

Distributed Multimodal LLM Inference in 6G NTN: A Digital Twin and Quantum Reservoir Computing-aided DRL Approach

Sasinda C. Prabhashana, *Student Member, IEEE*, Berk Canberk, *Senior Member, IEEE*, Hans-Jürgen Zepernick, *Senior Member, IEEE*, Dac-Binh Ha, Simon L. Cotton, *Fellow, IEEE*, Octavia A. Dobre, *Fellow, IEEE*, and Trung Q. Duong, *Fellow, IEEE*

Abstract—In this paper, we propose a novel digital-twin (DT)-enabled non-terrestrial network (NTN) that supports distributed multimodal large language model (LLM) inference across multiple satellites. The system features two low Earth orbit (LEO) satellites equipped with edge-computing capabilities, where one operates at a higher LEO altitude than the other. A transformer model is deployed in a distributed manner between these two satellites. The encoder and fusion stages operate on the lower-altitude LEO satellite, whereas the decoder stages are processed on the higher-altitude LEO satellite. At the base stations, each user generates a multimodal prompt that contains text and image inputs. The text is tokenized using generative pre-trained transformer (GPT)-style tokenizer (*o200k_harmony* vocabulary), while the image is embedded locally through a vision encoder to reduce communication overhead. Moreover, a DT is integrated to model the system dynamics and allocate resources across satellites for effective computation. Accordingly, we formulate a nonlinear programming problem that jointly optimizes uplink bandwidth, and power, as well as computational resources across satellites. The formulated optimization problem aims to minimize the total weighted system utility cost, which incorporates the energy and latency constraints, while maintaining reliable

multimodal LLM inference performance. To address this complex problem, we propose two learning frameworks, namely echo state network-enhanced deep deterministic policy gradient (ESN-Enhanced DDPG) and quantum reservoir computing-enhanced deep deterministic policy gradient (QRC-Enhanced DDPG). Simulation results show that the proposed methods outperform the baseline DDPG framework. In addition, QRC-Enhanced DDPG consistently provides the best overall performance and stronger robustness. The proposed algorithms achieve higher cumulative rewards, faster convergence, and lower weighted system cost.

Index Terms—Deep reinforcement learning, non-terrestrial network (NTN), Multimodal large language model, Quantum reservoir computing.

I. INTRODUCTION

THE evolution of wireless communications is driving the transition from the fifth generation (5G) to the sixth generation (6G) networks. The demand for data services keeps rising, while current 5G systems face major challenges [1]. These challenges become particularly evident for applications that require ultra-high-speed transmission, massive connectivity, and intelligent automation. Therefore, 6G is expected to overcome these barriers by enabling seamless integration between the physical and digital worlds. Thus, 6G will not only provide such advanced capabilities but also serve as the foundation for an intelligent, immersive, and sustainable digital ecosystem [2]. One of the most significant services in this evolution is artificial intelligence (AI)-based services. In particular, these services are powered by generative and multimodal large language models (LLMs). Even now, generative AI-based services are consuming a growing share of network and computing resources [3]. These include real-time text, image, audio, and video generation, as well as large-scale model training, virtual assistants, digital twins, and metaverse applications. Such services rely on continuous data exchange and require ultra-fast transmission, reliable connectivity, and strong edge-cloud cooperation. Accordingly, 6G will play a vital role in supporting AI-driven and LLM-based applications ensuring real-time intelligence, adaptability, and seamless interaction across intelligent environments [3].

LLMs learn patterns from massive datasets that include text, images, and other data types. They use billions or even trillions of parameters to understand context and generate responses. While this training builds strong intelligence, it also needs large clusters of graphics processing units (GPUs) and high

S. C. Prabhashana and O. A. Dobre are with the Faculty of Engineering and Applied Science, Memorial University of Newfoundland, St. John's, Canada (e-mail: cwellhengodag@mun.ca).

B. Canberk is with the School of Computing Engineering and Built Environment, Edinburgh Napier University, Edinburgh EH10 5DT, U.K., (e-mail: b.canberk@napier.ac.uk).

H.-J. Zepernick is with the Blekinge Institute of Technology, Sweden (e-mail: hans-jurgen.zepernick@bth.se).

D.-B. Ha is with Faculty of Electrical-Electronic Engineering, Duy Tan University, Da Nang 550000, Vietnam (e-mail: hadacbinh@duytan.edu.vn).

S. L. Cotton is with the Centre for Wireless Innovation (CWI), School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, BT3 9DT, Belfast, U.K. (email: simon.cotton@qub.ac.uk).

T. Q. Duong is with the Faculty of Engineering and Applied Science, Memorial University, St. John's, NL A1C 5S7, Canada, and also with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, BT7 1NN Belfast, U.K. (e-mail: tduong@mun.ca).

The work of Dac-Binh Ha was supported by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under Grant 102.04-2025.72. The work of Simon L. Cotton was funded in part by the U.K. Engineering and Physical Sciences Research Council (EPSRC) through the EPSRC Hub on All Spectrum Connectivity under Grant EP/X040569/1 and Grant EP/Y037197/1. The work of Octavia A. Dobre was supported in part Canada Research Chairs Program CRC-2022-00187 and by the Natural Sciences and Engineering Research Council of Canada (NSERC), Discovery Program, Grant RGPIN-2026-04392. The work of Trung Q. Duong was supported in part by the Canada Excellence Research Chair (CERC) Program CERC-2022-00109, in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant Program RGPIN-2025-04941, in part by the NSERC CREATE program (Grant number 596205-2025), and in part by the John R. Evans Leaders Fund (JELF) from Canada Foundation for Innovation (CFI).

memory bandwidth. As a result, the data centers that conduct these training jobs consume massive power for computation and cooling. The total energy use increases sharply with model size and training duration [4]. In particular, multimodal LLMs require significant amounts of power as they process large volumes of image and video data during both training and inference stages [5]. This growing energy demand has become a serious concern, as generative AI services continue to expand globally. Thus, energy-efficient AI training and inference have become a key challenge for 6G-era networks, where intelligent resource allocation and green computing strategies aim to balance performance with sustainability.

In addition, the next wave of 6G development focuses on global connectivity through non-terrestrial networks (NTNs). Hence, satellite communication plays a key role in extending network coverage beyond terrestrial limits. Low Earth orbit (LEO) satellites offer wide connectivity while reducing latency compared to traditional geostationary Earth orbit and medium Earth orbit satellite systems [6]. They support real-time and mission-critical services in remote or underserved regions. Furthermore, this ensures continuous communication when ground networks are unavailable [7].

Modern LEO systems integrate edge computing capabilities, which allow data to be processed directly on board instead of sending everything to distant cloud servers. Thus, integrating AI-based services in satellites marks a key step toward intelligent and autonomous space communication systems. This allows terrestrial users to access AI services from space instead of relying only on terrestrial or cloud infrastructure [8]. Therefore, this satellite edge computing approach reduces backhaul congestion and lowers end-to-end delay. Recently, quantum key distribution (QKD) emerges as a promising solution for enabling secure communication between satellites. It allows encryption keys to be shared through quantum states which offers theoretically unbreakable security against eavesdropping. This capability makes QKD a potential foundation for building trusted inter-satellite links in future networks [9]. Digital twin (DT) technology offers a powerful tool for real-time monitoring and intelligent control. It mirrors the behavior of physical systems to predict performance, detect faults, and optimize operations [10]. Thus, integrating DT technology into NTNs supports intelligent network management and sustainable network operation [11].

However, managing such resources in 6G-enhanced NTNs remains a major challenge. To overcome this challenge, deep reinforcement learning (DRL) emerges as a powerful paradigm for intelligent network control [12]. It enables adaptive and real-time optimization in complex and time-varying environments. While traditional optimization relies on predefined models, DRL learns optimal strategies through continuous interaction with the environment. Furthermore, integrating reservoir computing with DRL can significantly improve the adaptability and efficiency of DRL agents [13]. In addition, quantum computing offers a new approach for accelerating and enhancing DRL. Quantum circuits use superposition and entanglement to create nonlinear feature representations for

complex state-action relationships. [11], [14].

Despite recent progress, the practical use of quantum-enhanced DRL remains restricted by the limits of current quantum hardware [15]. Present systems face issues such as qubit noise, limited coherence time, and scalability, which reduce performance in real-world conditions [11], [16]. Most existing quantum DRL studies rely on shallow circuit designs that cannot fully exploit temporal correlations. Furthermore, integrating quantum computing (QC) with reservoir computing (RC) offers a stronger foundation to overcome these gaps. The quantum reservoir naturally captures temporal dependencies and nonlinear transitions through its internal quantum dynamics, while the DRL agent learns optimal policies from these enhanced feature spaces [17]. Therefore, it is essential to explore the integration of quantum reservoir computing (QRC) with DRL as a new pathway for AI-based intelligent network resource optimization. This direction remains largely underexplored and holds strong potential to address the scalability and adaptability limitations.

A. Related Works

The rapid expansion of generative AI and multimodal LLM services has created new challenges for network resource allocation. The increasing model size and computational demands of LLMs and large vision models require efficient task distribution, memory management, and joint optimization of computation and communication resources. To address these challenges, several recent works have explored offloading strategies, model scheduling, and collaborative cloud-edge processing [4], [5], [18]–[21]. For instance, an LLM-based task offloading and resource allocation architecture for multi-satellite edge computing networks was proposed in [22]. Here, the authors applied an LLM-driven optimizer to manage offloading, power control, and computation allocation. The proposed framework demonstrated that LLMs can improve convergence and generalization for large-scale optimization in satellite networks. And also, an adaptive LLM inference system was presented in [18]. This proposed method employs the TightLLM design. This design specifically addresses the varying transfer overheads that occur during the decoding process. And also, it updates the offloading policy dynamically in response to these changes. And also, it uses key-value (KV) cache recomputation and model-weight slicing, which help improve inference throughput on devices with limited GPU memory.

Similarly, a large-scale LLM training strategy was proposed in [4]. Here, the tensor, pipeline, and data parallelism were combined to improve GPU-cluster efficiency. This method demonstrated that careful coordination of parallelism can support trillion-parameter model training within practical time limits. In [19], a collaborative mobile-edge training framework was introduced. In this framework, mobile devices fine-tuned the shallow layers of LLMs while edge servers trained the deeper layers. Here, the authors proposed an iterative rank penalization method to improve user-server association. In addition, a joint model-inferencing and task-offloading solution for

large vision model services was proposed in [5]. It combined a multi-agent proximal policy optimization (PPO) approach for real-time decisions with an optimization of computation and communication resources.

The ambition of 6G-native AI was discussed in [20]. There, the authors analyzed a foundation-model-driven cloud edge end architecture. The framework highlighted how pre-trained multimodal models can coordinate network functions, improve orchestration, and enable intelligent collaboration across devices and agents. On the other hand, in [21], an active-inference-based offloading strategy was proposed for 6G symbiotic Internet of things (IoT). This method replaced reward-driven DRL with inference-driven decision making which enables faster adaptation to dynamic workloads and uncertain network environments. Collectively, these works highlight steady progress in supporting generative AI and multimodal workloads across satellite networks, edge systems, and emerging 6G network architectures.

Moreover, RC, particularly echo state networks (ESNs), has been widely used in wireless networks due to its efficiency in handling temporal data [13], [23]–[26]. In [23], a RC-aided dynamic spectrum access method was introduced for distributed wireless environments. The authors used an ESN to capture temporal spectrum patterns and improve decision-making under partial observability. This approach reduced collisions and achieved better convergence compared with traditional Q -learning models. Furthermore, a fully spiking RC model for multi-agent dynamic spectrum access was then presented in [24]. Here, a liquid state machine-based reservoir with homeostatic regulation was used to stabilize the internal dynamics. This design improved energy efficiency while maintaining reliable spectrum sharing performance. In addition, this work was further extended for analyzing the effect of liquid state machine-based reservoir computing in the energy efficient spectrum access [25]. Here, spiking RC was applied to manage both temporal and spatial variations in channel availability. The study showed clear gains in spectrum utilization and energy savings when compared with DRL baselines.

In [13], RC was extended to higher-layer network control through a deep echo state Q -network for trust sharing in Internet of vehicles. The ESN successfully modeled sequential trust updates and improved decision accuracy over long short term memory (LSTM) networks and DRL. Additionally, RC supported handover optimization in wireless networks. A PPO-based handover controller was proposed in [26], where RC processed sequential user and cell measurements with low training overhead. This design improved handover stability in fast changing open radio access networks. In contrast, these works demonstrate that RC is effective for handling temporal correlations, partial observability, and energy constraints in wireless networks.

Recently, QC has become a promising direction for advancing intelligent wireless systems. It offers new capabilities through quantum superposition, entanglement, and high-dimensional state representations. These properties allow quantum models to process complex temporal patterns and

nonlinear relationships more efficiently than many classical approaches [11]. In [16], QC was combined with recurrent DRL for uncrewed aerial vehicle path planning. The authors introduced quantum-based deep deterministic policy gradient (Q-DDPG) and recurrent Q-DDPG models, where quantum circuits were used to encode states into a higher-dimensional feature space. This design improved convergence and reduced training complexity compared with classical DDPG methods. Moreover, this study shows that quantum-enhanced recurrent models can better handle the temporal dynamics in wireless networks. Hence, the integration of quantum processing with learning models opens new possibilities for efficient temporal modeling in communication systems [27]. QRC builds on this idea by using a quantum system as the reservoir layer while keeping only the readout trainable. The QRC provides a high-dimensional state space through superposition and entanglement, which enhances the ability to capture nonlinear and time-varying patterns [28]. Accordingly, in [17], an efficient QRC-based DRL method was proposed. Here, quantum LSTM was used as an untrained reservoir, and only the surrounding classical layers were optimized. This approach achieved performance comparable to fully trained quantum recurrent models while reducing training cost and stabilizing learning on sequential DRL benchmarks.

B. Motivation and Contributions

The rapid growth of generative AI and multimodal LLM services places huge demands on future 6G networks. These services require continuous exchange of high-dimensional data representations and rely on ultra-fast, energy-efficient inference pipelines [3], [4], [18], [22]. However, current terrestrial networks struggle to meet these requirements due to limited coverage, constrained edge resources, and escalating power consumption during LLM processing. Consequently, 6G-enhanced NTN are emerging as a critical architecture for providing resilient global connectivity and supporting distributed AI workloads.

Furthermore, modern LEO satellites now integrate edge computing capabilities which can enable the execution of transformer-based inference stages directly in space. This shift reduces backhaul congestion and enhances real-time intelligence for remote and underserved regions [7], [12]. However, managing these heterogeneous resources becomes increasingly challenging as the nonlinear coupling between communication, processing, and energy intensifies in 6G-enabled NTNs. This coupling creates a complex decision space where conventional optimization methods and classical DRL often struggle to remain stable or converge efficiently. Consequently, RC and QRC combined with DRL emerge as promising solutions, as they provide stronger temporal modeling capabilities and enable more stable and efficient policy learning for dynamic 6G NTN environments [13], [23].

Inspired by the current advancements, this paper proposes an energy-efficient and latency-aware framework for distributed multimodal LLM inference in 6G-enabled NTNs. We develop

a system model that considers split multimodal LLM inference across two LEO satellites, DT-based processing-rate estimation, QKD refresh overhead, and satellite mobility. We then formulate a constrained nonlinear programming problem to minimize the total weighted system cost. This cost incorporates the latency and energy consumption of multimodal LLM tasks. The problem jointly optimizes bandwidth allocation, BS uplink transmit power, and satellite processing rates. To solve this problem, we propose two methods, namely ESN-Enhanced DDPG and QRC-Enhanced DDPG. To the best of our knowledge, this is the first work to apply QRC-Enhanced DDPG to resource allocation for distributed multimodal LLM inference in 6G-enabled NTN. The main contributions of this work are summarized as follows:

- We develop a system model that integrates distributed transformer-based multimodal LLM inference across two satellites. The model incorporates satellite mobility and coverage constraints, Ku-band multi-link channel modeling, DT-based compute rate estimation, inter-satellite security, solar-powered energy budgets, and multimodal prompt preprocessing using the tokenized text (*o200k_harmony*) [29] and CLIP ViT-B/32-based visual embeddings [30].
- We formulate a nonlinear programming (NLP) problem to jointly optimize the bandwidth allocation, BS uplink transmit power, and satellite processing-rate allocation. The objective is to minimize a weighted system cost that includes the total task latency and energy consumption. The optimization problem is subject to resource-allocation limits such as satellite coverage-window constraints, task QoS requirements, effective processing-rate constraints that include DT deviations, QKD refresh-delay constraints, and solar-energy limits.
- We propose two learning frameworks, namely echo state network-enhanced deep deterministic policy gradient (ESN-Enhanced DDPG) and quantum reservoir computing-enhanced deep deterministic policy gradient (QRC-Enhanced DDPG). The ESN-based critic uses a fixed classical reservoir for temporal feature extraction, whereas the QRC-based critic uses higher-order quantum encoding and fixed quantum reservoir layers for value-function learning.
- We conduct numerical evaluations under realistic 6G NTN settings. The results show that both ESN-enhanced and QRC-Enhanced DDPG significantly reduce the weighted system cost, latency, and energy consumption compared with conventional DRL. Moreover, the QRC-Enhanced DDPG achieves the best performance, offering faster convergence and more efficient resource allocation than the ESN-Enhanced DDPG.

C. Paper Structure and Notations

The structure of this paper is organized as follows. Section II describes the system model and the problem formulation, including satellite mobility, channel modeling, distributed multimodal LLM inference pipeline model, QKD overhead, and

the formulation of the NLP-based optimization problem. Section III presents the DRL framework, detailing its constrained MDP formulation and the integration of classical reservoir computing and quantum reservoir computing. Section IV evaluates the effectiveness of our proposed approach and presents comprehensive simulation results and a detailed performance analysis. Finally, Section V concludes the paper, summarizing our key findings and discussing potential avenues for future research.

Notations: Throughout this paper, scalars, vectors, and matrices are distinguished by lightface x , boldface lowercase $\mathbf{x}$, and boldface uppercase $\mathbf{X}$ letters, respectively. The symbols $(\cdot)^T$ and $(\cdot)^H$ denote the transpose and complex-conjugate transpose. We use $\text{diag}(\cdot)$ to create a diagonal matrix from a vector and $|\mathbf{x}|$ for the length of a vector $\mathbf{x}$. The set of complex numbers is $\mathbb{C}$, and $\otimes$ denotes the tensor product. A variable $x_{m,r}(t)$ represents a value at time t associated with the m -th transmitter and r -th receiver. Furthermore, $\mathcal{CN}(\mu, \sigma^2)$ refer to complex Gaussian distributions with mean μ and variance σ^2 .

II. SYSTEM MODEL

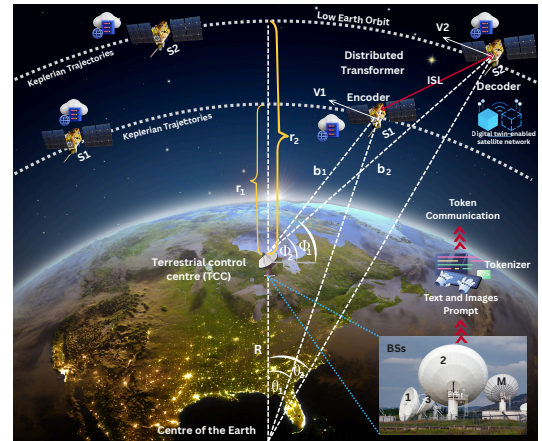


Fig. 1: An illustration of a 6G-enabled NTN that supports token communication and distributed multimodal LLM inference for generative AI services across the terrestrial segment and satellite edge nodes.

Fig. 1 illustrates a 6G DT-enabled NTN that supports distributed multimodal LLM inference. The architecture comprises a terrestrial control centre (TCC) and two LEO satellites equipped with edge-computing capabilities, where S_1 operates at a lower altitude and S_2 operates at a higher altitude. Moreover, both satellites assumed to follow ideal Keplerian trajectories. The TCC serves as the centralized terrestrial control unit and includes a set of BSs, denoted by $\mathcal{M} = \{1, 2, \dots, M\}$. In order to support generative AI workloads such as image generation and visual question answering, we extend the NTN model to enable distributed transformer-based multimodal LLM across satellites S_1 and S_2 . Each BS $m \in \mathcal{M}$ generates a sequence of multimodal prompts $\mathcal{P}_m = \{q_m, I_m\}$, consisting of a text query q_m and an image I_m .

Then, these prompts are transmitted to S_1 , which is equipped with N antennas. Satellite S_1 has sufficient onboard energy and processing capability to execute the initial inference stage, including prompt encoding, fusion, and compression of intermediate hidden representations during its coverage period. The compressed data is then transmitted to satellite S_2 for the decoding process using the inter-satellite link (ISL). Then, satellite S_2 execute the decoding process. The low altitude and rapid orbital motion of S_1 limit its coverage time with the BSs.

Since LEO satellites generally support multiple communication services and platform operations, only a portion of each satellite's coverage window is allocated to the considered multimodal AI inference services. Therefore, a higher-orbit satellite S_2 with a longer coverage window is employed to provide greater service continuity and scheduling flexibility for ISL transmission and decoding while maintaining QoS. Moreover, satellites S_1 and S_2 are equipped with solar panels to harvest energy from solar radiation. The harvested energy is stored in onboard batteries to support computational and communication tasks. In this work, we assume that the Doppler shift caused by the high orbital velocity of LEO satellites is compensated. This compensation is possible because LEO satellite trajectories are predictable. Therefore, the Doppler shift can be estimated using the orbital information. Based on this estimate, frequency pre-compensation and receiver-side tracking can be applied. Accordingly, the transmission rates are calculated using the effective channel conditions after Doppler compensation.

A. Satellite Mobility and Coverage

1) *Satellite Mobility*: We consider that S_1 operates along orbit Y_1 with an orbital velocity denoted by v_1 . This velocity can be calculated as $v_1 = \sqrt{\frac{GM_E}{R+r_1}}$, where G is the gravitational constant, M_E is the mass of the Earth, R is the Earth's radius, and r_1 is the altitude of S_1 above the Earth's surface. Similarly, S_2 moves along orbit Y_2 with a velocity denoted by v_2 , which can be calculated as $v_2 = \sqrt{\frac{GM_E}{R+r_2}}$, where r_2 is the altitude of S_2 above the Earth's surface.

2) *Satellite coverage*: Due to orbital motion, satellites S_1 and S_2 can maintain communication with BSs only for a limited time during each pass. This duration depends on the satellite altitude, orbital speed, and the minimum elevation angle required for reliable connectivity. We consider that all ground stations observe a common elevation angle ϕ_1 toward S_1 and ϕ_2 toward S_2 . Therefore, the geocentric angle θ_1 associated with satellite S_1 can be calculated as $\theta_1 = \arccos\left(\frac{R}{R+r_1} \cos(\phi_1)\right) - \phi_1$, whereas the geocentric angle θ_2 for satellite S_2 can be expressed as $\theta_2 = \arccos\left(\frac{R}{R+r_2} \cos(\phi_2)\right) - \phi_2$. [31] Accordingly, the coverage distances for satellites S_1 and S_2 can be calculated as

$$L_{S_1} = \frac{2(R+r_1)\theta_1\pi}{180}, \quad L_{S_2} = \frac{2(R+r_2)\theta_2\pi}{180}. \quad (1)$$

Therefore, the maximum coverage durations of satellite S_1 and S_2 can be expressed as [32]

$$T_{\text{cov},S_1} = \frac{L_{S_1}}{v_1}, \quad T_{\text{cov},S_2} = \frac{L_{S_2}}{v_2}. \quad (2)$$

B. Channel Modeling

We model the uplink channel between the m -th BS and S_1 as $\mathbf{h}_{m,S_1}(t) = \sqrt{g_{m,S_1}(t)} \bar{\mathbf{h}}_{m,S_1}(t) \in \mathbb{C}^{N \times 1}$, where $g_{m,S_1}(t)$ denotes the large-scale channel coefficient incorporating path loss and shadowing, and $\bar{\mathbf{h}}_{m,S_1}(t) \sim \mathcal{CN}(0, I_N)$ captures the small-scale fading [33]. The corresponding channel matrix for all BSs can be denoted as $\mathbf{H}_{S_1}(t) = [\mathbf{h}_{1,S_1}(t), \dots, \mathbf{h}_{M,S_1}(t)] \in \mathbb{C}^{N \times M}$. Then, the achievable data rate of m -th BS to S_1 can be calculated as [34]

$$\mathcal{R}_{m,S_1}(t) = \beta_m(t) W_1 \log_2 \left(1 + \frac{p_m(t) \|\mathbf{h}_{m,S_1}(t)\|^2}{\sigma_{S_1}^2(t)} \right). \quad (3)$$

where $p_m(t)$ is the transmit power of the m -th BS. W_1 is the system bandwidth which utilizes the Ku-band capacity and $\beta_m(t) \in [0, 1]$ bandwidth allocation coefficient for m -th BS. Moreover, $\sigma_{S_1}(t)$ is the instantaneous noise power characterized by the Gaussian complex normal distribution $\sim \mathcal{CN}(0, \sigma^2)$. Similarly, the channel between the satellite S_1 and S_2 can be modeled as $\mathbf{h}_{S_1,S_2}(t) = \sqrt{g_{S_1,S_2}(t)} \bar{\mathbf{h}}_{S_1,S_2}(t) \in \mathbb{C}^{1 \times N}$, where $g_{S_1,S_2}(t)$ denotes the large-scale fading and $\bar{\mathbf{h}}_{S_1,S_2}(t) \sim \mathcal{CN}(0, I)$ represents the small-scale fading [33]. Consequently, the achievable data rate of ISL can be expressed as [34]

$$\mathcal{R}_{S_1,S_2}(t) = W_2 \log_2 \left(1 + \frac{P_{S_1,S_2}(t) \|\mathbf{h}_{S_1,S_2}(t)\|^2}{\sigma_{S_2}^2(t)} \right), \quad (4)$$

where σ_{S_2} is the instantaneous noise power characterized by the Gaussian complex normal distribution $\sim \mathcal{CN}(0, \sigma^2)$. Moreover, W_2 is the ISL system bandwidth.

C. DT-enhanced Distributed Multimodal LLM Inference Pipeline with Latency and Energy model

We consider that each BS $m \in \mathcal{M}$ generates a multimodal prompt $\mathcal{P}_m = \{q_m, I_m\}$. Here q_m represents the text query and I_m denotes the associated image of the m -th BS. Moreover, we represent the prompt generated by the m -th BS to a task which can be expressed as a tuple $J_m = \{D_m(t), C_m(t), T_m^{\max}\}$, where $D_m(t)$ is the data size of the m -th BS's prompt. $C_m(t)$ is the computational complexity of the m -th BS's prompt and $T_m^{\max}$ is the maximum tolerable delay of the m -th BS's prompt. In order to calculate the data size $D_m(t)$, the prompt $\mathcal{P}_m$ undergoes preprocessing locally at each BS before it is transmitted to satellite S_1 . At the BS, the text q_m in the prompt is tokenized using the tiktoken library, aligned with the *o200k_harmony* vocabulary. Therefore, the embedded size of the text query q_m can be calculated as $D_m^{\text{text}}(t) = T_{\text{txt}}(t) b_{\text{tok}}$, where $T_{\text{txt}}(t)$ is the number of tokens and b_{tok} is the bit-width per token. Moreover, the image I_m in the prompt of m -th BS is encoded using a vision encoder using CLIP ViT-B/32 into $T_I(t)$ visual

tokens, each of dimension D_I [30]. Thus, the embedded image size can be calculated as $D_m^{\text{img}}(t) = T_I(t)D_I b$, where b is the bit precision. This preprocessing reduces transmission overhead by converting raw data to compact embeddings. Accordingly, the total input data size sent to satellite S_1 from the m -th BS can be expressed as $D_m(t) = D_m^{\text{text}}(t) + D_m^{\text{img}}(t)$. Therefore, the time to transmit the prompt data of m -th BS to satellite S_1 can be calculated as

$$T_{m,S_1}^{\text{trans}}(t) = \frac{D_m(t)}{\mathcal{R}_{m,S_1}(t)}. \quad (5)$$

Furthermore, to transmit the prompt data of the m -th BS to satellite S_1 energy consumption can be calculated as

$$E_{m,S_1}^{\text{trans}}(t) = p_m(t) \frac{D_m(t)}{\mathcal{R}_{m,S_1}(t)}. \quad (6)$$

Moreover, to incorporate the structural modularity of the multimodal LLM into our inference model, we consider that the inference pipeline follows a transformer architecture comprising $L_1 + L_2$ layers, where each layer contains both a multi-head attention module and a feed-forward network. The multimodal LLM is split across satellites S_1 and S_2 to optimize computation and coverage. Let L_1 denote the number of encoder and fusion layers executed on satellite S_1 , while the decoder layers L_2 are executed on satellite S_2 .

At satellite S_1 , the computational complexity is measured in floating-point operations (FLOPs). The per-layer complexity for processing the prompt of the m -th BS can be approximated as $C_{m,S_1}(t) \approx \rho_{S_1} (\alpha_{\text{mat}} T_m^2(t) n_h d_k + \alpha_{\text{proj}} T_m(t) d_{\text{model}}^2 + \alpha_{\text{ff}} T_m(t) d_{\text{model}} d_{\text{ff}})$. This complexity includes the encoding and fusion processes. Here, ρ_{S_1} represents the dimensionless workload scaling factor associated with the GPU efficiency at satellite S_1 . The input sequence length is $T_m(t) = T_{\text{txt}}(t) + T_I(t)$. Moreover, d_{model} is the embedding dimension. The parameter d_{ff} is the hidden dimension of the feed-forward network. Furthermore, α_{mat} , α_{proj} , and α_{ff} denote the corresponding scaling coefficients. They represent the computational contributions of attention matrix multiplications, projection layers, and feed-forward transformations, respectively. These coefficients enable a flexible and architecture-agnostic complexity model [5], [35].

After completing the encoder and fusion processes, satellite S_1 obtains the final encoder representation. This representation is used to generate the cross-attention key and value tensors required by the L_2 decoder layers. These key-value (KV) tensors are compressed and transferred to satellite S_2 through the ISL. For modeling simplicity, the computational overhead associated with generating these tensors is incorporated into the effective projection coefficient α_{proj} . Therefore, this overhead is included in the aggregate S_1 workload $L_1 C_{m,S_1}(t)$. Then, the ISL transmission latency can be calculated as

$$T_{m,S_1,S_2}^{\text{trans}}(t) = \frac{2\vartheta L_2 T_m(t) n_h d_k}{\mathcal{R}_{S_1,S_2}(t) \eta}, \quad (7)$$

where the factor 2 accounts for the key and value tensors. The parameter ϑ is the bit precision per tensor element.

Moreover, η is the compression ratio. It is defined as the ratio of the uncompressed tensor size to the compressed tensor size. Furthermore, n_h is the number of attention heads. The per-head dimension used for both the key and value tensors is $d_k = d_{\text{model}}/n_h$. Accordingly, the corresponding ISL transmission energy consumption can be calculated as

$$E_{m,S_1,S_2}^{\text{trans}}(t) = P_{S_1,S_2}(t) \frac{2\vartheta L_2 T_m(t) n_h d_k}{\mathcal{R}_{S_1,S_2}(t) \eta}, \quad (8)$$

where $P_{S_1,S_2}(t)$ is the transmit power from satellite S_1 to satellite S_2 . At satellite S_2 , each decoder layer receives a separate compressed KV tensor pair. Each tensor pair is decompressed independently. It is then mapped into the attention space of its corresponding decoder layer. The mapped tensors are subsequently used in the cross-attention operations. The computational complexity in FLOPs per layer for decoding the m -th BS prompt can be approximated as $C_{m,S_2}(t) \approx \rho_{S_2} (\beta_{\text{sa-mat}} T_{\text{dec}}^2 n_h d_k + \beta_{\text{sa-proj}} T_{\text{dec}} d_{\text{model}}^2 + \beta_{\text{ca-mat}} T_{\text{dec}} T_m(t) n_h d_k + \beta_{\text{ca-proj}} (T_{\text{dec}} + T_m(t)) d_{\text{model}}^2 + \beta_{\text{ff}} T_{\text{dec}} d_{\text{model}} d_{\text{ff}})$.

Here, ρ_{S_2} denotes the dimensionless workload scaling factor associated with the computational efficiency of satellite S_2 . The parameter T_{dec} is the decoding sequence length. Moreover, $\beta_{\text{sa-mat}}$, $\beta_{\text{sa-proj}}$, $\beta_{\text{ca-mat}}$, $\beta_{\text{ca-proj}}$, and β_{ff} represent the scaling coefficients associated with the decoder computations. Specifically, they capture the computational contributions of self-attention matrix multiplications and self-attention projection operations. They also capture the contributions of cross-attention matrix multiplications, cross-attention projection operations, and feed-forward transformations [5], [35]. In particular, $\beta_{\text{ca-proj}}$ captures the overall decoder-side cross-attention projection workload. The T_{dec} component accounts for the query and output projection operations associated with the decoder sequence. The $T_m(t)$ component accounts for the decoder-side mapping of the received compressed KV tensors. This mapping is performed independently for each decoder layer. This is because the L_2 KV tensor pairs are transferred, decompressed, and processed separately. The constant multiplicities associated with these projection operations are incorporated into $\beta_{\text{ca-proj}}$. Therefore, the total task complexity can be expressed as $C_m(t) = L_1 C_{m,S_1}(t) + L_2 C_{m,S_2}(t)$.

Moreover, we define the DT services at satellite S_1 and satellite S_2 for processing the m -th BS prompt as $\mathbf{DT}_m^{S_1}$ and $\mathbf{DT}_m^{S_2}$, respectively. At satellite S_1 , the DT service can be expressed as $\mathbf{DT}_m^{S_1} = \{f_m^{S_1}(t), \hat{f}_m^{S_1}(t)\}$. Here, $f_m^{S_1}(t)$ is the estimated processing rate in FLOP/s. Moreover, $\hat{f}_m^{S_1}(t)$ is the deviation between the estimated and actual processing rates of satellite S_1 . Similarly, the DT service at satellite S_2 can be expressed as $\mathbf{DT}_m^{S_2} = \{f_m^{S_2}(t), \hat{f}_m^{S_2}(t)\}$. Here, $f_m^{S_2}(t)$ is the estimated processing rate in FLOP/s, while $\hat{f}_m^{S_2}(t)$ is its corresponding deviation. Then, the time required to process the m -th BS prompt at satellites S_1 and S_2 can be calculated as [34]

$$T_{m,S_1}^{\text{proc}} = \frac{L_1 C_{m,S_1}(t)}{f_m^{S_1}(t) - \hat{f}_m^{S_1}(t)}, \quad T_{m,S_2}^{\text{proc}} = \frac{L_2 C_{m,S_2}(t)}{f_m^{S_2}(t) - \hat{f}_m^{S_2}(t)}. \quad (9)$$

The processing latency in (9) consists of two components, namely, the estimated processing latency and the deviation latency. For satellite S_1 , the processing latency can be expressed as $T_{m,S_1}^{\text{proc}} = \bar{T}_{m,S_1}^{\text{proc}} + \Delta T_{m,S_1}^{\text{proc}}$. Here, the estimated processing latency is $\bar{T}_{m,S_1}^{\text{proc}} = L_1 C_{m,S_1}(t) / f_m^{S_1}(t)$. The corresponding deviation latency is $\Delta T_{m,S_1}^{\text{proc}} = L_1 C_{m,S_1}(t) \hat{f}_m^{S_1}(t) / [f_m^{S_1}(t)(f_m^{S_1}(t) - \hat{f}_m^{S_1}(t))]$. Similarly, the processing latency at satellite S_2 can be expressed as $T_{m,S_2}^{\text{proc}} = \bar{T}_{m,S_2}^{\text{proc}} + \Delta T_{m,S_2}^{\text{proc}}$. Here, the estimated processing latency is $\bar{T}_{m,S_2}^{\text{proc}} = L_2 C_{m,S_2}(t) / f_m^{S_2}(t)$. The corresponding deviation latency is $\Delta T_{m,S_2}^{\text{proc}} = L_2 C_{m,S_2}(t) \hat{f}_m^{S_2}(t) / [f_m^{S_2}(t)(f_m^{S_2}(t) - \hat{f}_m^{S_2}(t))]$. This decomposition allows the processing latency model to account for the mismatch between the DT-estimated processing rate and the actual processing capability of each satellite.

Based on the effective processing rates, the processing energy consumption at satellite S_1 can be calculated as $E_{m,S_1}^{\text{proc}}(t) = \zeta_{S_1} L_1 C_{m,S_1}(t) (f_m^{S_1}(t) - \hat{f}_m^{S_1}(t))^2$. Similarly, the processing energy consumption at satellite S_2 can be calculated as $E_{m,S_2}^{\text{proc}}(t) = \zeta_{S_2} L_2 C_{m,S_2}(t) (f_m^{S_2}(t) - \hat{f}_m^{S_2}(t))^2$, where ζ_{S_1} and ζ_{S_2} are the computation power parameters for the processors at satellites S_1 and S_2 , respectively [33]. It is important to note that, in this work, we assume that the DT models of satellites S_1 and S_2 are periodically synchronized with their physical counterparts during the available communication windows. Since LEO satellite trajectories are predictable, the synchronization periods can be determined based on their orbital information and coverage durations. During each synchronization, the DT parameters are updated according to the current processing states of the satellites.

Therefore, the total latency to complete a multimodal LLM task of the m -th BS can be expressed as

$$T_m^{\text{tot}}(t) = T_{m,S_1}^{\text{trans}}(t) + T_{m,S_1}^{\text{proc}}(t) + T_{m,S_1,S_2}^{\text{trans}}(t) + T_{m,S_2}^{\text{proc}}(t), \quad (10)$$

Moreover, total energy consumption for completing a multimodal LLM task of the m -th BS can be expressed as

$$E_m^{\text{tot}}(t) = E_{m,S_1}^{\text{trans}}(t) + E_{m,S_1}^{\text{proc}}(t) + E_{m,S_1,S_2}^{\text{trans}}(t) + E_{m,S_2}^{\text{proc}}(t). \quad (11)$$

D. Quantum Key Distribution Overhead

To secure the ISL between satellite S_1 and S_2 , we use a QKD mechanism operating over a dedicated free-space optical (FSO) link. The QKD protocol runs periodically to refresh the cryptographic keys, which are then used to encrypt the RF data transmission over the ISL. We adopt a discrete-time model indexed by $t \in \mathbb{Z}_{\geq 0}$, with one-second time slots. Furthermore, the QKD refresh interval $T_{\text{QKD}}^{\text{interval}} \in \mathbb{Z}_{>0}$ is measured in time slots. Each refresh has a fixed latency cost, denoted by $T_{\text{QKD}}^{\text{cost}}$, which accounts for the time required for QKD execution, including pointing, acquisition, and tracking of the FSO terminals. Let $Q(t)$ be an indicator variable such that $Q(t) = 1$ if a QKD key refresh occurs at slot t , and $Q(t) = 0$ otherwise. A refresh occurs whenever $(t \bmod T_{\text{QKD}}^{\text{interval}} = 0)$. Accordingly, the QKD latency at slot t can be expressed as $T_{\text{QKD}}(t) = Q(t) T_{\text{QKD}}^{\text{cost}}$. This

QKD latency consumes time within the S_2 coverage window and is captured at the system level through the S_2 window constraint. While this constraint is necessary, it creates a trade-off between the key refresh frequency and the time available for ISL data transmission and decoding.

E. Problem Formulation

In this paper, we aim to minimize the total system utility cost by jointly optimizing the system bandwidth allocation, uplink transmit powers, and computational power allocations at the satellite edges. Accordingly, the objective function can be expressed as

$$f(X) = w_1 \sum_{m=1}^M T_m^{\text{tot}}(t) + w_2 \sum_{m=1}^M E_m^{\text{tot}}(t), \quad (12)$$

Where, X is the set of optimization variables which can be represented as $X \triangleq \{\mathbf{p}(t), \boldsymbol{\beta}(t), \mathbf{f}^{S_1}(t), \mathbf{f}^{S_2}(t)\}$, where $\mathbf{p}(t) \triangleq \{p_m(t)\}_{m \in \mathcal{M}}$ denotes the uplink transmit powers of the BSs, $\boldsymbol{\beta}(t) \triangleq \{\beta_m(t)\}_{m \in \mathcal{M}}$ denotes the bandwidth allocation factors, $\mathbf{f}^{S_1}(t) \triangleq \{f_m^{S_1}(t)\}_{m \in \mathcal{M}}$ and $\mathbf{f}^{S_2}(t) \triangleq \{f_m^{S_2}(t)\}_{m \in \mathcal{M}}$ denote the computational power allocations at satellites S_1 and S_2 , respectively. Accordingly, the optimization problem can be formulated as follows:

$$(\mathbf{P1}) \quad \min_X f(X) = w_1 \sum_{m=1}^M T_m^{\text{tot}}(t) + w_2 \sum_{m=1}^M E_m^{\text{tot}}(t) \quad (13a)$$

$$\text{s.t.} \quad 0 \leq \beta_m(t) \leq 1, \quad \forall m, \quad \sum_{m=1}^M \beta_m(t) \leq 1, \quad (13b)$$

$$0 \leq p_m(t) \leq P_m^{\text{max}}, \quad \forall m, \quad (13c)$$

$$0 \leq f_m^{S_1}(t) \leq F_{m,S_1}^{\text{max}}, \quad 0 \leq f_m^{S_2}(t) \leq F_{m,S_2}^{\text{max}}, \quad \forall m, \quad (13d)$$

$$\sum_{m=1}^M f_m^{S_1}(t) \leq F_{S_1}^{\text{max}}, \quad \sum_{m=1}^M f_m^{S_2}(t) \leq F_{S_2}^{\text{max}}, \quad (13e)$$

$$f_m^{S_1}(t) - \hat{f}_m^{S_1}(t) > 0, \quad i \in \{1, 2\}, \quad \forall m, \quad (13f)$$

$$\max_{m \in \mathcal{M}} (T_{m,S_1}^{\text{trans}}(t) + T_{m,S_1}^{\text{proc}}(t)) \leq \chi_1 T_{\text{cov},S_1}, \quad (13g)$$

$$T_{\text{QKD}}(t) + \sum_{m=1}^M T_{m,S_1,S_2}^{\text{trans}}(t) + \max_{m \in \mathcal{M}} T_{m,S_2}^{\text{proc}}(t) \leq \chi_2 T_{\text{cov},S_2}, \quad (13h)$$

$$T_m^{\text{tot}}(t) \leq T_m^{\text{max}}, \quad \forall m, \quad (13i)$$

$$\sum_{m=1}^M (E_{m,S_1,S_2}^{\text{trans}}(t) + E_{m,S_1}^{\text{proc}}(t)) \leq E_{S_1}^{\text{solar}}, \quad (13j)$$

$$\sum_{m=1}^M E_{m,S_2}^{\text{proc}}(t) \leq E_{S_2}^{\text{solar}}. \quad (13k)$$

In (13), the objective function in (13a) minimizes the total weighted utility cost of the system. Constraint (13b) ensures that each bandwidth allocation factor satisfies $\beta_m(t) \in [0, 1]$ and that the total allocated bandwidth does not exceed the

available system bandwidth. Constraint (13c) limits the transmit power of each BS to $P_m^{\max}$. Constraint (13d) bounds the processing rates allocated to each task at satellites S_1 and S_2 , while constraint (13e) ensures that the total allocated processing rates do not exceed $F_{S_1}^{\max}$ and $F_{S_2}^{\max}$. Constraint (13f) guarantees that the effective processing rates $f_m^{S_1}(t) - \hat{f}_m^{S_1}(t)$ remain positive. The parameters χ_1 and χ_2 are fixed coverage-window utilization factors that represent the portions of the complete coverage windows allocated to the considered multimodal LLM inference tasks at satellites S_1 and S_2 , respectively. Constraint (13g) ensures that the FDMA uplink transmissions and parallel processing at S_1 are completed within its allocated coverage portion. Similarly, constraint (13h) ensures that the periodic QKD refresh, sequential ISL transmissions, and parallel processing at S_2 are completed within its allocated coverage portion. The QKD overhead is treated as a system-level latency and is therefore included only once through the S_2 coverage-window constraint. Constraint (13i) ensures that the total latency of each task does not exceed its maximum tolerable latency $T_m^{\max}$. Constraints (13j) and (13k) limit the energy consumed at S_1 for ISL transmission and processing, and at S_2 for processing, respectively. The corresponding energy consumption must remain within the solar-energy budgets $E_{S_1}^{\text{solar}}$ and $E_{S_2}^{\text{solar}}$.

III. PROPOSED SOLUTION

The optimization problem in (13) is a complex NLP problem involving high-dimensional continuous decision variables. Here, the high-dimensional continuous action space, combined with the nonlinear coupling across energy, latency, and dynamic satellite coverage constraints leads to a complex search space that is intractable for conventional optimization techniques. To address these challenges, we propose two reservoir-assisted DRL frameworks, namely ESN-Enhanced DDPG and QRC-Enhanced DDPG. Here, ESN-Enhanced DDPG serves as a classical reservoir-assisted benchmark to quantify the gains from temporal reservoir memory, whereas QRC-Enhanced DDPG constitutes the proposed quantum reservoir critic for further stability and accelerated value-function learning. Both frameworks reformulate (13) as a constrained Markov decision process and learn continuous resource allocation policies. The ESN-based critic exploits a fixed classical reservoir to strengthen temporal feature extraction, whereas the QRC-based critic employs higher-order quantum encoding with fixed multi-layer quantum reservoirs to improve learning stability and accelerate value-function approximation in high-dimensional environments. The resulting policies dynamically minimize the weighted system cost, which jointly captures total task latency and total energy consumption, thereby enabling scalable and energy-efficient generative AI resource allocation in 6G-enabled NTN.

A. Deep Reinforcement Learning Representation

In this section, the formulated NLP problem defined in (13) is transformed as a constrained MDP to enable adaptive

and sequential decision-making under dynamic network conditions and uncertainty. This transformation allows the DRL agent to learn optimal control strategies through continuous interaction with the environment. Within the DRL framework, the MDP formulation can be characterized by three essential components. This can be expressed as the action space, which represents the set of possible control decisions, the observation space which captures the system state information perceived by the agent and the reward function, which quantifies the trade-off between latency and energy efficiency to guide the learning process toward the optimal policy.

Action Space: The action space $\mathcal{A}$ comprises all continuous control variables that the DRL agent is responsible for optimizing at each decision making step t . This action $a(t)$ is a high-dimensional vector that directly influences the system's performance and resource consumption. Therefore, this action vector $\mathbf{a}(t)$ can be expressed as $\mathbf{a}(t) = [\mathbf{p}(t), \boldsymbol{\beta}(t), \mathbf{f}^{S_1}(t), \mathbf{f}^{S_2}(t)]$. Here, $\mathbf{p}(t) = [p_m(t)]_{m \in \mathcal{M}}$ is the vector of uplink transmit powers for the M BSs. Moreover, $\boldsymbol{\beta}(t) = [\beta_m(t)]_{m \in \mathcal{M}}$ is the vector of bandwidth allocation factors controlling the FDMA channel access for the BSs. Furthermore, $\mathbf{f}^{S_1}(t) = [f_m^{S_1}(t)]_{m \in \mathcal{M}}$ and $\mathbf{f}^{S_2}(t) = [f_m^{S_2}(t)]_{m \in \mathcal{M}}$ are the vectors of computational power allocations in FLOP/s assigned to each task on satellites S_1 and S_2 , respectively.

Observation Space: The observation space $\mathcal{S}$ represents the information provided to the agent at each time slot t . The design of this space is critical, as it forms the entire basis of knowledge from which the agent should derive its understanding of the system and select its subsequent actions. Accordingly, the observation vector is expressed as $\mathbf{s}(t) = [\tilde{\mathbf{D}}(t), \tilde{\mathbf{C}}(t), \tilde{\mathbf{g}}_{S_1}(t), \tilde{\mathbf{T}}_{S_1}^{\text{trans}}(t), \tilde{\mathbf{E}}_{S_1}^{\text{trans}}(t), \tilde{\mathbf{T}}_{S_1}^{\text{proc}}(t), \tilde{\mathbf{E}}_{S_1}^{\text{proc}}(t), \tilde{\mathbf{g}}_{S_1, S_2}(t), \tilde{\mathbf{T}}_{S_2}^{\text{proc}}(t), \tilde{\mathbf{E}}_{S_2}^{\text{proc}}(t)]$. Here, $\tilde{\mathbf{D}}(t) = [\tilde{D}_m(t)]_{m \in \mathcal{M}}$ and $\tilde{\mathbf{C}}(t) = [\tilde{C}_m(t)]_{m \in \mathcal{M}}$ are the normalized data sizes and computational complexities of the tasks, respectively. Where $\tilde{D}_m(t) = \frac{D_m(t)}{\text{NormFactor}_D}$ and $\tilde{C}_m(t) = \frac{C_m(t)}{\text{NormFactor}_C}$. Moreover, $\tilde{\mathbf{g}}_{S_1}(t)$ represents the normalized uplink channel gains between the BSs and satellite S_1 . The $\tilde{\mathbf{T}}_{S_1}^{\text{trans}}(t)$ and $\tilde{\mathbf{E}}_{S_1}^{\text{trans}}(t)$ represent the normalized uplink transmission latencies and energy-consumption values. Furthermore, $\tilde{\mathbf{T}}_{S_1}^{\text{proc}}(t)$ and $\tilde{\mathbf{E}}_{S_1}^{\text{proc}}(t)$ represent the normalized processing latencies and processing-energy consumption at satellite S_1 . The $\tilde{\mathbf{g}}_{S_1, S_2}(t)$ denotes the normalized ISL channel gain between satellites S_1 and S_2 . Moreover, $\tilde{\mathbf{T}}_{S_2}^{\text{proc}}(t)$ and $\tilde{\mathbf{E}}_{S_2}^{\text{proc}}(t)$ represent the normalized processing latencies and processing-energy consumption at satellite S_2 , respectively.

Reward Function: The reward function $\mathcal{R}$ is a critical component that quantifies the immediate feedback for the agent's action $a(t)$ at time slot t . It is designed to guide the agent toward the primary objective of the optimization problem in (13), which is to minimize the weighted system cost. Since the DRL agent aims to maximize the cumulative reward, we formulate the reward function as the negative of the objective function $f(X)$ defined in (12). Furthermore, to ensure that the agent learns a feasible policy while respecting the system's operational limits, we introduce a penalty term $P(t)$

that penalizes constraint violations. Accordingly, the reward function at time t is expressed as

$$r(t) = \frac{1}{W_0} \left[- \left(w_1 \sum_{m=1}^M T_m^{\text{tot}}(t) + w_2 \sum_{m=1}^M E_m^{\text{tot}}(t) \right) - P(t) \right], \quad (14)$$

where W_0 is the reward-scaling factor. Moreover, the penalty term $P(t)$ can be expressed as

$$\begin{aligned} P(t) = & \lambda_1 \max \left\{ 0, \max_{m \in \mathcal{M}} \left(T_{m,S_1}^{\text{trans}}(t) + T_{m,S_1}^{\text{proc}}(t) \right) - \chi_1 T_{\text{cov},S_1} \right\} \\ & + \lambda_2 \max \left\{ 0, T_{\text{QKD}}(t) + \sum_{m=1}^M T_{m,S_1,S_2}^{\text{trans}}(t) \right. \\ & \quad \left. + \max_{m \in \mathcal{M}} T_{m,S_2}^{\text{proc}}(t) - \chi_2 T_{\text{cov},S_2} \right\} \\ & + \lambda_3 \sum_{m=1}^M \max \left\{ 0, T_m^{\text{tot}}(t) - T_m^{\text{max}} \right\} \\ & + \lambda_4 \max \left\{ 0, \sum_{m=1}^M \left(E_{m,S_1,S_2}^{\text{trans}}(t) + E_{m,S_1}^{\text{proc}}(t) \right) \right. \\ & \quad \left. - E_{S_1}^{\text{solar}} \right\} \\ & + \lambda_5 \max \left\{ 0, \sum_{m=1}^M E_{m,S_2}^{\text{proc}}(t) - E_{S_2}^{\text{solar}} \right\}. \end{aligned} \quad (15)$$

Here, $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, and λ_5 are the weight parameters for constraints (13g)-(13k), respectively. To enforce these constraints, their violation terms are added to the reward function as penalties. Each penalty is zero when its constraint is satisfied and increases with the amount of the violation. A higher weight gives a larger negative reward, while a lower weight reduces its effect. The weight parameters are selected to balance the system cost and constraint violations during training. Moreover, w_1 and w_2 balance the latency and energy terms. After combining the weighted system cost and penalties, the reward is divided by W_0 . This normalizes the reward and keeps its value within a suitable range during training without changing the relative contribution of its terms. In contrast, constraints (13b)-(13f) are enforced through clipping and normalization of the actor outputs before the actions are applied to the environment.

B. Quantum Reservoir Computing-Enhanced DDPG Algorithm

To address the formulated MDP, this study introduces the QRC-Enhanced DDPG framework. The framework extends the DDPG algorithm, a model-free actor-critic DRL approach developed for high-dimensional continuous control problems. The DDPG algorithm is an off-policy actor-critic method. A key challenge in continuous domains is that traditional Q-learning, which searches for the maximum Q-value over all possible actions, becomes intractable since it requires optimization over a very large-dimensional action space at each step. DDPG overcomes this limitation by jointly learning two separate neural networks known as an actor and a critic.

Thus, the actor network, denoted as $\pi(\mathbf{s}(t)|\theta^\pi)$, represents the agent's deterministic policy. It receives the current state $s(t)$ as input and produces a continuous action $\mathbf{a}(t) = \pi(\mathbf{s}(t)|\theta^\pi)$ as output. The objective of the actor, parameterized by θ^π , is to learn the optimal policy that maximizes the expected long-term discounted reward. In the proposed framework, the actor is implemented as a classical, fully connected deep neural network. Furthermore, the critic network, denoted as $Q(\mathbf{s}(t), \mathbf{a}(t)|\theta^Q)$, evaluates the actions generated by the actor. It estimates the action-value function, parameterized by θ^Q . The network takes both the state $\mathbf{s}(t)$ and the action $\mathbf{a}(t)$ as inputs and outputs a scalar value $Q(\mathbf{s}(t), \mathbf{a}(t)|\theta^Q)$. This value represents the expected cumulative reward obtained by taking action $\mathbf{a}(t)$ in state $\mathbf{s}(t)$ and following the policy π which can be denoted as

$$Q^\pi(\mathbf{s}(t), \mathbf{a}(t)) = \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k r(t+k) \mid s(t) = s, \mathbf{a}(t) = \mathbf{a} \right]. \quad (16)$$

Here, γ denotes the discount factor, whereas $r(t+k)$ represents the reward at time step $t+k$. Moreover, this formulation allows the critic to provide accurate value estimates, which guide the actor during policy updates [7].

QRC-Enhanced DDPG Architecture: In the proposed QRC-Enhanced DDPG framework in Fig. 2, the classical critic network is substituted by a hybrid quantum-classical QRC module which processes sequences of states and actions. This hybrid quantum-classical configuration leverages the high-dimensional quantum feature representation produced by the fixed quantum reservoir and the temporal information maintained by a classical leaky integrator to more effectively approximate the action-value function, thereby enhancing learning stability and policy performance. Thus, the QRC critic is composed of three distinct components, namely, a classical trainable encoder, a fixed quantum reservoir, and a classical trainable readout. Here, the encoder is a classical neural network, $F_{\text{ENC}}(\cdot|\theta_{\text{ENC}})$, that functions as a trainable feature map. It processes the concatenated state $\mathbf{s}(t)$ and action $\mathbf{a}(t)$ at each time step t in a sequence. Let $\mathbf{u}(t) = [\mathbf{s}(t), \mathbf{a}(t)] \in \mathbb{R}^{d_s+d_a}$ be the concatenated input vector. The encoder transforms this vector into an n -dimensional input $\mathbf{x}(t) \in \mathbb{R}^n$ suitable for the n -qubit quantum reservoir, which can be denoted as

$$\mathbf{x}(t) = \tanh \left[F_{\text{ENC}}(\mathbf{u}(t)|\theta_{\text{ENC}}) \right], \quad (17)$$

where F_{ENC} is a multi-layer perceptron and the tanh activation scales the output to the range $[-1, 1]$. Thus, the core of the critic is the quantum reservoir. This is an n -qubit quantum circuit defined by a fixed, non-trainable unitary operator. At each time step, this module functions as a nonlinear quantum feature map that processes the corresponding encoded input $\mathbf{x}(t)$. The temporal information across the sequence is subsequently accumulated through the classical leaky-integrator state. The encoded classical vector $\mathbf{x}(t)$ from the encoder is injected into the quantum state using a nonlinear feature map. This map captures both first-order features and second-order feature interactions, which is referred to as higher-order

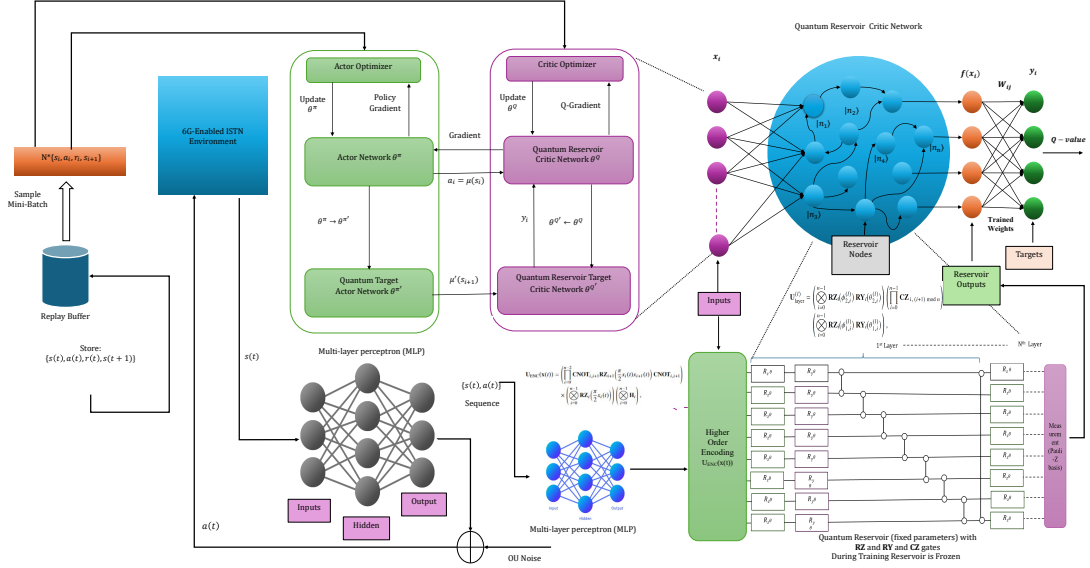


Fig. 2: Proposed quantum reservoir computing-enhanced deep deterministic policy gradient algorithm architecture.

encoding. The unitary matrix for higher-order encoding can be expressed as [11]

$$\mathbf{U}_{\text{ENC}}(\mathbf{x}(t)) = \left(\prod_{i=0}^{n-2} \mathbf{CNOT}_{i,i+1} \mathbf{RZ}_{i+1} \left(\frac{\pi}{2} x_i(t) x_{i+1}(t) \right) \mathbf{CNOT}_{i,i+1} \right) \times \left(\bigotimes_{i=0}^{n-1} \mathbf{RZ}_i \left(\frac{\pi}{2} x_i(t) \right) \right) \left(\bigotimes_{i=0}^{n-1} \mathbf{H}_i \right), \quad (18)$$

where the Hadamard gates $\bigotimes_{i=0}^{n-1} \mathbf{H}_i$ transform the initial state $|0\rangle^{\otimes n}$ into the equal-superposition state $|+\rangle^{\otimes n}$, enabling the subsequent phase rotations to encode the classical input features. Moreover, $\mathbf{RZ}_i(\frac{\pi}{2} x_i(t))$ represents a single-qubit rotation around the Z-axis that encodes the first-order feature component $x_i(t)$ of the classical vector $\mathbf{x}(t)$. The $\mathbf{CNOT}_{i,i+1}$ gates, together with the intermediate $\mathbf{RZ}_{i+1}$ rotation, implement an interaction between adjacent qubits i and $i+1$, enabling correlations between neighboring features to be represented quantum-mechanically. In particular, the rotation $\mathbf{RZ}_{i+1}(\frac{\pi}{2} x_i(t) x_{i+1}(t))$ applies a phase determined by the product of the two corresponding features. The product over i sequentially applies these ZZ-interaction blocks across all adjacent qubit pairs, producing a global quantum state whose phase encodes both the individual feature magnitudes and their pairwise product interactions.

Following the encoding, the state evolves through the main reservoir unitary matrix. This is a quantum circuit with a fixed, non-trainable set of parameters. The encoded vector $\mathbf{x}(t)$ drives the evolution of the reservoir's quantum state at each time step. The quantum state is initialized from $|0\rangle^{\otimes n}$ for each encoded input, and hence the quantum circuit functions as a nonlinear feature transformation rather than maintaining temporal memory between consecutive inputs. This hybrid architecture im-

plements the leaky-integrator memory mechanism classically. This design is a direct analogue to a classical ESN, separating the complex nonlinear mapping, which is performed by the quantum circuit, from the temporal memory, which is handled by a classical equation. Furthermore, this quantum reservoir's unitary transformation consists of L_q quantum circuit layers. Thus, the unitary matrix for the quantum reservoir can be denoted as $\mathbf{U}_{\text{RES}} = \prod_{l=1}^{L_q} \mathbf{U}_{\text{layer}}^{(l)}$. Here, $\mathbf{U}_{\text{layer}}^{(l)}$ can be expressed as

$$\mathbf{U}_{\text{layer}}^{(l)} = \left(\bigotimes_{i=0}^{n-1} \mathbf{RZ}_i(\phi_{2,i}^{(l)}) \mathbf{RY}_i(\theta_{2,i}^{(l)}) \right) \left(\prod_{i=0}^{n-1} \mathbf{CZ}_{i, (i+1) \bmod n} \right) \left(\bigotimes_{i=0}^{n-1} \mathbf{RZ}_i(\phi_{1,i}^{(l)}) \mathbf{RY}_i(\theta_{1,i}^{(l)}) \right), \quad (19)$$

where $\theta_{1,i}^{(l)}$, $\phi_{1,i}^{(l)}$, $\theta_{2,i}^{(l)}$, and $\phi_{2,i}^{(l)}$ are the fixed, non-trainable parameters that are randomly initialized and then frozen. Moreover, the term $\prod_{i=0}^{n-1} \mathbf{CZ}_{i, (i+1) \bmod n}$ represents the circular entangling block, which applies a CZ gate to all adjacent qubit pairs. Furthermore, this includes a wrap-around connection from qubit $n-1$ to qubit 0. Accordingly, for each encoded input $\mathbf{x}(t)$, the quantum state generated by the fixed quantum reservoir can be expressed as

$$|\psi(t)\rangle = \mathbf{U}_{\text{RES}} \mathbf{U}_{\text{ENC}}(\mathbf{x}(t)) |0\rangle^{\otimes n}. \quad (20)$$

Following the reservoir transformation at each time step t in the sequence, the model estimates the expectation value of the Pauli-Z operator for each qubit i through projective measurements. This process yields a classical feature vector $\mathbf{z}(t) \in \mathbb{R}^n$ for that specific time step, which can be denoted as [11]

$$\mathbf{z}(t)_{\text{QRC}} = [\langle \psi(t) | \mathbf{Z}_i | \psi(t) \rangle]_{i=0}^{n-1}. \quad (21)$$

Consequently, the final reservoir state is a classical vector $\mathbf{h}(t_L) \in \mathbb{R}^n$. It is computed iteratively using the standard ESN leak-rate equation, expressed as $\mathbf{h}(t) = (1 - \kappa)\mathbf{h}(t-1) + \kappa\mathbf{z}(t)$, where κ is the tunable leak-rate hyperparameter. Moreover, $\mathbf{h}(t-1)$ is the previous classical state, initialized as $\mathbf{h}(0) = \mathbf{0}$ for each sampled sequence, and $\mathbf{z}(t)$ is the newly measured feature vector from the quantum circuit. This resulting classical vector $\mathbf{h}(t_L)$ represents the compressed temporal information from the entire sequence. This final classical state vector $\mathbf{h}(t_L)$ is then fed into the classical readout network, $F_{\text{read}}(\cdot|\theta_{\text{read}})$. This network is a standard multi-layer perceptron (MLP) that functions as a trainable, nonlinear function approximator (decoder) to output the final Q -value. It takes an input obtained by concatenating the quantum reservoir's feature vector $\mathbf{h}(t_L)$ with the final raw state-action input $\mathbf{u}(t_L) = [\mathbf{s}(t_L), \mathbf{a}(t_L)]$. Therefore, the input dimension of the readout network is $n + d_s + d_a$. This hybrid input strategy is a key design choice. It allows the critic to base its final Q -value estimation on both the temporal context over the sampled sequence from the classical state $\mathbf{h}(t_L)$ and the immediate, high-precision state-action information from $\mathbf{u}(t_L)$. Therefore, this heterogeneous input allows the readout to learn a more effective function. It can correlate the temporal patterns within the sampled sequence with the immediate, precise inputs. For a sequence of length L ending at $t = t_L$, let $\mathbf{S}_t = [\mathbf{s}(t-L+1), \dots, \mathbf{s}(t)]$ and $\mathbf{A}_t = [\mathbf{a}(t-L+1), \dots, \mathbf{a}(t)]$ denote the corresponding state and action sequences, respectively. Then, the readout network F_{read} functions as the decoder that processes the combined classical vector to produce the final scalar Q -value, which can be denoted as

$$Q(\mathbf{S}_t, \mathbf{A}_t|\theta^Q)_{\text{QRC}} = F_{\text{read}}(\mathbf{h}(t_L) \parallel \mathbf{u}(t_L)|\theta_{\text{read}}), \quad (22)$$

where $\parallel$ denotes vector concatenation and $\theta^Q = \{\theta_{\text{ENC}}, \theta_{\text{read}}\}$ represents the trainable parameters of the QRC critic. This final Q -value is then used in the QRC-Enhanced DDPG update rules. During the training process, the Q -value is updated using a target value $y(t)$ derived from the Bellman equation. This process is adapted to the proposed recurrent QRC critic network, which learns from sequences of data. The QRC-Enhanced DDPG algorithm relies on a replay buffer for stable training. This buffer serves as a memory, accumulating data tuples from the agent's environmental interactions, with each tuple formatted as $(s(t), a(t), r(t), s(t+1))$. Importantly, a critical feature of the proposed method is the sampling of random mini-batches of contiguous sequences from this memory without crossing episode boundaries. This approach is vital for two reasons: it provides the temporally ordered transition sequences required by the recurrent QRC critic to capture temporal dependencies, and it allows the agent to reuse past experiences during training. In order to perform the training of the QRC critic network, the target Q -value $y(t)_{\text{QRC}}$ needs to be calculated. For the sequence ending at time step t , let $\mathbf{S}_{t+1} = [\mathbf{s}(t-L+2), \dots, \mathbf{s}(t+1)]$ denote the next-state sequence. The corresponding target action sequence is defined as $\mathbf{A}'_{t+1} = [\mathbf{a}(t-L+2), \dots, \mathbf{a}(t), \pi'(\mathbf{s}(t+1))]$, where the past actions are obtained from the replay buffer and the final action

is generated by the target actor network. Accordingly, the target Q -value can be expressed as [11]

$$y(t)_{\text{QRC}} = r(t) + \gamma Q'(\mathbf{S}_{t+1}, \mathbf{A}'_{t+1}|\theta^{Q'}), \quad (23)$$

where $r(t)$ is the immediate reward and γ is the discount factor. The $Q'(\mathbf{S}_{t+1}, \mathbf{A}'_{t+1}|\theta^{Q'})$ term represents the future value component, which is estimated by the target QRC critic network, $Q'(\cdot|\theta^{Q'})$. Consequently, the QRC critic network is then trained by minimizing the loss $L(\theta^Q)_{\text{QRC}}$ between its predicted Q -value $Q(\mathbf{S}_t, \mathbf{A}_t|\theta^Q)$ and the target Q -value $y(t)_{\text{QRC}}$. This loss can be calculated as [11]

$$L(\theta^Q)_{\text{QRC}} = \mathbb{E} \left[\left(y(t)_{\text{QRC}} - Q(\mathbf{S}_t, \mathbf{A}_t|\theta^Q) \right)^2 \right]. \quad (24)$$

Concurrently, the actor network $\pi(\mathbf{s}(t)|\theta^\pi)$ is optimized. Its objective is to learn a policy that maximizes the Q -value provided by the QRC critic network. This optimization is performed using the policy gradient method. It is important to note that the actor is non-recurrent and takes only the final state $\mathbf{s}(t)$ from the state sequence as its input. The resulting action $\mathbf{a}(t) = \pi(\mathbf{s}(t)|\theta^\pi)$ is then used to form the action sequence $\mathbf{A}_t^\pi = [\mathbf{a}(t-L+1), \dots, \mathbf{a}(t-1), \pi(\mathbf{s}(t))]$ for the critic's evaluation, where the preceding actions are obtained from the sampled sequence. Thus, the actor's loss $L(\theta^\pi)_{\text{QRC}}$ is formulated as the negative expected Q -value, which can be denoted as [11]

$$L(\theta^\pi)_{\text{QRC}} = -\mathbb{E} [Q(\mathbf{S}_t, \mathbf{A}_t^\pi|\theta^Q)]. \quad (25)$$

The actor parameters θ^π are subsequently updated by ascending the policy gradient $\nabla_{\theta^\pi} J(\theta^\pi)_{\text{QRC}}$. The gradient is then calculated using the chain rule, which can be denoted as [11]

$$\nabla_{\theta^\pi} J(\theta^\pi)_{\text{QRC}} = \mathbb{E} \left[\nabla_{\mathbf{a}(t)} Q(\mathbf{S}_t, \mathbf{A}_t^\pi|\theta^Q) \Big|_{\mathbf{a}(t)=\pi(\mathbf{s}(t))} \cdot \nabla_{\theta^\pi} \pi(\mathbf{s}(t)|\theta^\pi) \right]. \quad (26)$$

In summary, the proposed QRC-Enhanced DDPG algorithm for solving the problem formulated in (13) is provided in Algorithm 1.

C. Echo State Network-Enhanced DDPG Algorithm

In order to solve the formulated problem, we propose the ESN-Enhanced DDPG Algorithm. This framework extends the DDPG algorithm to leverage temporal dynamics. In this architecture, the actor network $\pi(\mathbf{s}(t)|\theta^\pi)$ remains a classical, non-recurrent feed-forward network that maps the current state $\mathbf{s}(t)$ to a deterministic action $\mathbf{a}(t)$.

ESN-Enhanced DDPG Architecture: The core of this model is its critic network, $Q(\mathbf{S}_t, \mathbf{A}_t|\theta^Q)$. This critic network is implemented using a classical ESN. This ESN-based critic network is composed of two main components. The first is a fixed, non-trainable classical reservoir. The second is a trainable classical readout network. The fixed classical reservoir is a large recurrent neural network (RNN) with fixed weights. It processes sequences of state-action pairs. At each time step t in a sequence, the concatenated input vector

Algorithm 1 Proposed QRC-Enhanced DDPG Algorithm for solving (13)

```

1: Initialize actor network  $\pi(s|\theta^\pi)$  with random parameters  $\theta^\pi$ .
2: Initialize QRC critic  $Q(S, A|\theta^Q)$  with parameters  $\theta^Q = \{\theta_{\text{ENC}}, \theta_{\text{read}}\}$ , where
   the quantum reservoir parameters in (19) are fixed.
3: Initialize target networks  $\pi'$  and  $Q'$  with  $\theta^{\pi'} \leftarrow \theta^\pi$  and  $\theta^{Q'} \leftarrow \theta^Q$ .
4: Set up replay buffer  $\mathcal{R}$  with capacity  $N_{\text{buffer}}$ .
5: Set sequence length  $L$  for temporal learning.
6: Initialize Ornstein-Uhlenbeck noise process  $\mathcal{O}$  for action exploration.
7: for episode = 1 to  $N_{\text{episodes}}$  do
8:   Reset noise process  $\mathcal{O}$ .
9:   Receive initial observation  $s(1)$  from the environment.
10:  for  $t = 1$  to  $T_{\text{max}}$  do
11:    Select action  $a(t) = \pi(s(t)|\theta^\pi) + \mathcal{O}(t)$ .
12:    Map  $a(t)$  to the feasible action space through clipping and normalization.
13:    Execute action  $a(t)$ .
14:    Observe reward  $r(t)$  and next state  $s(t+1)$ .
15:    Store transition  $(s(t), a(t), r(t), s(t+1))$  in  $\mathcal{R}$ .
16:    if at least  $N_{\text{batch}}$  complete sequences of length  $L$  are available in  $\mathcal{R}$  then
17:      Sample a random mini-batch of  $N_{\text{batch}}$  contiguous sequences of length
       $L$  from  $\mathcal{R}$  without crossing episode boundaries.
18:      for each sequence  $i$  in the mini-batch do
19:        Extract state sequences from  $\mathcal{R}$ .
20:        Extract action sequences from  $\mathcal{R}$ .
21:        Construct next-state sequences from  $\mathcal{R}$ .
22:        Generate the target action sequence for the next-state sequence using
         $\pi'$ .
23:        Compute target  $Q$ -value  $y_i(t)_{\text{QRC}}$  using (23).
24:      end for
25:      Update QRC Critic Network:
26:        Encode sequences  $S_i$  and  $A_i$  into  $\mathbf{x}_i$  using (17).
27:        Process  $\mathbf{x}_i$  sequentially over the  $L$  time steps to obtain  $\mathbf{z}_i(t)$  using
        (18) and (19) and update  $\mathbf{h}_i(t)$ , starting from  $\mathbf{h}_i(0) = \mathbf{0}$ .
28:        Obtain the final classical state  $\mathbf{h}_i(t_L)$ .
29:        Compute  $Q$ -value  $Q_i$  using  $\mathbf{h}_i(t_L)$  and (22).
30:        Compute critic loss  $L(\theta^Q)_{\text{QRC}}$  using (24).
31:        Update  $\theta^Q$  by minimizing  $L(\theta^Q)_{\text{QRC}}$ .
32:      Update Actor Network:
33:        Determine the policy action using the final state of each sequence.
34:        Compute the policy gradient using (26).
35:        Update actor parameters  $\theta^\pi$ .
36:      Soft Update Target Networks:
37:         $\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$ .
38:         $\theta^{\pi'} \leftarrow \tau \theta^\pi + (1 - \tau) \theta^{\pi'}$ .
39:      end if
40:       $s(t) \leftarrow s(t+1)$ .
41:    end for
42:  end for
43: Return trained policy  $\pi(s(t)|\theta^\pi)$ .

```

$\mathbf{u}(t) = [\mathbf{s}(t), \mathbf{a}(t)]$ is fed into the reservoir. The internal state of the reservoir, $\mathbf{z}_{\text{ESN}}(t) \in \mathbb{R}^{N_r}$, evolves according to the discrete-time equation which can be expressed as

$$\mathbf{z}_{\text{ESN}}(t) = (1 - \lambda)\mathbf{z}_{\text{ESN}}(t-1) + \lambda \tanh(\mathbf{W}_{\text{res}}\mathbf{z}_{\text{ESN}}(t-1) + \mathbf{W}_{\text{in}}\mathbf{u}(t)), \quad (27)$$

where λ is the leak rate. The reservoir state is initialized as $\mathbf{z}_{\text{ESN}}(0) = \mathbf{0}$ for each sampled sequence. The input weight matrix $\mathbf{W}_{\text{in}}$ and the internal reservoir weight matrix $\mathbf{W}_{\text{res}}$ are key to the ESN. They are initialized randomly and are not trained. The $\mathbf{W}_{\text{res}}$ matrix is scaled to have a spectral radius less than 1 [36]. This ensures the echo state property. This property allows the reservoir to maintain a stable memory of past inputs. After the reservoir processes an entire input sequence of length L , the trainable readout network produces the final Q -value. The readout network is a standard multi-layer perceptron. It takes a hybrid input vector. This vector is formed by concatenating two elements. One is the final hidden state of the reservoir, $\mathbf{z}_{\text{ESN}}(t_L)$ and other one is the final raw input of the sequence, $\mathbf{u}(t_L) = [\mathbf{s}(t_L), \mathbf{a}(t_L)]$. This combined vector

gives the readout both the compressed temporal history from $\mathbf{z}_{\text{ESN}}(t_L)$ and the immediate state-action information. Then, the readout maps this concatenated vector to the scalar Q -value, which can be expressed as

$$Q(\mathbf{S}_t, \mathbf{A}_t|\theta^Q)_{\text{ESN}} = F_{\text{read}}(\mathbf{z}_{\text{ESN}}(t_L) \parallel \mathbf{u}(t_L) \mid \theta_{\text{read}}). \quad (28)$$

Here, the ESN-Enhanced DDPG algorithm's training process is adapted for its recurrent critic network. It relies on a replay buffer that stores $(s(t), a(t), r(t), s(t+1))$ transitions. From this buffer, the agent samples mini-batches of entire sequences without crossing episode boundaries. This sequential data is necessary for the ESN critic to build its internal hidden state and learn from temporal patterns. The training process begins by updating the ESN critic network. It is important to note that this update only affects the critic's trainable parameters, which are the readout weights θ_{read} . Here the ESN reservoir weights remain fixed during the training. This critic network is updated using the target Q -value. Let $\mathbf{S}_t$ and $\mathbf{A}_t$ denote the sampled state and action sequences, respectively. Moreover, let $\mathbf{S}_{t+1}$ denote the corresponding next-state sequence and $\mathbf{A}'_{t+1}$ denote the target action sequence generated using π' . This Q -value can be calculated as [12]

$$y(t)_{\text{ESN}} = r(t) + \gamma Q'(\mathbf{S}_{t+1}, \mathbf{A}'_{t+1}|\theta^{Q'}), \quad (29)$$

where γ is the standard discount factor for future rewards. Moreover, $r(t)$ is the immediate reward from the end of the sequence. The $Q'(\mathbf{S}_{t+1}, \mathbf{A}'_{t+1}|\theta^{Q'})$ term provides the estimated future value, which is calculated by the target ESN critic network $Q'(\cdot|\theta^{Q'})$. To get this value, the target critic processes the full sequence of next states $\mathbf{S}_{t+1}$ using the target action sequence $\mathbf{A}'_{t+1}$. Thus, the ESN critic network $Q(\mathbf{S}_t, \mathbf{A}_t|\theta^Q)$ is then trained. The objective is to minimize the loss $L(\theta^Q)_{\text{ESN}}$, which is the mean squared error between the critic's prediction and the target $y(t)_{\text{ESN}}$.

$$L(\theta^Q)_{\text{ESN}} = \mathbb{E} \left[\left(y(t)_{\text{ESN}} - Q(\mathbf{S}_t, \mathbf{A}_t|\theta^Q) \right)^2 \right]. \quad (30)$$

Following the critic update, the actor network $\pi(\mathbf{s}(t)|\theta^\pi)$ is optimized. The actor's goal is to learn a policy that produces actions that the ESN critic evaluates as having a high Q -value. This update uses the policy gradient method. Since the actor network is non-recurrent, it receives only the final state $\mathbf{s}(t)$ from the sequence as input. It then produces the action $\mathbf{a}(t) = \pi(\mathbf{s}(t)|\theta^\pi)$. This action is used to create the sequence $\mathbf{A}_t^\pi$ for the critic to evaluate. Moreover, the actor network loss $L(\theta^\pi)_{\text{ESN}}$ is formulated as the negative expected Q -value from the critic which can be expressed as [12]

$$L(\theta^\pi)_{\text{ESN}} = -\mathbb{E} [Q(\mathbf{S}_t, \mathbf{A}_t^\pi|\theta^Q)]. \quad (31)$$

To update the actor network parameters θ^π , the algorithm computes the policy gradient $\nabla_{\theta^\pi} J(\theta^\pi)_{\text{ESN}}$ which can be expressed as [12]

$$\begin{aligned} \nabla_{\theta^\pi} J(\theta^\pi)_{\text{ESN}} = \mathbb{E} \left[\nabla_{a(t)} Q(\mathbf{S}_t, \mathbf{A}_t^\pi|\theta^Q) \Big|_{\mathbf{a}(t)=\pi(\mathbf{s}(t))} \right. \\ \left. \cdot \nabla_{\theta^\pi} \pi(\mathbf{s}(t) \mid \theta^\pi) \right]. \end{aligned} \quad (32)$$

Accordingly, the proposed ESN-Enhanced DDPG algorithm is provided in Algorithm 2.

Algorithm 2 Proposed ESN-Enhanced DDPG Algorithm for solving (13)

```

1: Initialize actor  $\pi(s|\theta^\pi)$  with weights  $\theta^\pi$ .
2: Initialize ESN critic  $Q(S, A|\theta^Q)$  with trainable readout parameters  $\theta^Q = \{\theta_{\text{read}}\}$ .
3: Initialize fixed ESN reservoir with  $W_{\text{in}}$  and  $W_{\text{res}}$ , ensuring spectral radius of  $W_{\text{res}} < 1$ .
4: Initialize target networks  $\pi'$  and  $Q'$  with  $\theta^{\pi'} \leftarrow \theta^\pi$  and  $\theta^{Q'} \leftarrow \theta^Q$ .
5: Set up replay buffer  $\mathcal{R}$  with capacity  $N_{\text{buffer}}$ .
6: Set sequence length  $L$  for temporal learning.
7: Initialize Ornstein-Uhlenbeck noise process  $\mathcal{O}$  for action exploration.
8: for episode = 1 to  $N_{\text{episodes}}$  do
9:   Reset noise process  $\mathcal{O}$ .
10:  Receive initial observation  $s(1)$  from environment.
11:  for  $t = 1$  to  $T_{\text{max}}$  do
12:    Determine action  $a(t)$  using policy and noise:  $a(t) = \pi(s(t)|\theta^\pi) + \mathcal{O}(t)$ .
13:    Map  $a(t)$  to the feasible action space through clipping and normalization.
14:    Apply action  $a(t)$  to the environment.
15:    Receive reward  $r(t)$  and the subsequent state  $s(t+1)$ .
16:    Append the transition tuple  $(s(t), a(t), r(t), s(t+1))$  to  $\mathcal{R}$ .
17:    if at least  $N_{\text{batch}}$  complete sequences of length  $L$  are available in  $\mathcal{R}$  then
18:      Sample a random mini-batch of  $N_{\text{batch}}$  contiguous sequences of length  $L$  from  $\mathcal{R}$  without crossing episode boundaries.
19:      for each sequence  $j$  in mini-batch do
20:        Extract state sequences and action sequences from  $\mathcal{R}$ .
21:        Construct next-state sequences from  $\mathcal{R}$ .
22:        Calculate target action sequence using  $\pi'$ .
23:        Compute target  $Q$ -value  $y_j(t)_{\text{ESN}}$  using (29).
24:      end for
25:      Update ESN Critic Network:
26:        Process sequences with (27).
27:        Estimate  $Q$ -values using (28).
28:        Compute the critic loss  $L(\theta^Q)_{\text{ESN}}$  using (30).
29:        Update  $\theta_{\text{read}}$  by minimizing  $L(\theta^Q)_{\text{ESN}}$ .
30:      Update Actor Network:
31:        Determine the policy action using the final state of each sequence.
32:        Calculate policy gradient using (32).
33:        Update actor parameters  $\theta^\pi$ .
34:      Soft Update Target Networks (Polyak average):
35:         $\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$ .
36:         $\theta^{\pi'} \leftarrow \tau \theta^\pi + (1 - \tau) \theta^{\pi'}$ .
37:      end if
38:       $s(t) \leftarrow s(t+1)$ .
39:    end for
40:  end for
41: Return trained policy  $\pi(s(t)|\theta^\pi)$ .

```

D. Complexity Analysis

The computational complexity of the proposed ESN-Enhanced DDPG algorithm is primarily driven by the classical ESN reservoir updates required for the critic and actor evaluations. Let B be the batch size, L be the sequence length, N_r be the reservoir size, and d_{in} be the input dimension ($d_s + d_a$). The trainable parameters for the actor and critic readout are N_a and N_c , respectively. Each training step involves a forward pass through the reservoir for L steps. This process is dominated by the recurrent matrix multiplication, which costs $O(BLN_r^2)$. The critic's gradient calculation applies only to the readout parameters and costs $O(BN_c)$. Moreover, since only the final action $\mathbf{a}(t_L)$ depends on the policy, the calculation of $\nabla_a Q$ requires backpropagation only through the final recurrent step and the readout. However, obtaining the final reservoir state still requires the forward unrolling of the reservoir over the L steps. Therefore, the forward pass remains the dominant

cost, $O(BLN_r^2)$. Thus, the per-step training complexity can be expressed as $O(BL(N_r^2 + N_r d_{\text{in}}) + B(N_a + N_c))$. Over E episodes and T steps, the total complexity can be denoted as $O(ET[BL(N_r^2 + N_r d_{\text{in}}) + B(N_a + N_c)])$.

Similarly, the computational complexity of the QRC-Enhanced DDPG algorithm arises from its hybrid classical-quantum structure. Let B be the batch size, L be the sequence length, n be the number of qubits, and L_q be the number of quantum layers. Let N_a be the number of actor parameters and $N_c = N_{\text{readout}}$ be the number of trainable critic readout parameters. All forward and backward encoder operations are captured by the per-step encoder cost C_{enc} . Therefore, $N_c = N_{\text{readout}}$ includes only the trainable readout parameters. Each step in the critic's forward pass involves a classical encoder pass with cost C_{enc} . It also involves quantum evolution, which costs $O(L_q n 2^n)$ in classical statevector simulation. Calculating the n single-qubit Pauli-Z expectation values requires an additional $O(n 2^n)$ operations. This measurement cost is included in $O(L_q n 2^n)$ for $L_q \geq 1$. This is the simulation setting considered in this work. These operations are repeated for BL steps. The gradient calculation for the readout parameters costs $O(BN_c)$.

Moreover, the encoder parameters θ_{ENC} require backpropagation through the quantum circuit at each of the BL steps. This backpropagation has a similar order of complexity to the forward pass. In addition, since only the final action $\mathbf{a}(t_L)$ depends on the policy, the calculation of $\nabla_a Q$ requires backpropagation only through the final quantum circuit and the readout. However, the forward pass over the L steps remains the dominant cost. Consequently, the per-step training complexity in classical statevector simulation is $O(BL(C_{\text{enc}} + L_q n 2^n) + B(N_a + N_c))$. Over E episodes and T steps, the total complexity can be calculated as $O(ET[BL(C_{\text{enc}} + L_q n 2^n) + B(N_a + N_c)])$. The higher-order encoding requires $O(n)$ gates, while the L_q quantum reservoir layers require $O(L_q n)$ gates. Therefore, the total gate count is $O(n(L_q + 1))$. For a fixed L_q , the gate count scales linearly with n . When the gates are applied sequentially, the circuit depth has the same upper bound. The current statevector simulation calculates the expectation values analytically and does not use a finite number of measurement shots. For a shot-based implementation with N_{shots} shots, each input requires repeated circuit executions. Processing the measured Pauli-Z values adds a cost of $O(BLN_{\text{shots}}n)$. The total training cost also depends on the gradient-estimation method.

In contrast, the ESN reservoir update scales quadratically with N_r when the reservoir weight matrix is treated as dense. Increasing N_r to improve the classical model can therefore become computationally expensive. On the other hand, the quantum-circuit gate count scales linearly with n for a fixed L_q . However, classical statevector simulation still scales exponentially with n , which restricts the present study to a small number of qubits. Nevertheless, the numerical results show that a small quantum reservoir with $n = 8$ qubits is sufficient to outperform a classical reservoir with $N_r = 500$ neurons. The quantum reservoir also uses far fewer reservoir features

as inputs to the readout. This demonstrates the effectiveness of the quantum reservoir for the considered setup. Therefore, the reported improvement represents a learning-performance gain for the considered classical simulation setting and does not indicate a quantum hardware speedup over the ESN.

In this work, we did not use real quantum hardware. In the current noisy intermediate-scale quantum (NISQ) era, training quantum DRL models on real quantum hardware remains challenging. Quantum devices have limited resources and are affected by hardware noise. In addition, cloud-based quantum devices have limited access, and submitted jobs can face long queue times. Quantum DRL training requires a large number of repeated circuit executions, which further increases the training time. Therefore, we implemented the quantum circuits using `TorchQuantum` and performed the simulations on a classical GPU. Thus, the reported results represent the learning performance of the QRC model under classical quantum-circuit simulation.

IV. NUMERICAL RESULTS AND DISCUSSIONS

TABLE I: Simulation parameters for 6G-enabled ISTN [4], [5], [12]

Parameter	Value
Number of BSs (M)	10
Number of antennas (N)	8
S1 orbit altitude (r_1)	300 km
S2 orbit altitude (r_2)	500 km
BS-S1 bandwidth (W_1)	20 MHz
S1-S2 bandwidth (W_2)	7 MHz
Uplink frequency	14 GHz
ISL frequency	20 GHz
BS-S1 noise power ($\sigma_{S_1}^2$)	-165 dBm
S1-S2 noise power ($\sigma_{S_2}^2$)	-174 dBm
Maximum BS transmit power ($P_m^{\max}$)	2.0 W
S1-S2 transmit power ($P_{S1,S2}$)	2.0 W
Model layers (L_1, L_2)	12/12
Model dimension (d_{model})	1024
Attention heads (n_h)	16
Feed-forward dimension (d_{ff})	4096
Bits per pixel (b)	8
Bits per token (b_{tok})	16
Bit precision (θ)	16
Computational efficiency (ρ_{S1}, ρ_{S2})	(0.9, 0.9)
Compression factor (η)	5.0
Maximum computational capacity ($F_{S1}^{\max}, F_{S2}^{\max}$)	(0.5, 0.25) TFLOPS
Maximum task latency ($T_m^{\max}$)	[58, 60] s
S1 energy budget (E_{S1}^{solar})	600 J
S2 energy budget (E_{S2}^{solar})	480 J
QKD refresh interval ($T_{\text{QKD}}^{\text{interval}}$)	2.0 s
Reward weights (w_1, w_2)	(0.5, 0.5)
Penalty weights ($\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5$)	(2, 2, 2, 5, 2)
DT Error	4%

A. Simulation Settings

This subsection details the simulation settings and hyperparameters used for the implementation and evaluation of the proposed frameworks. All models were trained using the Adam optimizer. Both ESN-Enhanced DDPG and QRC-Enhanced DDPG share identical core DDPG hyperparameters to ensure a fair comparison. The actor learning rate is set to 1×10^{-4} and the critic learning rate is 1×10^{-3} . The discount factor is $\gamma = 0.99$, and the soft update coefficient

for target networks is $\tau = 0.005$. For the ESN-Enhanced DDPG critic network, the classical ESN reservoir is constructed with $N_r = 500$ neurons. Following standard ESN design principles, the internal weight matrix W_{res} is initialized with a sparse connectivity of 1% and scaled to a spectral radius of 0.9 to ensure the echo state property [26], [36]. The input scaling is 0.1, and the leak rate α is set to 0.8. Moreover, for the QRC-Enhanced DDPG critic network, we consider quantum reservoir employs $n = 8$ qubits and comprises $L_q = 4$ layers which utilize HOE scheme described in (18). Unlike the standard ESN, this hybrid critic uses a classical leaky rate $\alpha = 0.8$ to integrate the quantum measurements over time. The training process for both algorithms is carried over $E = 1000$ episodes, with each episode lasting a maximum of $T = 100$ steps. The agents utilize a replay buffer $\mathcal{R}$ with a capacity of 100,000 transitions. For training, mini-batches of $B = 128$ sequences are sampled, with each sequence having a length of $L = 5$. Action exploration is managed by an Ornstein-Uhlenbeck noise process with $\theta = 0.15$ and $\sigma = 0.2$ [11]. Simulations were conducted in Python using PyTorch library. The quantum components were simulated using the TorchQuantum library on a system equipped with an NVIDIA GPU (CUDA) [37]. Furthermore, the 6G-enabled ISTN environment parameters are summarized in Table I.

B. Numerical Results

In this section, we comprehensively evaluate the performance of the proposed ESN-Enhanced DDPG and QRC-Enhanced DDPG algorithms, as well as the overall system performance. Moreover, the reported performance value for each algorithm was obtained by averaging the results over the final 100 training episodes.

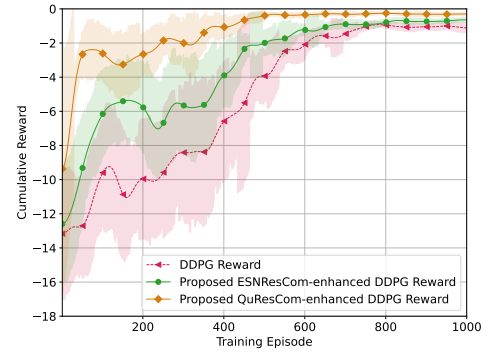


Fig. 3: Comparison of convergence performance of the proposed algorithms with the baseline.

1) *Convergence performance:* We evaluate the convergence behavior of the baseline DDPG algorithm in comparison with the proposed ESN-enhanced and QRC-enhanced DDPG frameworks, as illustrated in Fig. 3. All algorithms show a clear upward trend in cumulative reward as training progresses which indicates their ability to gradually learn improved policies over time. Among them, the baseline DDPG exhibits the slowest improvement and the highest variability. Although it initially

increases in reward, its learning curve fluctuates heavily during the first few hundred episodes and remains unstable throughout training. In contrast, the proposed ESN-Enhanced DDPG demonstrates faster and more consistent learning. Its reward values rise more steadily and maintain reduced variance compared with classical DDPG, reflecting improved temporal feature extraction and more stable gradient updates. As training continues, the ESN-based model sustains its advantage and converges to a higher performance level. The proposed QRC-Enhanced DDPG achieves the strongest convergence behavior overall. It rapidly improves during the early episodes and maintains a consistently higher reward across the entire training period. Furthermore, its variance band is the narrowest among all methods which shows highly stable learning dynamics. By the final stage of training, the QRC-enhanced model outperforms both DDPG and the ESN-enhanced variant that achieves the highest cumulative reward. Accordingly, these results demonstrate that the proposed reservoir-enhanced learning mechanisms significantly strengthen policy optimization.

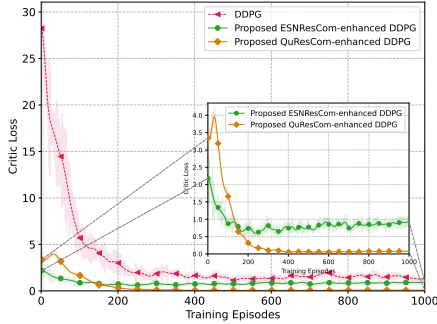


Fig. 4: Comparison of critic network loss of the proposed algorithms with the baseline.

2) *Critic network loss performance*: Fig. 4 illustrates the critic loss curves of DDPG, ESN-Enhanced DDPG, and QRC-Enhanced DDPG. All methods show a clear downward trend which indicate improved value-function estimation as training progresses. However, their convergence behaviors differ noticeably. The DDPG begins with the highest critic loss and requires a substantial number of episodes before reaching a stable region. Although the loss decreases over time, the curve exhibits strong fluctuations. These oscillations indicate unstable temporal-difference updates and higher sensitivity to noise in the critic network. Consequently, DDPG converges to a higher loss and shows weaker stability compared with the proposed approaches. The ESN-Enhanced DDPG achieves a faster reduction in loss and maintains smoother convergence. This reflects improved temporal state encoding and more consistent critic updates. Moreover, its loss stabilizes earlier which demonstrates enhanced robustness over classical DDPG. Furthermore, the QRC-Enhanced DDPG obtains the most efficient and stable convergence. After a brief initial rise linked to rapid exploration, the loss drops sharply and remains the lowest throughout training. Accordingly, both enhanced approaches outperform the baseline, whereas the QRC-enhanced model

provides the most reliable critic loss behavior across the entire training process.

3) *Average weighted system cost with varying transformer layers in satellite S_1 and satellite S_2 (L_1, L_2)*: To evaluate the impact of model depth, Fig. 5 presents the average weighted system cost for different encoder and decoder layer settings. When the transformer layers grows deeper, the cost rises consistently for all algorithms due to the additional computation required for each inference stage. The baseline DDPG

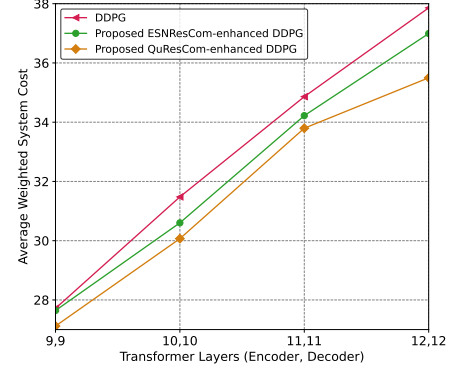


Fig. 5: Impact on average weighted system cost with varying transformer layers in satellite S_1 and satellite S_2 (L_1, L_2).

records the highest cost for every layer setting which indicates weaker adaptability when the model becomes deeper. The proposed ESN-Enhanced DDPG shows consistent improvements by lowering the system cost across all configurations. Furthermore, the proposed QRC-Enhanced DDPG maintains the lowest cost values throughout the entire depth range. Its advantage becomes slightly more visible as the transformer expands, showing that quantum-inspired reservoir computing helps the DRL agent respond more effectively to increasing computational complexity. By comparison, DDPG experiences the most pronounced increase in cost, whereas the proposed ESN and QRC versions handle deeper transformer settings with more stable and efficient performance.

4) *Average weighted system cost with varying system uplink bandwidth*: In this subsection, we analyze how the available uplink bandwidth influences the average weighted system cost for DDPG, the proposed ESN-Enhanced DDPG, and the proposed QRC-Enhanced DDPG, as shown in Fig. 6. As bandwidth increases from 10 to 40 MHz, all algorithms demonstrate a clear reduction in system cost. This behavior is expected because higher bandwidth reduces transmission delay and allows more efficient allocation of system resources. Across all bandwidth levels, the baseline DDPG exhibits the highest cost which indicates less efficient policy learning under varying channel conditions. The proposed ESN-Enhanced DDPG achieves lower costs by leveraging reservoir-based temporal feature extraction, which improves stability and responsiveness to bandwidth variations. Moreover, the proposed QRC-Enhanced DDPG consistently achieves the lowest cost across the entire bandwidth range. Its advantage becomes more pronounced at moderate-to-high bandwidths confirming,

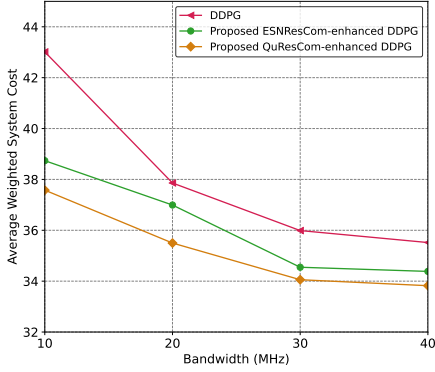


Fig. 6: Impact on average weighted system cost with varying system uplink bandwidth.

that quantum reservoir offers superior representational capacity for capturing complex system dynamics. In contrast, the proposed ESN-Enhanced DDPG and QRC-Enhanced DDPG methods maintain significantly lower system costs across all bandwidth levels which demonstrate their improved ability to handle bandwidth variability compared with the baseline DDPG.

5) *Average weighted system cost with varying transformer layers and DT error:* Fig.7 illustrates the average weighted system cost under different DT-error conditions as the number of transformer layers increases. As the model becomes deeper, the cost increases for all algorithms because the processing workload at satellites S_1 and S_2 becomes higher. The effect of the DT error differs across the algorithms. DDPG shows the

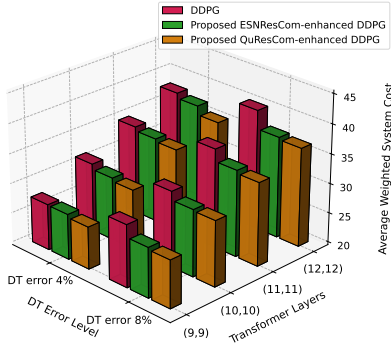


Fig. 7: Impact on average weighted system cost with varying transformer layers and DT error.

highest cost and the sharpest increase, while ESN-Enhanced DDPG achieves lower costs by using reservoir memory to capture temporal patterns. QRC-Enhanced DDPG consistently achieves the lowest cost and remains robust even when the DT error increases to 8%. This result is connected to the way the QRC critic processes the state information. The DT error changes the effective processing rates. This change affects the processing latency and energy values in the observation. The encoder and quantum reservoir first process these inputs. The leaky integrator then combines the extracted features over the sampled sequence and reduces sudden changes caused by

the DT error. The reservoir parameters remain fixed during training, so the quantum feature mapping remains unchanged. The higher-order encoding captures individual features and the relationships between neighboring feature pairs. The entangling circuit spreads the effect of a change in one feature across several quantum features. The measured single-qubit Pauli-Z values also remain within $[-1, 1]$, which limits the size of the feature changes. These properties help the QRC critic provide more stable value estimates under high DT-error conditions.

V. CONCLUSION AND FUTURE WORK

In this paper, we investigated a DT-enabled 6G NTN that supports distributed multimodal LLM inference across two LEO satellites with edge computing capability. The formulated NLP problem jointly optimizes uplink bandwidth allocation, uplink transmit power, and satellite computing resource allocation to minimize the total weighted system cost. To solve this challenging optimization problem, we transformed the problem into a constrained MDP and proposed two reservoir-assisted DDPG frameworks, namely ESN-Enhanced DDPG and QRC-Enhanced DDPG. Simulation results confirmed that both proposed algorithms achieve better learning stability and lower weighted system cost compared with the baseline DDPG. The ESN-Enhanced DDPG improves the critic estimation by capturing useful temporal patterns, which supports more reliable continuous control decisions. Moreover, the QRC-Enhanced DDPG provides the best overall performance among all compared methods, which indicates that the quantum reservoir-based critic can extract more informative features for value approximation. Consequently, the proposed framework provides an effective and scalable approach for latency- and energy-aware resource management in DT-enabled NTNs supporting multimodal LLM services.

For future work, this study can be extended in several directions. First, the framework can be expanded to multi-satellite and multi-orbit architectures. Moreover, different quantum reservoir circuit designs and entanglement patterns can be investigated to improve the representational power of the critic network and strengthen learning stability. Another direction could examine alternative quantum encoding strategies beyond the current higher-order encoding to better capture complex state relationships while keeping the model lightweight. Additionally, extending the proposed solution toward multi-agent learning can support coordinated decision-making among multiple satellites and ground nodes, which can improve scalability and enable more flexible resource allocation in large-scale 6G NTNs. Furthermore, lightweight visual encoders and model compression techniques, such as quantization and pruning, can be investigated to reduce preprocessing complexity in resource-constrained network environments.

REFERENCES

- [1] R. Sun *et al.*, "A comprehensive survey of knowledge-driven deep learning for intelligent wireless network optimization in 6G," *IEEE Commun. Surv. Tutorial.*, vol. 28, pp. 1099–1135, 2026.

- [2] P. Yu, J. Zhang, H. Fang, W. Li, L. Feng, F. Zhou, P. Xiao, and S. Guo, "Digital twin driven service self-healing with graph neural networks in 6G edge networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3607–3623, 2023.
- [3] Y. Usman, H. Oladipupo, A. D. During, R. Akl, and R. Chataut, "AI, ML, and LLM integration in 5G/6G networks: A comprehensive survey of architectures, challenges, and future directions," *IEEE Access*, vol. 13, pp. 168 914–168 950, 2025.
- [4] D. Narayanan *et al.*, "Efficient large-scale language model training on GPU clusters using Megatron-LM," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, St. Louis, Missouri, Nov. 14–19 2021.
- [5] X. Zhuang, J. Wu, H. Wu, T. Zhang, and L. Gao, "Joint optimization of model inferencing and task offloading for mec-empowered large vision model services," in *Proc. IEEE Conf. Comput. Commun.*, London, United Kingdom, May 19–22 2025, pp. 1–10.
- [6] T. T. Bui *et al.*, "Impact of 6G space-air-ground integrated networks on hard-to-reach areas: Tourism, agriculture, education, and indigenous communities," *EAI Endorsed Trans. on Tourism, Tech. and Intell.*, vol. 1, no. 1, pp. 1–8.
- [7] D. V. Huynh *et al.*, "Multi-agent reinforcement learning for optimal resource allocation in space-air-ground integrated networks," *IEEE Internet Things J.*, pp. 1–1, 2025.
- [8] W. Fan, Q. Meng, G. Wang, H. Bian, Y. Liu, and Y. Liu, "Satellite edge intelligence: DRL-based resource management for task inference in LEO-based satellite-ground collaborative networks," *IEEE Trans. Mobil. Comput.*, vol. 24, no. 10, pp. 10710–10728, 2025.
- [9] T. V. Nguyen, H. T. Le, H. T. T. Pham, V. Mai, and N. T. Dang, "Enhancing design and performance analysis of satellite entanglement-based CV-QKD/FSO systems," *IEEE Access*, vol. 11, pp. 112 097–112 107, 2023.
- [10] R. Zhao, J. Cai, J. Luo, J. Gao, and Y. Ran, "Demand-aware beam hopping and power allocation for load balancing in digital twin empowered LEO satellite networks," *IEEE Trans. Wireless Commun.*, vol. 24, no. 6, pp. 5084–5098, 2025.
- [11] S. C. Prabhashana, D. V. Huynh, H. Jung, B. Canberk, S. L. Cotton, and T. Q. Duong, "Quantum deep reinforcement learning for URLLC satellite-air-ground integrated networks with digital twin applications," *IEEE Internet Things J.*, vol. 13, no. 3, pp. 4230–4246, 2026.
- [12] S. C. Prabhashana *et al.*, "Machine learning-based resource allocation in 6G integrated space and terrestrial networks-aided intelligent autonomous transportation," *IEEE Trans. Intell. Transp. Syst.*, vol. 26, no. 10, pp. 17 750–17 762, 2025.
- [13] T. Jing, Y. Liu, X. Wang, and Q. Gao, "Deep echo state Q-network aided trust sharing provisioning for internet of vehicle," *IEEE Trans. Veh. Technol.*, vol. 73, no. 3, pp. 3993–4004, 2024.
- [14] S. C. Prabhashana, M.-H. T. Nguyen, V. Sharma, T. X. Vu, B. Canberk, H. Shin, and T. Q. Duong, "Quantum machine learning for wireless-powered UAV positioning in 6G digital twin SAGIN with cooperative nano-satellite constellations," *IEEE J. Sel. Areas Commun.*, vol. 44, pp. 5028–5042, 2026.
- [15] S. C. Prabhashana, D. V. Huynh, and T. Q. Duong, "Quantum DRL for UAV-RIS-aided maritime communications with 6G digital twin applications," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Montreal, Canada, Jun. 8–12 2025, pp. 396–401.
- [16] Silvrianti, B. Narottama, and S. Y. Shin, "UAV coverage path planning with quantum-based recurrent deep deterministic policy gradient," *IEEE Trans. Veh. Technol.*, vol. 73, no. 5, pp. 7424–7429, 2024.
- [17] S. Y.-C. Chen, "Efficient quantum recurrent reinforcement learning via quantum reservoir computing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Seoul, South Korea, Apr. 14–19 2024, pp. 13 186–13 190.
- [18] Y. Hu, X. Liu, G. Yang, L. Li, K. Zeng, Z. Zhao, S. Chen, L. Zhao, W. Li, and K. Li, "TightLLM: Maximizing throughput for LLM inference via adaptive offloading policy," *IEEE Trans. Comput.*, vol. 74, no. 7, pp. 2195–2209, 2025.
- [19] C. Liu and J. Zhao, "Enhancing stability and resource efficiency in LLM training for edge-assisted mobile systems," *IEEE Trans. Mob. Comput.*, vol. 25, no. 1, pp. 1–18, 2026.
- [20] X. Chen, Z. Guo, X. Wang, C. Feng, H. H. Yang, S. Han, X. Wang, and T. Q. S. Quek, "Toward 6G native-AI network: Foundation model-based cloud-edge-end collaboration framework," *IEEE Commun. Mag.*, vol. 63, no. 8, pp. 23–30, 2025.
- [21] X. He, Y. Jiang, X. Xu, H. Cui, Y. Liu, M. Chen, Y. Hong, and J. Zhang, "Large language model offloading using active inference in 6G symbiotic IoT," *IEEE Internet Things J.*, pp. 1–1, 2025.
- [22] M. Sun, J. Hou, K. Qiu, K. Wang, X. Chu, and Z. Zhang, "LLM-based task offloading and resource allocation in satellite edge computing networks," *IEEE Trans. Veh. Technol.*, pp. 1–6, 2025.
- [23] H.-H. Chang, H. Song, Y. Yi, J. Zhang, H. He, and L. Liu, "Distributive dynamic spectrum access through deep reinforcement learning: A reservoir computing-based approach," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1938–1948, 2019.
- [24] N. Mohammadi, L. Liu, and Y. Yi, "Policy-based fully spiking reservoir computing for multi-agent distributed dynamic spectrum access," in *Proc. IEEE Int. Conf. Commun.*, Seoul, South Korea, May 16–20 2022, pp. 1–6.
- [25] N. Mohammadi, L. Liu, Y. Song, and Y. Yi, "Energy-efficient dynamic and spatiotemporal spectrum access via spiking reservoir computing," *IEEE Trans. Wireless Commun.*, pp. 1–1, 2025.
- [26] J. Dai, S. Mahboob, H. Wang, and L. Liu, "Intelligent handover management enabled by O-RAN and deep reinforcement learning," in *Proc. IEEE Veh. Technol. Conf. (VTC-Fall)*, Washington, DC, May 16–20 2024, pp. 1–6.
- [27] B. Narottama, S. Aïssa, and Z. Mohamed, "AI-enabled framework for energy sustainability evaluation and enhancement of wireless networks," *IEEE Trans. on Green Commun. and Netw.*, pp. 1–1, 2025.
- [28] L. Domingo, "Classical and quantum reservoir computing: Development and applications in machine learning," *arXiv preprint arXiv:2310.07455*, 2023.
- [29] OpenAI, "tiktoken," GitHub repository, accessed: 10 Oct. 2025. [Online]. Available: <https://github.com/openai/tiktoken>
- [30] A. Dosovitskiy, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent.*, Vienna, Austria, May 04 2021.
- [31] Q. Dong, X. Xu, S. Han, R. Liu, and X. Zhang, "DDPG-based task offloading in satellite-terrestrial collaborative edge computing networks," in *Proc. IEEE Int. Conf. on Commun. Workshops*, Rome, Italy, 28 May – 01 Jun 2023, pp. 1541–1546.
- [32] S. Jung, S. Jeong, J. Kang, and J. Kang, "Marine IoT systems with space-air-sea integrated networks: Hybrid LEO and UAV edge computing," *IEEE Internet Things J.*, vol. 10, no. 23, pp. 20 498–20 510, 2023.
- [33] D. V. Huynh *et al.*, "Distributed communication and computation resource management for digital twin-aided edge computing with short-packet communications," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 10, pp. 3008–3021, 2023.
- [34] D. V. Huynh, S. R. Khosravirad, A. Masaracchia, O. A. Dobre, and T. Q. Duong, "Edge intelligence-based ultra-reliable and low-latency communications for digital twin-enabled metaverse," *IEEE Commun. Lett.*, vol. 11, no. 8, pp. 1733–1737, 2022.
- [35] A. Vaswani *et al.*, "Attention is all you need," in *Proc. 31st Int. Conf. on Neural Infor. Proc. Sys.*, Long Beach, California, Dec. 04–09 2017, pp. 6000–6010.
- [36] O. Onasami, M. Feng, H. Xu, M. Haile, and L. Qian, "Underwater acoustic communication channel modeling using reservoir computing," *IEEE Access*, vol. 10, pp. 56 550–56 563, 2022.
- [37] H. Wang, *et al.*, "Quantumnas: Noise-adaptive search for robust quantum circuits," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Seoul, South Korea, Feb. 02–06 2022, pp. 692–708.