in the satellite-ground integration network system to enable real-time modeling of the task status, semantic deviation, and encoder-decoder mismatch. This architecture encapsulates the dynamic evolution of task profiles, facilitating intelligent encoder selection and enhancing the overall system performance.

- **Semantic-Aware Offloading Optimization:** This novel offloading optimization problem incorporates DT-driven task deviation and semantic similarity uncertainty. A resource-efficient algorithm minimizes end-to-end delay while satisfying semantic fidelity constraints to address this problem.

- **PPO-Based Deep Reinforcement Learning Solution:** This work presents a PPO-based actor-critic algorithm to address the high-dimensional and nonconvex optimization problem. The agent interacts with a DT-driven simulation environment to iteratively learn robust and adaptive offloading strategies under uncertain and dynamic LEO-MEC network conditions.

- **Experimental Validation:** Extensive simulations are conducted to evaluate the proposed framework. The results reveal that the DT-driven semantic offloading approach considerably outperforms the baseline methods in terms of end-to-end task delay and semantic fidelity, particularly under stringent energy constraints in diverse LEO-MEC scenarios.

Beyond these technical contributions, the proposed framework is designed to address several operational challenges in satellite—ground integrated networks. First, the framework enhances bandwidth efficiency by transmitting only semantically relevant task features, alleviating the bandwidth limitations of LEO satellite links. Second, this framework improves the robustness to intermittent connectivity by applying a progressive semantic transmission strategy. The system maintains effective DT synchronization even under frequent satellite handovers and temporary link disruptions by prioritizing the delivery of highly important semantic content. Third, this framework addresses the resource constraints of satellite-based edge nodes by offloading semantically compressed data, substantially reducing the computational and energy load on LEO-MEC servers, promoting sustainable and reliable operations in resource-limited environments. Collectively, these design choices emphasize that the proposed DT-driven semantic offloading framework is not a direct adaptation of terrestrial MEC strategies but a fundamentally novel solution designed to satisfy the dynamic, constrained, and delay-sensitive demands of next-generation satellite—ground integrated network architectures.

The remainder of this paper is organized as follows. Section 2 introduces the system model, detailing the formal problem formulation and the architecture and components of the proposed DT-driven semantic data offloading framework for LEO-MEC-enabled remote IoT devices. Next, Section 3 describes the proposed optimization solution and training algorithms for policy learning. Then, Section 4 presents the experimental results and performance evaluation. Finally, Section 5 concludes the paper by summarizing the critical findings and outlining the potential directions for future research.

## 2. System Model and Problem Formulation

### *2.1. System Model*

As illustrated in fig. 1, we consider a DT-driven, semantic-aware image offloading system that consists of a set of UDs $\mathcal{U} = \{1, 2, \ldots, U\}$, a set of LEO satellites equipped with MEC servers $\mathcal{L} = \{1, 2, \ldots, L\}$, and a set of time slots $\mathcal{T} = \{1, 2, \ldots, T\}$. The overall architecture integrates a physical layer and a DT layer. The physical layer comprises geographically dispersed UDs (e.g., phones, smart sensors, ships, autonomous vehicles) operating in remote or infrastructure-scarce environments. These UDs generate computation-intensive tasks that can be processed locally or offloaded to LEO-MEC nodes over satellite links. The DT layer serves as a high-fidelity virtual replica of the physical environment, enabling real-time monitoring, predictive modeling, and intelligent control. It includes: (i) a unified data repository for collecting, storing, and evaluating performance metrics; (ii) unified data models, consisting of a basic model that captures network topology and a functional model for optimizing key decision variables (offloading strategy, semantic compression ratio, edge server selection, and transmit power); and (iii) DT entity management, responsible for model synchronization, network element lifecycle management, and continuous system improvement. DT models are updated dynamically via optimization and emulation to maintain accuracy and responsiveness.

A closed-loop control architecture tightly couples the physical and digital layers. Real-time data from the physical environment are continuously synchronized with the corresponding DTs, which analyze the information and issue context-aware control decisions for resource allocation and task scheduling. Each UD maintains a dedicated DT to track device-specific parameters (e.g., current task status, semantic content relevance, processing load). Similarly, each LEO-MEC server is associated with an edge-level DT that monitors available computing resources, task queues, and semantic workload profiles. This DT-enabled infrastructure supports fine-grained, predictive offloading decisions and coordinated resource orchestration across the network. The system operates under a sense-then-offload paradigm [36], whereby UDs first capture raw image data and then decide whether to process locally or offload to a satellite.

### *2.2. Sensing Model*

The sensing module (e.g., cameras and sensors) on each UD operates at a device-specific sampling rate, which determines the number of samples or images collected per second; accordingly, the data acquisition rate depends on the sampling rate and image size. At time slot $t$, UD $u$ must collect a certain amount of images (or image bits), denoted by $I_u[t] \in [I^{\min}, I^{\max}]$, to ensure effective DT synchronization, where $I^{\min}$ and $I^{\max}$ represent the minimum and maximum required sensed data sizes, respectively. The perceived load collected by UD $u$ at time $t$ is given by [15]:

$$I_u[t] = \eta \, s_u[t], \tag{1}$$

where $\eta \in [0, 1]$ denotes the time allocation factor between detection and processing, and $s_u[t]$ denotes the UD sensing rate $u$ at time $t$. The delay incurred during the sensing and capturing phases corresponds to the time to acquire $I_u[t]$ data bits and is calculated as follows:

$$t_u^{\text{sens}}[t] = \frac{I_u[t]}{s_u}[t]. \tag{2}$$

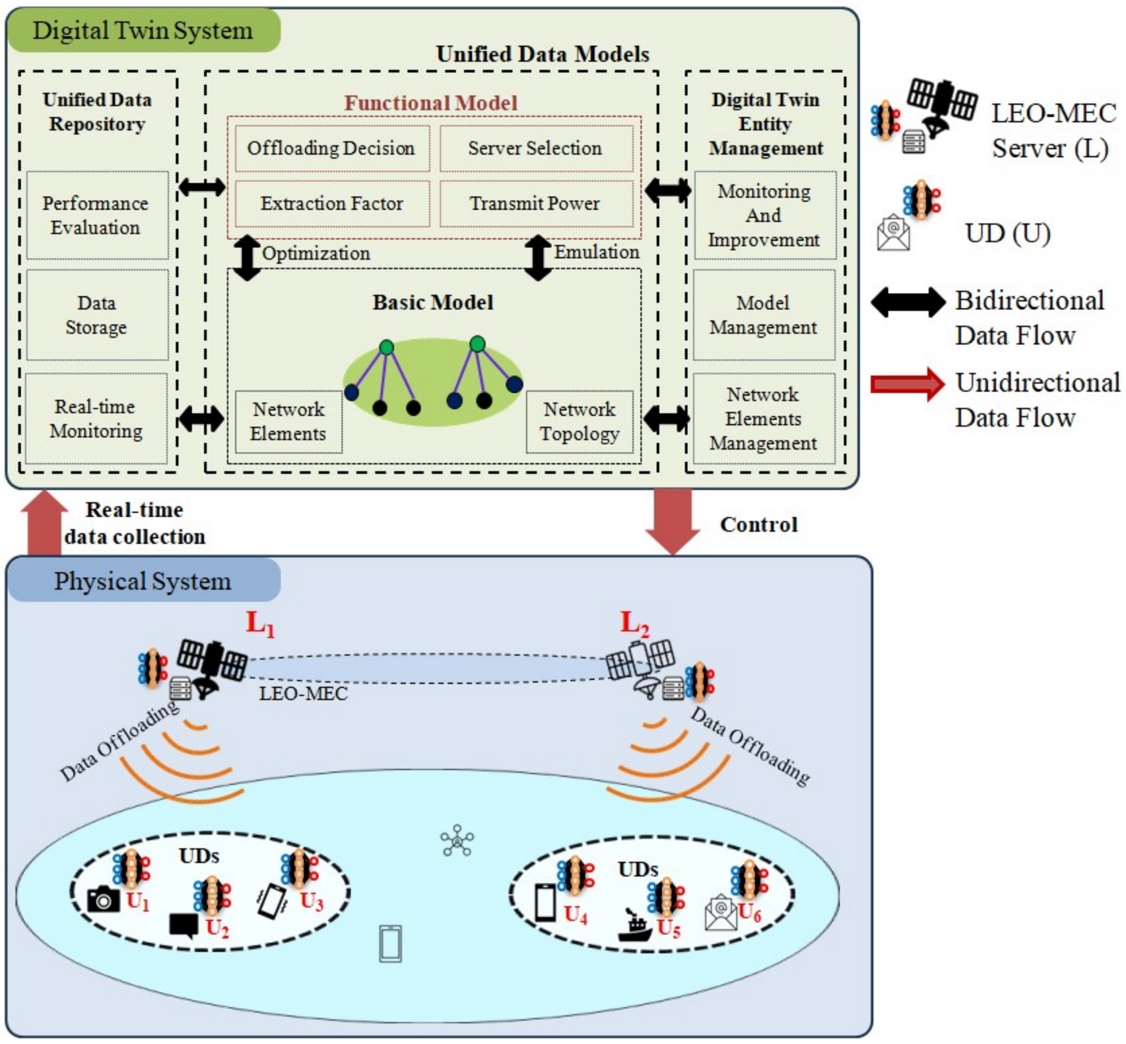

Fig. 1. System model.

The associated energy consumption for sensing at time $t$ is expressed as follows:

$$E_u^{\text{sens}}[t] = p_u^{\text{sens}}[t] \cdot t_u^{\text{sens}}[t], \tag{3}$$

where $p_u^{\text{sens}}[t]$ represents power consumption during sensing data.

## *2.3. Semantic Transmission Model*

Each UD may offload its sensed image to exactly one LEO–MEC server per slot. Let $\alpha_{u,l}[t] \in \{0,1\}$ denote the server association (with $\sum_l \alpha_{u,l}[t] \leq 1$ for all $u, t$). Orthogonal frequency-division multiple acces is adopted so that each time–frequency resource block is assigned to a single UD and inter-user interference is neglected.

*Decision vector.* For each slot $t$, the optimization chooses $\mathcal{D}[t] \triangleq \{\alpha_{u,l}[t],\ b_{u,l}[t] \geq 0,\ p_{u,l}^{\text{tr}}[t] \in [0, P_u^{\max}],\ \mu_u[t] \in (0,1]\}_{u,l}$. Here, $\alpha_{u,l}[t]$ (association), $b_{u,l}[t]$ (bandwidth), $p_{u,l}^{\text{tr}}[t]$ (transmit power), and $\mu_u[t]$ (semantic compression ratio) are the decision variables.

*Given parameters.* The channel gain $h_{u,l}[t]$ is measured/estimated at the beginning of slot $t$ and the noise PSD $n_0 > 0$ is known; both are parameters. Other constants include $\bar{I}_u$ and the calibration functions $k_u(\cdot)$ and $\xi_u(\cdot)$ [19, 20].

**Table 1.** Notation and Definitions

| Notation | Definition | Notation | Definition |
|---|---|---|---|
| $\mathcal{U}/U$ | Number/set of user devices (UDs) | $I_u[t]$ | Image collected by $u$ at time slot $t$ |
| $\mathcal{L}/L$ | Number/set of LEO-MEC servers | $s_u[t]$ | Sensing rate of $u$ at time slot $t$ |
| $p^{\text{tr}}_{u,l}[t]$ | Transmit power of $u$ to $l$ at time $t$ | $\eta$ | Time allocation factor |
| $\varepsilon_u$ | The energy budget of UD $u$ | $p^{\max}_u$ | Maximum transmission power of UDs $u$ |
| $\alpha_{u,l}[t]$ | LEO-MEC server $l$ selection at time $t$ | $\beta_{u,l}$ | Binary offloading decision at time slot $t$ |
| $\lambda_u[t]$ | Per-bit CPU-cycle requirement of $u$ at time $t$ | $\lambda_l[t]$ | Per-bit CPU-cycle requirement of $l$ at time $t$ |
| $f_u[t]$ | CPU frequency (in cycles/s) of UD $u$ at time slot $t$ | $f^u_l[t]$ | CPU frequency (in cycles/s) of LEO-MEC server $l$ at time slot $t$ |
| $\hat{f}_u[t]$ | CPU frequency deviation between UDs $u$ and its DT at time slot $t$ | $\hat{f}^u_l[t]$ | CPU frequency deviation between LEO-MEC server $l$ and its DT at time slot $t$ |
| $t^{\text{sens}}_u[t]$ | The delay time to sense and capture the image data | $t^{\text{loc}}_u[t]$ | The estimated local computation delay at the DT layer |
| $\tilde{t}^{\text{loc}}_u[t]$ | The local computation delay accounts for DT estimation error and sensing delay | $\Lambda_u$ | CPU's effective switched capacitance coefficient of UD $u$ |
| $E^{\text{sens}}_u[t]$ | Energy consumption for sensing data at time slot $t$ | $p_u\text{sens}[t]$ | Power consumed per bit of sensed data at time slot $t$ |
| $t^{\text{enc}}_u[t]$ | Semantic encoding delay at UDs $u$ | $\theta$ | Encoder/decoder parameter |
| $E^{\text{loc}}_u[t]$ | Energy consumed by UD $u$ during local processing | $w_u[t]$ | computational demands for semantic compression of UD $u$ at time $t$ |
| $d^{\max}$ | Maximum delay threshold | $E^{\text{off}}_{u,l}$ | Offloading energy consumption |
| $\psi_u[t]$ | Total synchronization delay of UD $u$ | $t^{\text{off}}_{u,l}$ | Overall offloading delay |
| $b_{u,l}$ | The allocated bandwidth between UDs $u$ and LEO-MEC server $l$ | $E^{\text{enc}}_u$ | Energy consumption of UD $u$ during semantic encoding |
| $t^{\text{cov}}_l[t]$ | Duration of connectivity of LEO-MEC server $l$ | $E^{tot}_u[t]$ | Total energy consumption of UDs $u$ at time slot $t$ |
| $\Xi$ | Time budget to deliver the payload | $B_l$ | bandwidth capacity of server $l$ |

*Induced SNR.* Given a choice of $(p^{\text{tr}}_{u,l}[t], b_{u,l}[t])$, the instantaneous SNR on link $(u, l)$ at slot $t$ is

$$\gamma_{u,l}[t] \;=\; \frac{p^{\text{tr}}_{u,l}[t]\, h_{u,l}[t]}{b_{u,l}[t]}\, n_0, \tag{4}$$

*Semantic rate.* Let the average semantics per image for UD $u$ be $\bar{I}_u$ (suts/image). A DeepSC-style encoder produces $k_u(\mu_u[t])$ semantic symbols per image at compression $\mu_u[t] \in (0, 1]$. With passband signaling, the symbol rate scales with allocated bandwidth. Denote by $\xi_u\big(\gamma, k\big) \in [0, 1]$ the semantic recovery accuracy at SNR $\gamma$ and symbolization level $k$. The semantic transmission rate (suts/s) on the active link is

$$\Gamma_{u,l}[t] \;=\; \alpha_{u,l}[t]\; b_{u,l}[t]\; \frac{\mu_u[t]\; \bar{I}_u}{k_u(\mu_u[t])}\; \xi_u\big(\gamma_{u,l}[t],\, k_u(\mu_u[t])\big), \tag{5}$$

*Resource constraints.* Per-server bandwidth and association obey $\sum_u b_{u,l}[t] \;\le\; B_l, \quad \sum_l \alpha_{u,l}[t] \le 1, \quad 0 \le p^{\text{tr}}_{u,l}[t] \le P^{\max}_u$.

*Semantic spectral efficiency.* By dividing eq.(5) by the bandwidth, we obtain the semantic spectral efficiency (S-SE)

$$\text{S}-\text{SE}_{u,l}[t] \;=\; \frac{\Gamma_{u,l}[t]}{b_{u,l}[t]} \qquad \text{(suts/s/Hz)}. \tag{6}$$

### *2.4. Computation Model*

Each UD can perform computational tasks locally or offload them to a LEO-MEC server. Local execution is suitable for lightweight tasks when the UD has sufficient computational resources and the overhead of offloading (including transmission delay, energy consumption, and bandwidth usage) outweighs its potential benefits. Conversely, offloading is advantageous for computationally intensive or time-sensitive tasks, particularly when the UD has limited processing capabilities or energy reserves.

Due to the nature of image-based data, partial offloading is infeasible because dividing image tasks between local and remote execution compromises the data integrity and interpretability. Hence, a binary offloading strategy is employed, where each task is fully processed locally or offloaded entirely to the selected LEO-MEC server.

### *2.4.1. Local Computation*

The estimated local computation delay at the DT layer is given by the following:

$$t_u^{\text{loc}}[t] = \frac{\lambda_u[t] I_u[t]}{f_u[t]}. \tag{7}$$

Due to possible mismatches between the DT model and physical device, the estimated computational capability $\hat{f}_u[t]$ of the DT may deviate from the true CPU frequency $f_u[t]$ [37]. This estimation error introduces a delay gap, calculated as follows [31, 38]:

$$\Delta t_u^{\text{loc}}[t] = \frac{\lambda_u[t] I_u[t] \hat{f}_u[t]}{f_u[t](f_u[t] - \hat{f}_u[t])}. \tag{8}$$

Accordingly, the total local computation delay, which accounts for DT estimation error and sensing delay, is

$$\tilde{t}_u^{\text{loc}}[t] = t_u^{\text{sens}} + t_u^{\text{loc}}[t] + \Delta t_u^{\text{loc}}[t]. \tag{9}$$

In addition to the delay, local computation also incurs energy consumption. Based on the dynamic voltage scaling theory, the energy consumed by UD $u$ during local processing is modeled as

$$E_u^{\text{loc}}[t] = \Lambda_u \lambda_u[t] I_u[t] (f_u[t] - \hat{f}_u[t])^2 + E_u^{\text{sens}}[t], \tag{10}$$

where $\Lambda_u$ represents the effective switched capacitance coefficient of the processor.

### *2.4.2. Remote Computation*

In remote computation, the offloading process extracts semantic data, wherein each UD compresses the raw image data into semantically meaningful representations using a tiny machine learning-based semantic encoder. This encoding introduces additional computational overhead on the device side. The semantically compressed data are transmitted to the LEO-MEC server, where a corresponding semantic decoder reconstructs the original content. This bidirectional semantic processing, comprising encoding at the UD and decoding at the edge, incurs nonnegligible computational workloads at both ends. The degree of semantic compression is governed by the semantic compression ratio. The semantic encoding delay is as follows [39]:

$$t_u^{\text{enc}}[t] = \frac{\lambda_u[t](I_u[t], \theta, \mu_u[t])}{f_u[t]}, \tag{11}$$

where $\theta$ denotes the encoder model. Due to mismatches between the real-time computational capacity and DT-estimated capability $\hat{f}_u[t]$, a delay estimation gap arises [38, 40], expressed as follows:

$$\Delta t_u^{\text{enc}}[t] = \frac{\lambda_u[t] w_u[t] \hat{f}_u[t]}{f_u[t](f_u[t] - \hat{f}_u[t])}. \tag{12}$$

where $w_u[t]$ denotes the computation cost incurred by UDs $u$ for semantic compression during slot $t$. Hence, the total semantic encoding delay is

$$\tilde{t}_u^{\text{enc}}[t] = t_u^{\text{enc}}[t] + \Delta t_u^{\text{enc}}[t]. \tag{13}$$

The baseline energy consumption at the UD during semantic encoding is modeled as follows:

$$E_u^{\text{enc}}[t] = \Lambda_u \lambda_u[t] w_u[t] (f_u[t] - \hat{f}_u[t])^2. \tag{14}$$

The transmission delay from UD $u$ to LEO-MEC server $l$ accounts for the time to upload semantically compressed DT synchronization data. If the offloading decision corresponds to the full task transmission (i.e., $\beta_{u,l}[t] = 1$), the entire dataset is sent without any local execution. The uplink transmission delay is defined as follows:

$$t_{u,}l^{\text{tx}}[t] = \frac{\mu_u[t] I_u[t]}{\Gamma_{u,l}[t]}. \tag{15}$$

The corresponding transmission energy consumption is calculated by

$$E_{u,l}^{\text{tx}}[t] = p_{u,l}^{\text{tr}}[t] t_{u,l}^{\text{tx}}[t]. \tag{16}$$

After transmitting from the UDs to the LEO-MEC server, the semantic information must be decoded at the edge to reconstruct the original sensed data. The delay associated with this semantic recovery process at the LEO-MEC server can be expressed as follows [15, 41]:

$$t_l^{\text{dec}}[t] = \frac{\lambda_l(I_u[t], \theta, \mu_u[t])}{f_l^u[t]}. \tag{17}$$

The decoding delay deviation is defined as follows to account for the discrepancies between the actual available computational capacity and the DT-estimated value $\hat{f}_l^u[t]$:

$$\Delta t_l^{\text{dec}}[t] = \frac{\mu_u[t] I_u[t] \lambda_l \hat{f}_l^u[t]}{f_l^u[t](f_l^u[t] - \hat{f}_l^u[t])}. \tag{18}$$

Accordingly, the total decoding delay at the LEO-MEC server becomes

$$\tilde{t}_l^{\text{dec}}[t] = t_l^{\text{dec}}[t] + \Delta t_l^{\text{dec}}[t]. \tag{19}$$

Before initiating remote computation at the LEO-MEC server $l$, whether the server has sufficient computational capacity to process the received task data within the allowable deadline must be evaluated. Let $I_l[t]$ denote the data size in bits after semantic decoding at time slot $t$. The remote computation delay at LEO-MEC server $l$ is calculated as follows:

$$t_l^{\text{comp}}[t] = \frac{\lambda_l I_l[t]}{f_l^u[t]}. \tag{20}$$

The resulting delay deviation is calculated as follows to account for discrepancies between the DT-estimated capacity $\hat{f}_l^u[t]$ and actual server capacity $f_l^u[t]$:

$$\Delta t_l^{\text{comp}}[t] = \frac{\lambda_l I_l[t] \hat{f}_l^u[t]}{f_l^u[t](f_l^u[t] - \hat{f}_l^u[t])}. \tag{21}$$

Thus, the total remote computation delay is expressed as follows:

$$\tilde{t}_l^{\text{comp}}[t] = t_l^{\text{comp}}[t] + \Delta t_l^{\text{comp}}[t]. \tag{22}$$

Due to the satellite altitude, the round-trip propagation delay is nonnegligible:

$$t_{u,l}^{\text{prop}} = \frac{2d}{v}, \tag{23}$$

where $d$ denotes the distance to the satellite and $v$ represents the speed of light.

In scenarios with server overload or delay constraint violations, the centralized DT controller may enable inter-satellite load balancing via inter-satellite links (ISLs). If neighboring LEO-MEC servers have spare capacity, tasks may be dynamically reassigned. Although this strategy enhances adaptability and ensures the quality of service, the added propagation and processing delays require further evaluation, which is left for future work. The overall offloading delay is

$$t_{u,l}^{\text{off}}[t] = t_u^{\text{sens}} + \tilde{t}_u^{\text{enc}}[t] + t_{u,l}^{\text{tx}}[t] + \tilde{t}_l^{\text{dec}}[t] + \tilde{t}_l^{\text{comp}}[t] + t_{u,l}^{\text{prop}}. \tag{24}$$

The total synchronization delay for UD $u$ at time $t$ is

$$\psi_u[t] = \begin{cases} \tilde{t}_u^{\text{loc}}[t], & \text{if } \beta_{u,l}[t] = 0, \\ t_{u,l}^{\text{off}}[t], & \text{if } \beta_{u,l}[t] = 1. \end{cases} \tag{25}$$

The total energy consumption of offloading is

$$E_u^{\text{off}} = E_u^{\text{sens}} + E_u^{\text{enc}} + E_{u,l}^{\text{tx}}. \tag{26}$$

Finally, the total energy consumption by UD $u$ at time slot $t$ is

$$E_u^{\text{tot}} = \begin{cases} \tilde{E}_u^{\text{loc}}, & \text{if } \beta_{u,l}[t] = 0, \\ E_u^{\text{off}}, & \text{if } \beta_{u,l}[t] = 1. \end{cases} \tag{27}$$

This energy model focuses on the UDs because LEO-MEC servers are assumed to operate using sustainable power sources, such as solar energy.

## *2.5. Problem Formulation*

In the DT-driven semantic-aware task-offloading framework, each UD can run computational tasks locally or offload them to a designated LEO-MEC server. The aim is to minimize the average DT synchronization delay over all UDs by jointly optimizing four interdependent control variables over time: the offloading policy, LEO-MEC selection, semantic compression ratio and transmission power control. This optimization is subject to limitations on energy consumption, delay requirements, computational resources, and semantic fidelity.

This work formulates the optimization problem to minimize the total delay of the DT synchronization for each UD $u$ over a finite time horizon $T$:

$$\begin{aligned}
\mathcal{P}: \quad & \min_{\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{p},\boldsymbol{\mu}} \sum_{t\in\mathcal{T}}\sum_{u\in\mathcal{U}} \psi_u[t] && && (28)\\
\text{s.t.} \quad & \sum_{t\in\mathcal{T}} E_u^{\text{tot}}[t] \le \varepsilon_u, && \forall u\in\mathcal{U}, && \text{(C1)}\\
& \beta_{u,l}[t]\in\{0,1\}, && \forall u\in\mathcal{U},\, \forall l\in\mathcal{L},\, \forall t\in\mathcal{T}, && \text{(C2)}\\
& \alpha_{u,l}[t]\in\{0,1\}, && \forall u\in\mathcal{U},\, \forall l\in\mathcal{L},\, \forall t\in\mathcal{T}, && \text{(C3)}\\
& 0\le p_{u,l}^{\text{tr}}[t]\le p_u^{\max}, && \forall u\in\mathcal{U},\, \forall l\in\mathcal{L},\, \forall t\in\mathcal{T}, && \text{(C4)}\\
& \mu^{\min}\le \mu_u[t]\le 1, && \forall u\in\mathcal{U},\, \forall t\in\mathcal{T}, && \text{(C5)}\\
& 0\le f_u[t]\le f_u^{\max}, && \forall u\in\mathcal{U},\, \forall t\in\mathcal{T}, && \text{(C6)}\\
& 0\le \sum_{u\in\mathcal{U}} f_l^u[t]\le f_l^{\max}, && \forall l\in\mathcal{L},\, \forall t\in\mathcal{T}, && \text{(C7)}\\
& \psi_u[t]\le d^{\max}, && \forall u\in\mathcal{U},\, \forall t\in\mathcal{T}, && \text{(C8)}\\
& \alpha_{u,l}[t]=1 \;\Rightarrow\; t_{u,l}^{\text{off}}[t]\le t_l^{\text{cov}}[t], && \forall u\in\mathcal{U},\, \forall l\in\mathcal{L},\, \forall t\in\mathcal{T}. && \text{(C9)}
\end{aligned}$$

Several constraints are imposed to ensure the feasibility of the proposed optimization framework. The constraint in (C1) ensures that the total energy consumed by the UD $u$ at time $t$ does not exceed the available energy budget $\varepsilon_u$. The constraint in (C2) restricts the binary offloading, where $\beta_{u,l}[t] = 0$ corresponds to the complete local computation and $\beta_{u,l}[t] = 1$ denotes complete data offloading. The constraint in (C3) employs edge server selection $\alpha_{u,l}[t]$ specifying whether UD $u$ selects LEO-MEC server $l$ for data offloading. The constraint in (C4) limits the uplink transmission power to a permissible range defined by the hardware and regulatory limits. The constraint in (C5) enforces a lower bound on the semantic compression ratio $\mu_u[t]$ to preserve semantic relevance, ensuring a minimum level of semantic fidelity. The constraint in (C6) guarantees that the local CPU frequency for task execution does not exceed the maximum computational capability of the UD. On the server side, the constraint in (C7) restricts the total computation resources allocated by a LEO-MEC server to remain within its capacity, $f_l^{\max}$. Furthermore, the constraint in (C8) ensures that the total DT synchronization delay $\psi_u[t]$ does not exceed the user-defined delay threshold $d^{\max}$, satisfying the delay-sensitive application requirements. Finally, the constraint in (C9) ensures that all offloading and computation tasks are completed within the coverage time window $t_l^{\text{cov}}[t]$ of the satellite, considering the limited connectivity period between the UD and LEO satellite.

Solving Problem $\mathcal{P}$ presents substantial challenges due to its intrinsic complexity as a mixed-integer programming problem. The problem involves discrete decision variables (e.g., offloading strategy, LEO-MEC server selection, and discretized semantic compression levels) and continuous variables (e.g., transmission power allocations), rendering the problem non-deterministic polynomial time hard [3, 42]. The difficulty is compounded by the highly dynamic and time-sensitive network environment, influenced by the continuous movement and high mobility of LEO-MEC servers. These dynamic characteristics make traditional optimization methods unsuitable for real-time decision making in non-stationary conditions. This work reformulates the original problem as an MDP and develops a novel DRL-based solution framework, drawing inspiration from recent advancements [43–45] to address these complexities.

## 3. Proposed Solution

### *3.1. MDP-Based DT-driven Semantic-Aware Offloading*

This work reformulates the original delay minimization problem as an MDP to address the joint optimization of offloading decisions, edge server selection, semantic compression, and power allocation in dynamic LEO-MEC environments. The MDP provides a mathematically grounded framework for sequential decision-making under uncertainty, making it suitable for modeling the dynamic characteristics of LEO satellite networks, such as their time-varying link availability, fluctuating channel conditions, and constrained device resources. This formulation enables each UD to act as an agent that continuously observes system states and learns optimal actions over time via interactions with the environment. The MDP setting accommodates discrete and continuous decision variables and enables long-term performance optimization by maximizing the cumulative reward. The model supports predictive and context-aware policy adaptation aligned with evolving network dynamics by integrating DT feedback into the state representation.

Formally, the MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T})$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ represents the action space, $\mathcal{R}$ indicates the reward function, and $\mathcal{T}$ denotes the transition probabilities. Each UD interacts with the DT-enabled LEO-MEC network to gather experience, update its policy, and make decisions that balance the delay minimization, energy efficiency, and semantic fidelity.

#### *1) State Space*

At each decision epoch $t$, a UD $u$ observes the system state $\mathcal{S}_t$, defined as follows:

$$\mathcal{S}_t = \left[I_u[t],\ h_{u,l}[t],\ d^{\max}[t], \epsilon[t],\ f_l^u[t], \mu_u[t-1]\right], \tag{29}$$

where each element captures the task size, channel gain, delay constraints, remaining energy, computational resources, and previous semantic compression ratio. This compact yet expressive representation captures the local resource availability and dynamic environmental conditions, enabling the agent to make informed and adaptive policy decisions.

#### *2) Action Space*

The action taken at time $t$ is defined as follows:

$$a_t = [\beta_{u,l}[t],\ \alpha_{u,l}[t],\ \mu_u[t],\ p_{u,l}^{\mathrm{tr}[t]}]. \tag{30}$$

This action space provides the agent with fine-grained control over the task placement and communication efficiency, fostering adaptive and resource-aware decision-making in dynamic LEO-MEC environments.

#### *3) Reward Function*

This approach reformulates the reward structure to suit a maximization framework, aligning the reinforcement learning objective with the aim of minimizing the total DT synchronization delay. Although DRL aims to maximize the cumulative rewards, this optimization problem is focused on delay minimization. To reconcile this problem, we designed a reward function that penalizes the delay and any violations of the system constraints, transforming the minimization objective into an equivalent maximization problem.

At each decision time step $t$, the reward $r[t]$ is the negative sum of the DT synchronization delay $\psi_u[t]$ and a penalty term $\kappa$, formulated as follows:

$$r[t] = -\psi_u[t] - \kappa. \tag{31}$$

The penalty term $\kappa$ aggregates the constraint violations, expressed as $\kappa = \kappa_E + \kappa_{f_u} + \kappa_{f_l^u}$, where each subcomponent addresses a specific system limitation. Specifically, $\kappa_E$ penalizes the excessive energy consumption of a UD when its cumulative energy usage exceeds a predefined threshold. The term $\kappa_{f_u}$ captures violations of the computational capacity of the UD. Last, $\kappa_{f_l^u}$ imposes a penalty when the processing workload assigned to a LEO-MEC server surpasses its maximum capacity. These penalties are defined as follows: $\kappa_E = \mathbb{I}\left(\sum_{t=1}^{T} E^{tot}[t] > \varepsilon[t]\right)$, $\kappa_{f_u} = \mathbb{I}\left(\sum_{u\in\mathcal{U}} f_u[t] > f_u^{\max}\right)$, and $\kappa_{f_l^u} = \mathbb{I}\left(\sum_{l\in\mathcal{L}} f_l^u[t] > f_l^{\max}\right)$, where $\mathbb{I}(\cdot)$ denotes the indicator function, which returns 1 if the condition is satisfied and zero otherwise. This reward design ensures that the reinforcement learning agent minimizes delay and satisfies critical energy and resource constraints. Thus, the primary goal of the agent is to determine the optimal set of decisions $\beta_{u,l}[t], \alpha_{u,l}[t], \mu_u[t]$, and $p_{u,l}^{\text{tr}}[t]$ that maximize the long-term expected discount. At each time slot $t$, the agent observes the current state of the system $s_t$, selects and executes an action $a_t$, and then receives an immediate reward $r_t$ along with the next state $S_{t+1}$. The goal is to identify the sequence of actions that maximizes the long-term discounted return $R_t$, which accounts for both immediate and future rewards. Consequently, the original optimization problem (28) can be reformulated as a reinforcement learning problem with a discount factor $\gamma \in [0, 1]$:

$$\mathbf{R} = \max_{a_t\in\mathcal{A}} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t(S_t, a_t)\right], \tag{32}$$

where $\mathcal{A}$ denotes the feasible action space and $\gamma$ determines the relative importance of future rewards.

### *3.2. Proposed PPO Algorithm*

Our problem is an MDP with a hybrid action space (discrete offloading edge server selection plus continuous semantic ratio and transmit power). We adopt a factorized PPO policy—a categorical head for discrete actions and Gaussian heads for continuous ones—so the agent optimizes all decisions jointly in a single actor–critic framework, as is standard for hybrid control [19, 46]. We choose PPO for its strong stability–complexity trade-off in a constraint-sensitive and quasi-non-stationary setting: the clipped surrogate objective bounds policy updates, curbing oscillations under tight latency/energy penalties, while on-policy learning matches DT-driven operation where state distributions shift as the DT refreshes (reducing bias from stale experience). PPO also has few sensitive hyperparameters and is simpler to tune than parameterized-hybrid schemes that split discrete and continuous learners. In our ablations against a lighter baseline, PPO converged faster and incurred fewer constraint violations than DQN; it also converged faster than A2C and DDQN, consistent with results reported in related MEC control studies [30].Compared to SAC, it required less temperature tuning and compute to reach stable performance. Overall, PPO provides a pragmatic and effective solution for latency-critical hybrid control in LEO-MEC: it handles mixed actions cleanly, remains robust under DT-induced non-stationarity, and delivers stable improvements without onerous tuning.

The LEO-MEC network enabled by DT is modeled as an MDP, where each UD is an agent. At every time step $t$, the agent observes the system state $S_t$ and selects an action $a_t = \{\beta_{redu,l}red[t], \alpha_{u,l}[t], \mu_u[t], p_{u,l}^{\text{tr}}[t]\} \in \mathcal{A}$, which jointly determines $\beta_{u,l}[t]$, $\alpha_{u,l}[t]$, $\mu_u[t]$, and $p_{u,l}^{\text{tr}}[t]$. The reward function, derived from optimization problem $\mathcal{P}$, minimizes total delay while including penalties for energy consumption and semantic inaccuracies, as given in eq.(31).

The PPO algorithm learns a stochastic policy $\pi_\theta(a_t|s_t)$ parameterized by $\theta$ to maximize the expected cumulative reward. The actor network generates actions based on the policy, whereas the critic network estimates the value function to evaluate the quality of selected actions. The reward function is carefully designed to reflect the negative synchronization delay and penalize constraint violations related to energy and resource limits.

The goal of the PPO-based training process is to maximize the expected long-term reward for each trajectory, expressed as follows:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R_t] = \left[ \sum_{t=0}^{T} \gamma^t r_t(S_t, a_t) \right], \tag{33}$$

where $\theta$ denotes the policy parameters, $\tau$ indicates a trajectory generated by the policy $\pi_\theta$, $r_t$ represents the reward at time $t$, and $\gamma \in (0,1)$ indicates the discount factor.

To update the policy, PPO relies on the policy gradient method. The expected return gradient with respect to the policy parameters is calculated as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[ \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|S_t)\, R_t \right], \tag{34}$$

where $R_t$ denotes the cumulative return from time step $t$.

In standard on-policy learning, experience from previous policies is often discarded due to distribution mismatch, making learning inefficient. The PPO mitigates this problem by incorporating an importance sampling ratio, reusing trajectories from recent policies, and stabilizing updates. The importance sampling ratio is formulated as follows:

$$\rho_t(\theta) = \frac{\pi_\theta(a_t \mid S_t)}{\pi_{\theta_{\text{old}}}(a_t \mid S_t)}, \tag{35}$$

where $\pi_{\theta_{\text{old}}}$ denotes the policy employed to collect data.

Accordingly, the policy gradient with importance sampling is the following:

$$\nabla_{\theta'} J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta'}} \left[ \sum_{t=0}^{T} \rho_t(\theta)\, \nabla_\theta \log \pi_\theta(a_t \mid S_t)\, R_t \right]. \tag{36}$$

This formulation enables PPO to benefit from reusing samples while maintaining stable learning by controlling the extent to which the policy can change between updates.

The action-value function is defined to evaluate the quality of an action under a given policy $\pi$:

$$Q^\pi(S,a) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \,\middle|\, S_t = S, a_t = a \right]. \tag{37}$$

Instead of directly comparing the action-value function, the PPO algorithm applies the advantage function, which quantifies the performance improvement that an action provides compared to the average performance at a given state:

$$A^\pi(s,a) = Q^\pi(S,a) - V^\pi(S), \tag{38}$$

where $V^\pi(S)$ denotes the state-value function under policy $\pi$.

Further, the policy gradient of the expected cumulative return with respect to the policy parameters can also be expressed via the action-value function:

$$\nabla_\theta V^{\pi_\theta}(s) = \mathbb{E}_{a \sim \pi_\theta} \left[ \nabla_\theta \log \pi_\theta(a|S) Q^{\pi_\theta}(S,a) \right]. \tag{39}$$

By substituting the action-value function with the advantage function, the policy gradient becomes

$$\nabla_\theta J(\theta) = \mathbb{E}_{(S,a)\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|S) A^{\pi_\theta}(S,a)\right]. \tag{40}$$

In practice, $Q^\pi(S,a)$ and $V^\pi(S)$ are not directly available, so the advantage is estimated using temporal-difference (TD)learning:

$$\hat{A}_t = \rho_t + \gamma V'_\theta(S_{t+1}) - V'_\theta(S_t), \tag{41}$$

where $V'_\theta$ represents the critic network parameterized by $\theta'$.

However, temporal difference-based estimates can suffer from high variance. The PPO algorithm employs the generalized advantage estimation (GAE) to balance bias and variance [47], providing a smoother and more robust estimate of the advantage:

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\vartheta)} = \sum_{l=0}^{T-t} (\gamma\vartheta)^l \delta_{t+l}, \tag{42}$$

where $\delta_t = \rho_t + \gamma V'_\theta(s_{t+1}) - V'_\theta(S_t)$, with the GAE parameter $\vartheta$ controls the bias-variance trade-off.

The critic network is trained by minimizing the mean squared error between the predicted state value and empirical return:

$$L^{\mathrm{CR}}(\theta') = \frac{1}{T}\sum_{t=0}^{T} \left(V'_\theta(S_t) - R_t\right)^2, \tag{43}$$

where $R_t$ represents the discounted cumulative reward.

The PPO algorithm employs a clipped surrogate objective for the actor network to ensure stable training and avoid large policy shifts. The clipped actor loss function is given by the following:

$$L^{\mathrm{CL}}(\theta) = \frac{1}{T}\sum_{t=0}^{T} \min\left(\rho_t(\theta)\hat{A}_t, \mathrm{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right), \tag{44}$$

where $\epsilon$ denotes a small hyperparameter that controls the trust region.

Convergence and stability of PPO are ensured, given the assumptions of Lipschitz continuity, bounded rewards, and a learning rate satisfying the Robbins–Monro conditions. The final total loss combines the actor and critic losses:

$$L^{\mathrm{tot}} = L^{\mathrm{CL}}(\theta) - c \cdot L^{\mathrm{CR}}(\theta'), \tag{45}$$

where $c$ denotes the scalar weight determining how much influence the critic loss has on the overall loss.

## *3.3. Training Algorithm*

Fig. 2 presents the PPO-based training framework for DT-driven data offloading in LEO-MEC-enabled IoT networks. The framework leverages the DT of the real physical system, enabling the intelligent data offloading algorithm to operate in a simulated digital environment and learn the optimal policy based on DT feedback. At each time slot $t$, the PPO agent receives the current system state $S_t$ from the DT as input to the actor network, which then determines the action to take according to algorithm 1. To encourage exploration of the action space, a normally distributed random noise $\sigma_t$ is added to the output action, $a_t = \pi_\theta(S_t) + \sigma_t$, where $\pi_\theta(S_t)$ is the policy output parameterized by $\theta$ and $\sigma_t \sim \mathcal{N}(0, \Sigma_t)$.

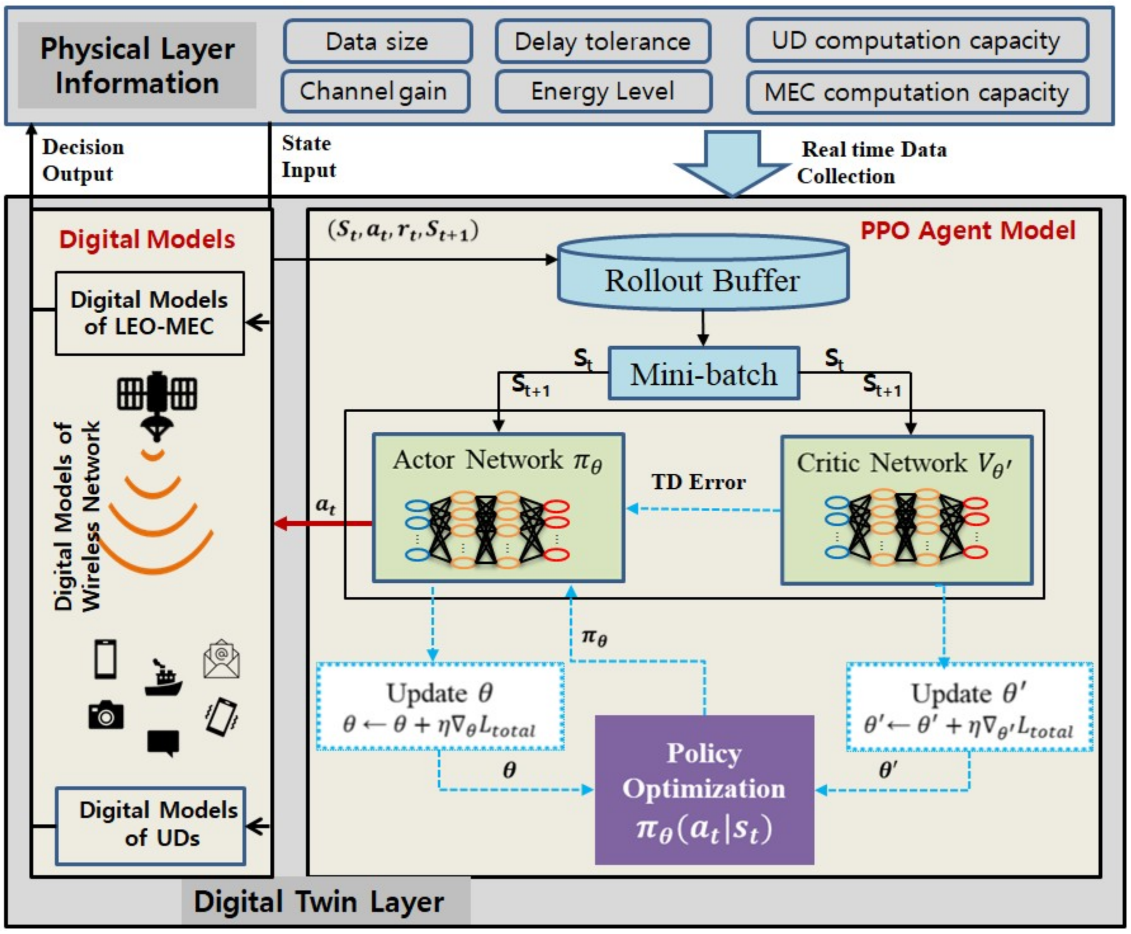
Physical Layer Information
Data size
Delay tolerance
UD computation capacity
Channel gain
Energy Level
MEC computation capacity
Decision Output
State Input
Real time Data Collection
Digital Models
$(S_t, a_t, r_t, S_{t+1})$
PPO Agent Model
Rollout Buffer
Digital Models of LEO-MEC
$S_t$
$S_{t+1}$
Mini-batch
Digital Models of Wireless Network
Actor Network $\pi_\theta$
TD Error
Critic Network $V_{\theta'}$
$a_t$
$\pi_\theta$
Update $\theta$
$\theta \leftarrow \theta + \eta\nabla_\theta L_{total}$
Update $\theta'$
$\theta' \leftarrow \theta' + \eta\nabla_{\theta'} L_{total}$
$\theta$
$\theta'$
Policy Optimization $\pi_\theta(a_t|s_t)$
Digital Models of UDs
Digital Twin Layer

**Fig. 2. PPO-based algorithm framework.**

The selected action is executed in the DT, which simulates the system's future evolution and returns the immediate reward $r_t$ along with the next state $S_{t+1}$. Using this state–transition–reward tuple $(S_t, a_t, r_t, S_{t+1})$, the PPO agent computes the TD error to update the parameters of both the actor and critic networks.

The PPO agent employs an actor–critic architecture and is trained using interactions stored in a temporary rollout buffer, which is cleared after each update to maintain the on-policy learning requirement. The actor network outputs binary offloading decisions, semantic compression ratios, MEC server assignments, and transmission power allocations, while the critic network evaluates the quality of these decisions. PPO iteratively refines the policy through minibatch sampling and the clipped surrogate objective, ensuring stable updates even under dynamic and uncertain LEO-MEC conditions. Once trained, the optimized policy is deployed in the physical system to enable intelligent, adaptive, and energy-efficient resource control.

The training objective is to learn a semantic-aware offloading policy that jointly optimizes semantic compression ratio, offloading decisions, MEC server selection and transmission power control to minimize DT synchronization delay while satisfying communication and computation constraints. Algorithm 1 summarizes the training process: in each iteration, the agent interacts with the DT environment to collect state–action–reward trajectories, computes GAE, and updates the actor and critic networks using the clipped surrogate objective. This approach supports joint optimization across the hybrid action space and guarantees stable policy improvement.

**Algorithm 1:** Proximal Policy Optimization-based Procedure

```
Input: Initial policy parameter $\theta$, value network parameter $\theta'$, learning rate $\eta$,
       clipping parameter $\epsilon$, discount factor $\gamma$, GAE parameter $\vartheta$, number of
       iterations $n$, trajectory length $T$
Output: Optimized policy $\pi_\theta$
1  for iteration $i = 1$ to $n$ do
2      Initialize rollout buffer $d \leftarrow \emptyset$;
3      for each episode do
4          Obtain state $S_t$;
5          Select actions $a_t = \pi_\theta(S_t) + \sigma_t$;
6          Execute $a_t$, observe reward $r_t$ and next state $S_{t+1}$;
7          Store $(S_t, a_t, r_t, S_{t+1}, \pi_\theta(a_t|S_t))$ in $D$;
8          Sample m random mini-batch of experiences from $D$;
9          Update the policy gradient using 40;
10         Compute the TD error using (41);
11         Compute the GAE advantage using (42);
12         Compute the return: $R_t = \hat{A}_t + V_{\theta'}(S_t)$;
13     Minimize the critic loss using (43);
14     Maximize the clipped surrogate objective using (44);
15     Compute the total loss using (45);
16     Update actor network parameter $\theta \leftarrow \theta + \eta \nabla_\theta L^{\mathrm{tot}}$;
17     Update critic network parameter $\theta' \leftarrow \theta' + \eta \nabla_{\theta'} L^{\mathrm{tot}}$;
18 return Optimized policy ($\pi_\theta$)
```

### *3.4. Complexity and Scalability Analysis*

We adopt a DRL workflow supported by DT: the policy is trained offline in the DT and only inference runs online on the physical network, similar to [48]. The DT continuously tracks environment changes and refreshes the policy through periodic fine-tuning. Let $T$ be the number of environment steps (trajectory length) collected per PPO iteration, $k$ the number of policy–update epochs, $D$ the mini–batch size (thus $m = T/D$ mini–batches per epoch), and $p_a, p_c$ the parameter counts of the actor and critic, respectively. The per–iteration time is dominated by: (i) data collection, $t$ forward passes of actor/critic, $\mathcal{O}\big(t(p_a + p_c)\big)$; and (ii) parameter updates, $k$ epochs over all mini–batches, $\mathcal{O}\big(kT(p_a + p_c)\big)$. Hence one PPO iteration is $\mathcal{O}\big((1+k)\,T\,(p_a + p_c)\big)$, and over $n$ iterations the training cost is $\mathcal{O}\big(n(1+k)T(p_a + p_c)\big)$. At deployment, each control decision requires a single actor forward pass, yielding online complexity $\mathcal{O}(p_a)$ per slot, which is negligible compared with radio transmission, propagation, and edge–compute latencies in LEO–MEC.

*Scalability.* With $\mathcal{U}$ UDs and $\mathcal{L}$ LEO-MEC servers, we use batched inference so the per–slot controller cost scales as $\mathcal{O}(\mathcal{U}\,p_a)$. Server selection is produced by a categorical output whose dimension grows linearly with $\mathcal{L}$, affecting only the final layer by $\mathcal{O}(\mathcal{L})$. The rollout buffer stores $T$ transitions per user, so memory scales as $\mathcal{O}(T\,\mathcal{U})$. Overall, the method scales linearly with the number of UDs and only mildly with the number of LEO-MEC servers, making it practical for large–scale LEO–MEC server deployments while keeping run–time decision latency small.

## 4. Experiments

### *4.1. Setup and Protocol*

We evaluate semantic offloading for DT synchronization in a representative LEO–MEC setting and intentionally keep the modeling choices close to real deployment conditions: short coverage windows, non-stationary radio links, and heterogeneous tasks. Our goal is to measure the end-to-end synchronization delay $\psi_u$ under realistic contention and to compare learning-based policies under the same information and budget.

*Topology and dynamics.* The constellation comprises $L \in \{1, \ldots, 9\}$ satellites in low Earth orbit at roughly 600 km. Each satellite carries one MEC server. This keeps compute locality realistic (satellite-level scheduling rather than an abstract "cloud") and limits per-pass contact to $\approx$200 s, which we replay as a time-varying visibility mask. One-way propagation delay averages $4 \pm 1$ ms; inter-satellite links have sub-millisecond latency with ample capacity so that the performance bottleneck remains the access link and the edge queue rather than the backhaul. These numbers are conservative but within the envelope of current LEO systems and ensure that improvements must come from better offloading and compression decisions, not from optimistic physics.

*Users and workloads.* We vary the number of UDs $U$ from 10 to 200 to traverse light-load to congestion regimes. Each device initiates synchronization tasks via a Poisson process; the sensing payload size $I_u$ is drawn in $[0.5, 20]$ MB to cover "thumbnail" to "raw frame" scales used in DT updates (fig. 3). To avoid overfitting to a single compute profile, we mix task classes: 50% light CNN inference, 30% object detection, and 20% segmentation. Each class maps to a per-task compute intensity $\vartheta$ and (implicitly) to different codec costs, so the agent must learn not only where to offload but also how aggressively to compress.

*Communication and compute models.* Per-user uplink bandwidth $b_{u,l}$ ranges from 0.5 to 2 MHz (default 1 MHz), with Rician fading (factor $K$=6 dB) to reflect dominant LoS with occasional fluctuations. User transmit power $p^{\text{tr}}_{u,l}$ is constrained to $[0, 0.2]$ W, which is typical for battery-powered radios and prevents the trivial solution of "powering through" congestion. On the compute side, user CPUs are capped at $f_u \leq 1$ GHz and satellite CPUs at $f^u_l \leq 10$ GHz with small Gaussian jitter to emulate scheduler noise and co-resident workloads. Together, these choices induce the right trade-offs: local execution becomes attractive for tiny payloads, whereas the edge wins as payloads and user counts grow—provided the policy chooses compression, power, and target server judiciously.

*Hybrid decision space and its semantics.* Actions are hybrid by construction. The discrete part comprises $\beta_{u,l} \in \{0, 1\}$ (local vs. offload) and $\alpha_{u,l} \in \{1, \ldots, L\}$ when offloading is chosen. The continuous part comprises $\mu_u \in [\mu^{\min}, 1]$ with $\mu^{\min}$=0.1 (to enforce a minimum fidelity) and $p^{\text{tr}}_{u,l} \in [0, 0.2]$ W. This factorization mirrors deployable stacks: $\beta_{u,l}/\alpha_{u,l}$ are routing/scheduling choices, while $(\mu_u, p_{u,l})$ are codec and PHY knobs. Evaluating learning algorithms on this joint space, rather than on decoupled subproblems, is essential to expose the coupling between radio, compute, and semantics.

*Objective and reward shaping.* Our primary metric is the end-to-end DT synchronization delay $\psi$ in seconds, including sensing/encode/transmit/queue/decode/compute/propagation. All RL agents optimize the same scalar reward

$$r[t] \; = \; -\,\psi_u[t] \; - \; \kappa \cdot \mathbb{I}\{\text{deadline, energy, or coverage violated}\},$$

where the penalty weight $\kappa$ is identical across methods. This shaping makes the learning objective strictly anti-delay while explicitly discouraging infeasible schedules (e.g., trying to offload outside coverage or exceeding the energy budget). In practice, the penalty term acts as a guardrail that stabilizes training when the environment drifts (handover, bursty contention).

*Baselines and fairness.* We include three non-learning baselines that anchor the task and disambiguate what "semantic" means in our setting.

- *Local execution* ("Local"): all tasks are executed on the device CPU without offloading.
- *Non-semantic offloading* ("Non-semantic"): a content-agnostic, conventional stack with no semantic compression (we fix $\mu_u$=1), classical link adaptation and queue-aware routing to the nearest visible MEC (Greedy min-delay under M/M/1 approximation), and standard rate/power control without semantics; see representative MEC/offloading heuristics [49, 50]. This baseline isolates the value of semantics by holding the communication/computation pipeline conventional.
- *Random offloading* ("Random"): a semantics-enabled but uninformed policy. At each decision epoch it samples uniformly from the feasible hybrid action set $\mathcal{A} = (\beta_{u,l}, \alpha_{u,l}, \mu_u, p_{u,l})$, thereby allowing semantic compression (variable $\mu_u$) and power control but without learning or optimization.

Beyond the three non-learning baseline, we evaluate four DRL semantic–offloading variants under the same observation interface and simulator:

- *DQN-SO (based on DQN)* [51]: we adapt Deep Q-Network to our setting by controlling only the discrete routing $(\beta_{u,l}, \alpha_{u,l})$; the continuous knobs $(\mu_u, p_{u,l})$ follow fixed heuristics, mirroring stacks that keep PHY/codec fixed.
- *A2C-SO (based on A2C)* [52]: factorized hybrid policy with a categorical head for $(\beta_{u,l}, \alpha_{u,l})$ and Gaussian heads for $(\mu_u, p_{u,l})$.
- *SAC-SO (based on SAC)* [53]: same hybrid parameterization as A2C-SO, but off-policy with entropy regularization and automatic temperature.
- *Proposed Method (PPO-SO)*: on-policy with a clipped surrogate objective, using the same factorized hybrid policy for $(\beta_{u,l}, \alpha_{u,l}, \mu_u, p_{u,l})$.

*Note.* The citations above refer to the algorithmic foundations (DQN/A2C/SAC/PPO), not to prior work on semantic offloading. Our "–SO" variants are our implementations of those algorithms in the hybrid action space of semantic offloading.

We intentionally keep architectures and budgets symmetric across DRL methods to ensure comparability: all use a two-layer MLP backbone (128 and 64 hidden units) for actor/critic, the same training budget (25k steps), discount $\gamma$=0.99, and mini-batch size 64. PPO-SO uses clipping $\epsilon$=0.2 and generalized advantage estimation (GAE) with $\vartheta$=0.95; SAC-SO uses automatic temperature tuning. For completeness, our semantic encoder/decoder follows the standard learned semantic communication paradigm [54] (instantiated within our simulator), but DRL algorithms only control how much semantics to apply via $\mu_u$, not the encoder weights.

*Training protocol and statistics.* Each condition is trained three times with different seeds; evaluation uses disjoint seeds and draws 100 episodes per condition to estimate means and standard deviations. We quote $p_{u,l}$-values from paired Wilcoxon tests with Holm correction when comparing PPO-SO to the strongest alternative (typically SAC-SO) to avoid inflating significance through multiple comparisons.

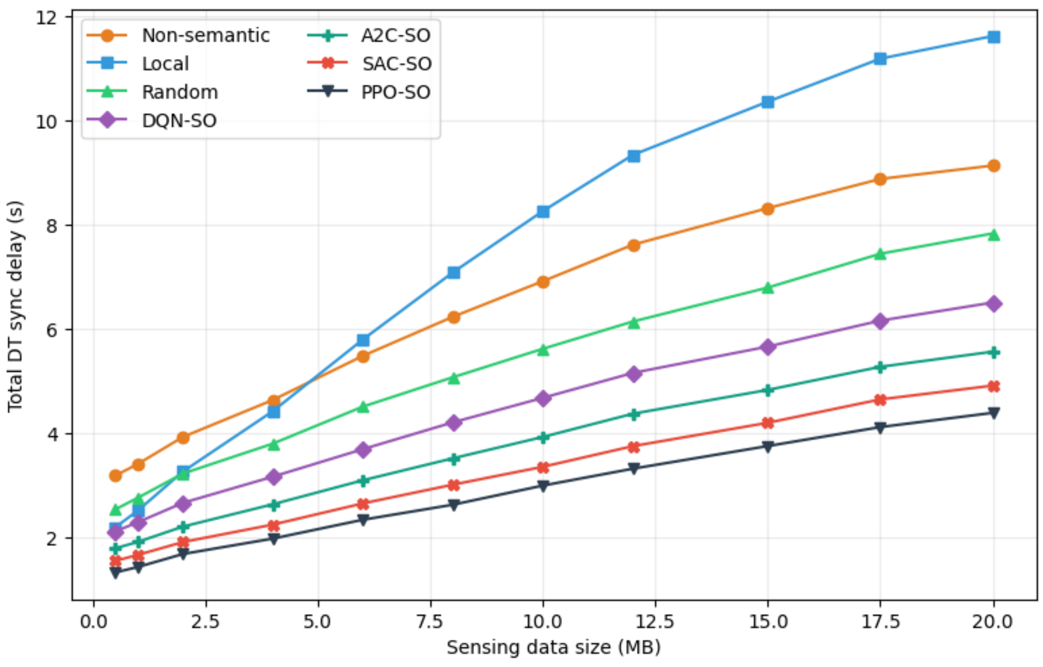

**Fig. 3. Total DT synchronization delay vs. sensing payload $I_u$ (MB).** All methods rise monotonically; a stable break-even appears near $I_u \approx 2$–3 MB where **PPO-SO** starts beating *Local*. At $I_u = 20$ MB, PPO-SO achieves 5.5±0.2 s, improving over Non-semantic by 26.6% and over Local by 45.0% ($p_{u,l}$<0.01).

### *4.2. Total DT Synchronization Delay Reduction*

Fig. 3 examines how the total DT synchronization delay $\psi_u$ scales with the sensing payload $I_u$. Across the entire range $I_u \in [0, 20]$ MB, all methods exhibit a monotone increase, which is expected as the transmission and queueing components become dominant for larger payloads. The salient feature is a consistent break-even around $I_u \approx 2$–3 MB at which PPO-SO begins to outperform Local execution. For tiny payloads ($\leq 1$ MB) Local remains competitive due to the absence of transmission overhead, but as the payload grows, PPO-SO jointly decreases the semantic ratio $\mu_u$ and steers the routing/power pair $(\alpha_{u,l}, p_{u,l})$ to less congested links, thereby curbing the very terms (airtime and queueing) that dominate the tail.

At $I_u = 20$ MB, the mean delays over three runs are $5.5 \pm 0.2$ s for PPO-SO, $6.1 \pm 0.2$ s for SAC-SO, $6.8 \pm 0.3$ s for A2C-SO, $7.9 \pm 0.4$ s for DQN-SO, $8.6 \pm 0.4$ s for Random, $7.5 \pm 0.3$ s for Non-semantic, and $10.0 \pm 0.5$ s for Local. Relative to the strongest non-learning baseline, PPO-SO reduces delay by 26.6% versus non-semantic and by 45.0% versus local; the gap to the strongest learning alternative, SAC-SO, is 9.8%. These effects are not cosmetic: beyond $I_u \geq 8$ MB the PPO–SAC separation is statistically significant ($p_{u,l}$<0.01; paired Wilcoxon with Holm correction) and widens with $I_u$, which is consistent with the mechanism—lowering $\mu_u$ shrinks airtime roughly linearly in $I_u$, while adaptive $(\alpha_{u,l}, p_{u,l})$ reduces queueing in a load-dependent manner. Fig. 4 complements this view by varying the number of users. As $U$ increases from 10 to 200, the curves acquire a visibly convex profile: queueing intensifies, service times broaden, and inferior policies steepen faster. Local can win in the extreme low-load corner, but it CPU-saturates quickly and is overtaken near $U \approx 40$–60, after which PPO-SO widens the margin by rebalancing server choice and semantic compression to the observed contention. At $U = 200$, the means are $12.1 \pm 0.5$ s (PPO-SO), $12.9 \pm 0.6$ s (SAC-SO), $13.8 \pm 0.6$ s (A2C-SO), $15.2 \pm 0.7$ s (DQN-SO), $16.3 \pm 0.8$ s (Random), $17.0 \pm 0.9$ s (Non-semantic), and $18.6 \pm 1.0$ s (Local). In the practically relevant high-load regime ($U \geq 120$), PPO-SO improves over SAC-SO by 5–8% and over Non-semantic by 25–35% ($p_{u,l}$<0.01), indicating that the benefit of coordinating $(\beta_{u,l}, \alpha_{u,l}, \mu_u, p_{u,l})$ grows precisely where congestion is the bottleneck. Taken together, fig. 3–4 show that the absolute ranking (PPO-SO $\prec$ SAC-SO $\prec$ A2C-SO $\prec$ DQN-SO $\prec$ Random $\prec$ Non-semantic $\prec$ Local) is stable, while the gaps widen with payload and load for reasons grounded in the model: semantics reduces airtime; routing and power reduce contention; and the joint policy learns to trade these levers as conditions shift.

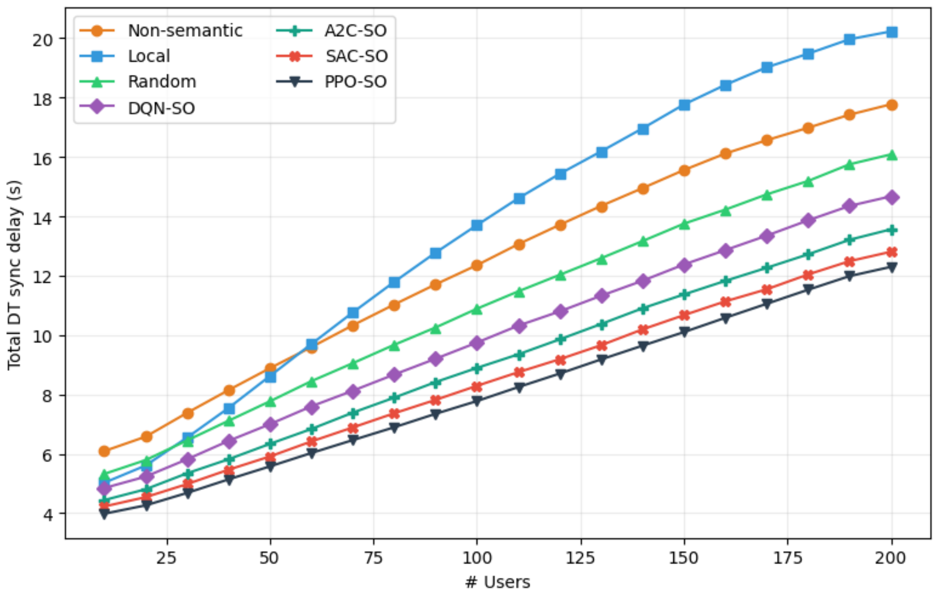

**Fig. 4. Total delay vs. number of users $U$.** Curves steepen with load; Local is competitive only at very low $U$ and is overtaken around $U \approx 40$–60. At $U = 200$, PPO-SO $12.1 \pm 0.5$ s vs. SAC-SO $12.9 \pm 0.6$ s and Non-semantic $17.0 \pm 0.9$ s; gaps are significant for $U \geq 120$ ($p<0.01$).

### *4.3. Effect of the Satellite Density on the Synchronization Delay*

Fig. 5 varies the constellation size from one to nine satellites to quantify how additional spatial diversity and coverage translate into end-to-end delay. All offloading methods benefit, but with diminishing returns: each added satellite enlarges the feasible set for the discrete choice $\alpha_{u,l}$, yet the marginal gain tapers as the policy has already skirted the dominant hotspots. The hierarchy among methods persists throughout the sweep—PPO-SO consistently leads SAC-SO, then A2C-SO and DQN-SO, with Random and Non-semantic trailing and Local nearly flat because it never offloads. The magnitude of the effect is material: for PPO-SO, the median delay drops from approximately 16.8 s (one satellite) to 11.3 s (nine satellites), a 33% reduction; the PPO–SAC gap widens modestly (about 0.3–0.5 s by $L{=}9$, $p_{u,l}{<}0.01$), reflecting that a stronger policy extracts more value from enlarged choice sets. Qualitatively, this is the same mechanism that governed fig. 3–4: when more routing options exist, policies that internalize link load and queue states in their action selection benefit disproportionately, whereas non-semantic and random strategies cannot reliably convert additional visibility into shorter waits.

### *4.4. Convergence and Stability*

To test robustness under non-stationarity, Fig. 6 injects a handover/congestion regime shift at training steps 18k–26k. All agents experience a drop in average reward (proportional to negative delay), but their trajectories differ in both depth and recovery. PPO-SO exhibits the smallest maximum drop (about 0.7 in arbitrary units) and the fastest return to its pre-shift plateau—within roughly 4 k steps to 95%—whereas SAC-SO drops by about 1.3 and requires $\sim$ 8 k steps to recover; A2C-SO drops by $\sim$ 1.0 and recovers in $\sim$ 9 k steps;

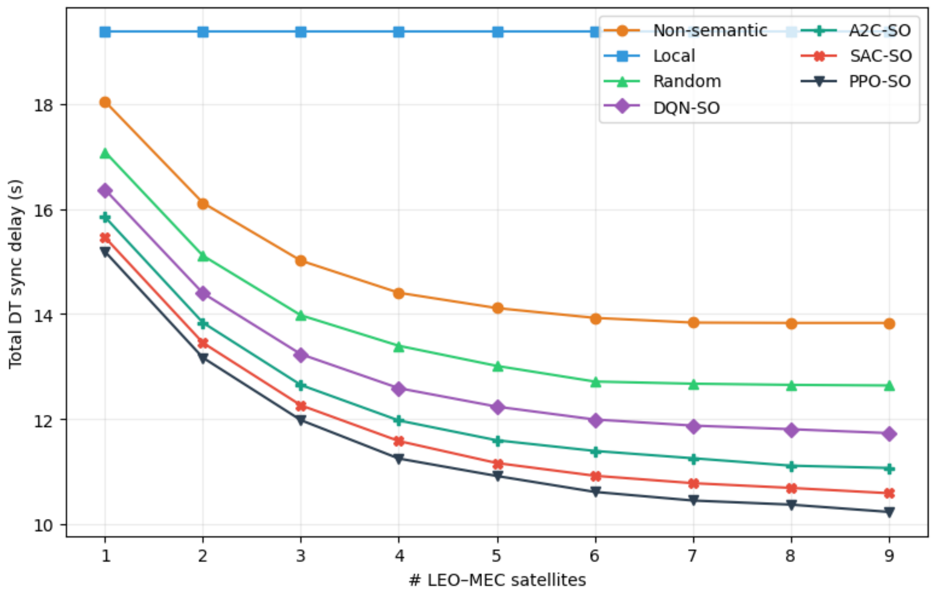

**Fig. 5. Effect of satellite density $L$ on total delay.** More satellites reduce delay with diminishing returns; **PPO-SO** benefits most (median 16.8→11.3 s from $L{=}1 \to 9$, $\approx 33\%$).

DQN-SO shows pronounced post-shift oscillations. The curves are uniformly shifted downward for display to avoid touching zero; shaded bands denote one standard deviation across three runs. The advantage of PPO-SO over SAC-SO in the post-shock window is statistically significant ($p_{u,l}{<}0.01$), and it aligns with the algorithmic design: clipped on-policy updates curb over-corrections when the data distribution moves, stabilizing the coupled decisions ($\beta_{u,l}, \alpha_{u,l}, \mu_u, p_{u,l}$) and speeding re-convergence once the new regime is learned. From a systems perspective, this matters exactly where LEO–MEC deployments are brittle—in the minutes around frequent handovers and bursty congestion—suggesting that the performance gains reported in fig. 3–5 are not only larger in steady state but also more durable when the environment shifts.

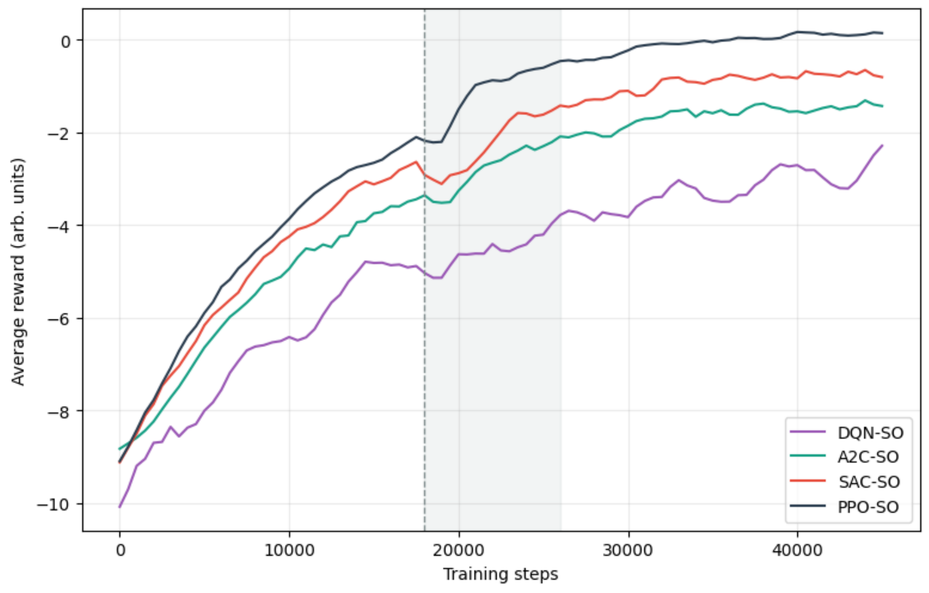

**Fig. 6. Training stability under a handover/congestion regime shift (18k–26k steps). PPO-SO** shows the smallest drop ($\Delta r \approx 0.7$) and fastest recovery (to 95% within ~4k steps); SAC-SO drops $\approx 1.3$ with ~8k-step recovery. Shaded bands indicate one std over three runs; traces are uniformly down-shifted for display.

### *4.5. Bandwidth Consumption under Semantic Compression*

Fig. 7 shows that all three stacks—Non-semantic ($\mu_u = 1$), Semantic (fixed $\mu_u$) with $\mu_u = 0.60$, and Semantic (optimized $\mu_u$) that gradually approaches $\mu_u \approx 0.35$ for large $I_u$—exhibit a monotone increase in bandwidth as $I_u$ grows, yet the separation between curves widens sharply in the high-payload regime where the numerator dominates. At the large-payload endpoint ($I_u = 32$ MB), Non-semantic requires on the order of $3.4 \times 10^2$ MHz under our settings, whereas Semantic (fixed) needs only about $1.1 \times 10^2$ MHz and Semantic (optimized) roughly $5.3 \times 10^1$ MHz; these correspond to reductions of about 67% and 84–85% relative to Non-semantic. Even at the opposite end ($I_u = 1$ MB), where absolute demands are modest, the ranking is consistent: Non-semantic sits near $1.7 \times 10^1$ MHz, while Semantic (fixed) and Semantic (optimized) consume roughly 4.8 and 3.8 MHz, respectively—still a three- to fourfold saving that materially eases scheduling within a sub-second slot.

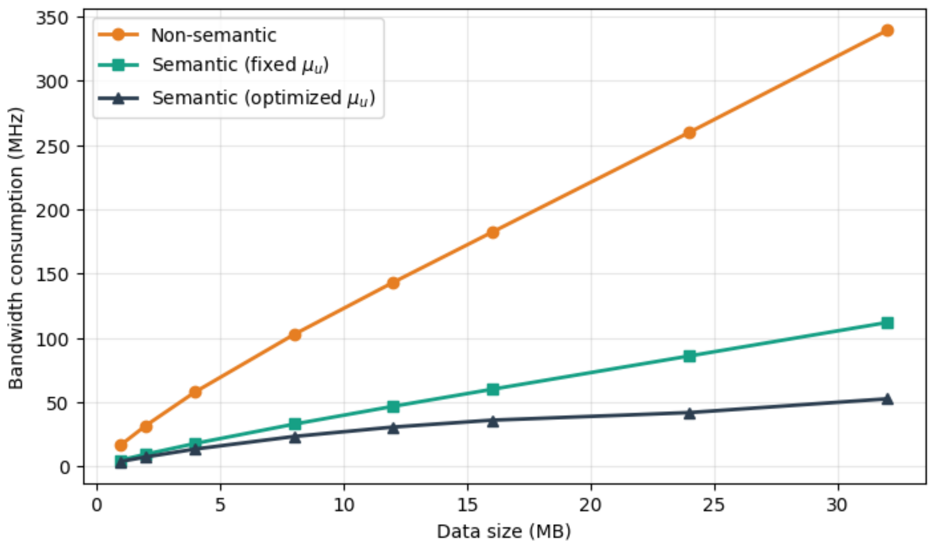

**Fig. 7. Required bandwidth vs. payload under conservative timing/efficiency** ($\Xi$=0.8 s, S-SE=1.6 bps/Hz).

## 5. Conclusion

This paper presents a novel DT-driven semantic-aware data offloading framework for LEO–MEC–enabled remote IoT networks. The joint control of offloading, server selection, semantic ratio, and transmit power was cast as an MDP with a hybrid action space and solved via PPO-SO, alongside fair DRL baselines (SAC-SO, A2C-SO, DQN-SO) and non-learning baselines (local, random, non-semantic). In realistic LEO settings with short coverage windows, nonstationary links, and heterogeneous workloads, PPO-SO reduced end-to-end DT synchronization delay by up to 45% vs. local, 26–35% vs. non-semantic, and 5–10% vs. SAC-SO at high load, with statistically significant gaps. These results indicate that combining DT feedback with semantic communication and learning-based control can deliver resilient, low-latency performance under the dynamics and constraints of satellite—ground integrated systems. The framework is directly applicable to 6G satellite–terrestrial use cases (e.g., maritime monitoring, disaster response, wide-area agriculture/environmental sensing, remote industrial analytics) where low-latency DT synchronization matters under tight energy and coverage constraints.

**Future work** (i) Constellation-level orchestration with ISL-aware load balancing (graph RL or distributed optimization) to coordinate tasks across satellites; (ii) freshness-aware

DT synchronization that couples semantic ratio and update frequency to value-of-update and channel dynamics; (iii) hierarchical integration with UAV-assisted integrated sensing, communication, and computing (ISCC), joining trajectory planning with semantic offloading; (iv) explore heavier multi-agent and model-based state-of-the-art (e.g., MAPPO/MADDPG for decentralized coordination; MPC/MBPO for planning): these approaches are orthogonal to our semantic-compression focus and are left as future work for constellation-level scheduling and tighter queue-aware control.

## References

[1] C. Ding, J.-B. Wang, H. Zhang, M. Lin, G. Y. Li, Joint optimization of transmission and computation resources for satellite and high altitude platform assisted edge computing, IEEE Transactions on Wireless Communications 21 (2) (2021) 1362–1377.

[2] N.-N. Dao, Q.-V. Pham, N. H. Tu, T. T. Thanh, V. N. Q. Bao, D. S. Lakew, S. Cho, Survey on aerial radio access networks: Toward a comprehensive 6g access infrastructure, IEEE Communications Surveys & Tutorials 23 (2) (2021) 1193–1225.

[3] D. S. Lakew, A.-T. Tran, N.-N. Dao, S. Cho, Intelligent self-optimization for task offloading in leo-mec-assisted energy-harvesting-uav systems, IEEE Transactions on Network Science and Engineering (2024).

[4] R. Schlegel, S. Kumar, E. Rosnes, A. G. i Amat, Privacy-preserving coded mobile edge computing for low-latency distributed inference, IEEE Journal on Selected Areas in Communications 40 (3) (2022) 788–799.

[5] W. J. Yun, D. Kwon, M. Choi, J. Kim, G. Caire, A. F. Molisch, Quality-aware deep reinforcement learning for streaming in infrastructure-assisted connected vehicles, IEEE Transactions on Vehicular Technology 71 (2) (2021) 2002–2017.

[6] Z. Song, Y. Liu, X. Sun, Joint task offloading and resource allocation for noma-enabled multi-access mobile edge computing, IEEE Transactions on Communications 69 (3) (2020) 1548–1564.

[7] Q. Lan, D. Wen, Z. Zhang, Q. Zeng, X. Chen, P. Popovski, K. Huang, What is semantic communication? a view on conveying meaning in the era of machine intelligence, Journal of Communications and Information Networks 6 (4) (2021) 336–371.

[8] Z. Weng, Z. Qin, Semantic communication systems for speech transmission, IEEE Journal on Selected Areas in Communications 39 (8) (2021) 2434–2444.

[9] H. Xie, Z. Qin, G. Y. Li, B.-H. Juang, Deep learning enabled semantic communication systems, IEEE transactions on signal processing 69 (2021) 2663–2675.

[10] D. B. Kurka, D. Gündüz, Deep joint source-channel coding of images with feedback, in: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020, pp. 5235–5239.

[11] P. Jiang, C.-K. Wen, S. Jin, G. Y. Li, Wireless semantic communications for video conferencing, IEEE Journal on Selected Areas in Communications 41 (1) (2022) 230–244.

[12] L. Wang, W. Wu, F. Tian, H. Hu, Intelligent resource allocation for uav-enabled spectrum sharing semantic communication networks, in: 2023 IEEE 23rd International Conference on Communication Technology (ICCT), IEEE, 2023, pp. 1359–1363.

[13] F. Jiang, Y. Peng, L. Dong, K. Wang, K. Yang, C. Pan, X. You, Large ai model-based semantic communications, IEEE Wireless Communications 31 (3) (2024) 68–75.

[14] G. Zheng, Q. Ni, K. Navaie, H. Pervaiz, Semantic communication in satellite-borne edge cloud network for computation offloading, IEEE Journal on Selected Areas in Communications (2024).

[15] B. Li, H. Cai, L. Liu, Z. Fei, Delay-aware digital twin synchronization in mobile edge networks with semantic communications, IEEE Transactions on Vehicular Technology (2025).

[16] T. Liu, L. Tang, W. Wang, Q. Chen, X. Zeng, Digital-twin-assisted task offloading based on edge collaboration in the digital twin edge network, IEEE Internet of Things Journal 9 (2) (2021) 1427–1444.

[17] Y. Wu, K. Zhang, Y. Zhang, Digital twin networks: A survey, IEEE Internet of Things Journal 8 (18) (2021) 13789–13804.

[18] T. Liu, L. Tang, W. Wang, Q. Chen, X. Zeng, Digital-twin-assisted task offloading based on edge collaboration in the digital twin edge network, IEEE Internet of Things Journal 9 (2) (2022) 1427–1444.

[19] Z. Ji, Z. Qin, X. Tao, Z. Han, Resource optimization for semantic-aware networks with task offloading, IEEE Transactions on Wireless Communications (2024).

[20] L. Yan, Z. Qin, R. Zhang, Y. Li, G. Y. Li, Resource allocation for text semantic communications, IEEE Wireless Communications Letters 11 (7) (2022) 1394–1398.

[21] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, D. O. Wu, Energy-optimal mobile cloud computing under stochastic wireless channel, IEEE Transactions on Wireless Communications 12 (9) (2013) 4569–4581.

[22] T. Q. Dinh, J. Tang, Q. D. La, T. Q. Quek, Offloading in mobile edge computing: Task allocation and computational frequency scaling, IEEE Transactions on Communications 65 (8) (2017) 3571–3584.

[23] K. Guo, M. Sheng, T. Q. Quek, Z. Qiu, Task offloading and scheduling in fog ran: A parallel communication and computation perspective, IEEE Wireless Communications Letters 9 (2) (2019) 215–218.

[24] G. Cui, X. Li, L. Xu, W. Wang, Latency and energy optimization for mec enhanced sat-iot networks, IEEE Access 8 (2020) 55915–55926.

[25] B. Cao, J. Zhang, X. Liu, Z. Sun, W. Cao, R. M. Nowak, Z. Lv, Edge–cloud resource scheduling in space–air–ground-integrated networks for internet of vehicles, IEEE Internet of Things Journal 9 (8) (2021) 5765–5772.

[26] X. Zhu, C. Jiang, Delay optimization for cooperative multi-tier computing in integrated satellite-terrestrial networks, IEEE Journal on Selected Areas in Communications 41 (2) (2022) 366–380.

[27] Y. Liu, L. Jiang, Q. Qi, S. Xie, Energy-efficient space–air–ground integrated edge computing for internet of remote things: A federated drl approach, IEEE Internet of Things Journal 10 (6) (2022) 4845–4856.

[28] D. Won, G. Woraphonbenjakul, A. B. Wondmagegn, A. T. Tran, D. Lee, D. S. Lakew, S. Cho, Resource management, security, and privacy issues in semantic communications: A survey, IEEE Communications Surveys & Tutorials (2024).

[29] Y. Zheng, T. Zhang, R. Huang, Y. Wang, Computing offloading and semantic compression for intelligent computing tasks in mec systems, in: 2023 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2023, pp. 1–6.

[30] N. T. Hoa, C. T. T. Hai, H. Le Hung, N. C. Luong, D. Niyato, Joint edge computing and semantic communication in uav-enabled networks, IEEE Communications Letters (2024).

[31] T. Do-Duy, D. Van Huynh, O. A. Dobre, B. Canberk, T. Q. Duong, Digital twin-aided intelligent offloading with edge selection in mobile edge computing, IEEE Wireless Communications Letters 11 (4) (2022) 806–810.

[32] Z. Ji, S. Wu, C. Jiang, Cooperative multi-agent deep reinforcement learning for computation offloading in digital twin satellite edge networks, IEEE Journal on Selected Areas in Communications 41 (11) (2023) 3414–3429.

[33] A. Masaracchia, V.-L. Nguyen, D. B. da Costa, E. Ak, B. Canberk, V. Sharma, T. Q. Duong, Toward 6g-enabled urllcs: Digital twin, open ran, and semantic communications, IEEE Communications Magazine 9 (1) (2025) 13–20.

[34] G. Shi, Y. Xiao, Y. Li, X. Xie, From semantic communication to semantic-aware networking: Model, architecture, and open problems, IEEE Communications Magazine 59 (8) (2021) 44–50.

[35] H. Xie, Z. Qin, A lite distributed semantic communication system for internet of things, IEEE Journal on Selected Areas in Communications 39 (1) (2020) 142–153.

[36] Z. Liang, H. Chen, Y. Liu, F. Chen, Data sensing and offloading in edge computing networks: Tdma or noma?, IEEE Transactions on Wireless Communications 21 (6) (2021) 4497–4508.

[37] W. Sun, S. Lei, L. Wang, Z. Liu, Y. Zhang, Adaptive federated learning and digital twin for industrial internet of things, IEEE Transactions on Industrial Informatics 17 (8) (2020) 5605–5614.

[38] B. Li, Y. Liu, L. Tan, H. Pan, Y. Zhang, Digital twin assisted task offloading for aerial edge computing and networks, IEEE Transactions on Vehicular Technology 71 (10) (2022) 10863–10877.

[39] H. Saadat, A. Albaseer, M. Abdallah, A. Mohamed, A. Erbad, Energy-aware service offloading for semantic communications in wireless networks, in: ICC 2024-IEEE International Conference on Communications, IEEE, 2024, pp. 5467–5472.

[40] H. Guo, X. Zhou, J. Wang, J. Liu, A. Benslimane, Intelligent task offloading and resource allocation in digital twin based aerial computing networks, IEEE Journal on Selected Areas in Communications 41 (10) (2023) 3095–3110.

[41] J. Tang, J. Nie, J. Bai, J. Xu, S. Li, Y. Zhang, Y. Yuan, Uav-assisted digital twin synchronization with tiny machine learning-based semantic communications, IEEE Internet of Things Journal (2024).

[42] Y. Pochet, L. A. Wolsey, Production planning by mixed integer programming, Vol. 149, Springer, 2006.

[43] D. S. Lakew, A.-T. Tran, N.-N. Dao, S. Cho, Intelligent offloading and resource allocation in heterogeneous aerial access iot networks, IEEE Internet of Things Journal 10 (7) (2022) 5704–5718.

[44] S. Park, C. Park, J. Kim, Learning-based cooperative mobility control for autonomous drone-delivery, IEEE Transactions on Vehicular Technology 73 (4) (2023) 4870–4885.

[45] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, X. Chen, Convergence of edge computing and deep learning: A comprehensive survey, IEEE communications surveys & tutorials 22 (2) (2020) 869–904.

[46] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347 (2017).

[47] X. Chen, D. Feng, W. Jiang, Q. Luo, G. Chen, Y. Sun, Online multi-task offloading for semantic-aware edge computing systems, arXiv preprint arXiv:2407.11018 (2024).

[48] R. Dong, C. She, W. Hardjawana, Y. Li, B. Vucetic, Deep learning for hybrid 5g services in mobile edge computing systems: Learn from a digital twin, IEEE Transactions on Wireless Communications 18 (10) (2019) 4692–4707.

[49] T. X. Tran, D. Pompili, Joint task offloading and resource allocation for multi-server mobile-edge computing networks, IEEE Transactions on Vehicular Technology 68 (1) (2018) 856–868.

[50] K. Wang, T. Y. Chai, W.-C. Wong, Routing, power control and rate adaptation: A q-learning-based cross-layer design, Computer Networks 102 (2016) 20–37.

[51] M. Xue, H. Wu, G. Peng, K. Wolter, Ddpqn: An efficient dnn offloading strategy in local-edge-cloud collaborative environments, IEEE Transactions on Services Computing 15 (2) (2021) 640–655.

[52] K. Li, W. Ni, X. Yuan, A. Noor, A. Jamalipour, Deep-graph-based reinforcement learning for joint cruise control and task offloading for aerial edge internet of things (edgeiot), IEEE Internet of Things Journal 9 (21) (2022) 21676–21686.

[53] C. Sun, X. Wu, X. Li, Q. Fan, J. Wen, V. C. Leung, Cooperative computation offloading for multi-access edge computing in 6g mobile networks via soft actor critic, IEEE Transactions on Network Science and Engineering 11 (6) (2021) 5601–5614.

[54] J. Xie, Z. Qin, Y. Shi, Y. Wang, H. Lu, T. Zhang, Deepsc: Deep learning-enabled semantic communication systems, IEEE Transactions on Signal Processing 69 (2021) 2663–2678.