Quantum Deep Q-Networks for Adaptive Resource Control in SAGIN-enabled Augmented Reality Systems

Dang Van Huynh*, Sasinda C. Prabhashana* Hyundong Shin†, Trung Q. Duong*,‡
* Memorial University, Canada, e-mail:{vdhuynh, cwelhengodag, tduong}@mun.ca
† Kyung Hee University, Korea, e-mail: hshin@khu.ac.kr
‡ Queen's University Belfast, UK.

***Abstract*—This paper presents a quantum deep Q-network (QDQN) approach for adaptive resource control in augmented reality (AR) systems supported by space-air-ground integrated network (SAGIN) architecture. Specifically, we jointly optimise the image sampling rates of ground cameras, task offloading decisions at unmanned aerial vehicles (UAVs), and bandwidth allocation to minimise a weighted multi-objective function of latency and bandwidth utilisation. The resulting problem is formulated as a *mixed-integer non-linear programming* (MINLP) problem, which poses significant computational challenges for conventional optimisation methods. By leveraging the expressibility of parameterised quantum circuits (PQCs) and the capability of advanced reinforcement learning, the proposed QDQN solution efficiently addresses the problem and enables adaptive decision-making. Simulation results demonstrate that the QDQN achieves significantly better training stability and faster convergence compared to the classical DQN approach. Notably, the proposed method reduces task delay by up to 100 times compared to conventional benchmarks, highlighting the substantial advantages of quantum-enhanced reinforcement learning in complex resource-constrained AR scenarios.**

A. Introduction

Space-air-ground integrated networks (SAGIN) technology is a key enabler of 6G networks, offering ubiquitous connectivity to support services in remote and hard-to-reach areas. SAGIN can robustly provide communication infrastructure for a wide range of applications across various domains, including environmental monitoring, smart oilfields, wildlife tracking, disaster management, and precision agriculture [1]. Enabling these applications, however, involves addressing numerous challenging problems that are gaining increasing attention in the research community. These challenges span from the physical layer to higher layers and include fundamental issues such as the optimal design of waveforms and modulation schemes, adaptive data scheduling, and resource management [2]. Recent efforts in this area have tackled problems such as wireless resource allocation in heterogeneous SAGIN environments [3], cost-efficient computation offloading [4], and energy-efficient cloud-edge collaboration [5]. Among the promising approaches, deep reinforcement learning has emerged as a key solution to address these complex and dynamic optimisation challenges [4]–[6].

Recently, SAGIN has been applied to support immersive applications such as augmented reality (AR), where a SAGIN-based mobile edge computing (MEC) architecture is leveraged to handle computationally intensive tasks. Unmanned aerial vehicles (UAVs) and low-Earth-orbit (LEO) satellites equipped with edge servers can provide high-throughput and low-latency services for AR systems. Advanced solutions—such as edge caching, adaptive task offloading, and related optimisation techniques—have been actively explored to realise these innovative systems [7]. MEC, with its inherent capability to offer near real-time and powerful computing resources, has been extensively adopted to support immersive applications such as AR and the metaverse [8], [9]. The integration of SAGIN and MEC technologies continues to open up unprecedented possibilities for deploying intelligent applications that enhance quality of life.

Motivated by the above discussions and the development of these emerging technologies, this study considers a SAGIN-enabled AR system by addressing a weighted multi-objective function problem of latency minimisation and bandwidth utilisation. The objective is to jointly optimise image sampling rates, bandwidth allocation, and task offloading decisions under queuing-aware latency and energy constraints. To efficiently solve the formulated mixed-integer non-linear programming problem, we propose a quantum deep Q-network (QDQN) solution that leverages the expressibility of parameterised quantum circuits (PQC) and the adaptability of reinforcement learning. Our contributions include the design of a realistic system model with parallel processing and queuing dynamics, a reinforcement learning formulation of the control problem, and the development of a QDQN framework that achieves superior delay performance and training efficiency compared to classical baselines.

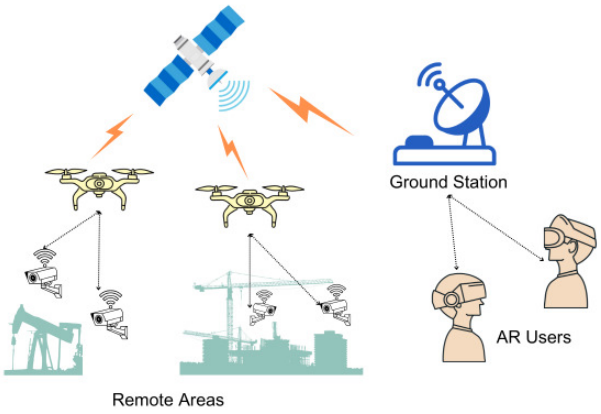

Fig. 1. An illustration of the SAGIN-enabled augmented reality systems.

I. System Model and Problem Formulation

A. Network Architecture and Working Flow

We consider a SAGIN-enabled AR system that is illustrated as in Figure 1. The system comprises five main components: a set of M ground-based IoT cameras that capture image and video data; K UAVs equipped with edge computing capabilities that serve as intermediate processing nodes for $\mathcal{M}_k$ IoT cameras in its network; a single satellite cloud providing additional computation and acting as a relay; ground gateway nodes (GGNs) that forward processed data; and AR users who interact with services based on the delivered results.

The system operates in two phases. In the data collection and processing phase, each IoT device captures images at a rate τ_m, transmits data to its associated UAV, which then either processes the task locally or offloads it to the satellite. In the AR user request phase, users retrieve processed data from GGNs, which access results stored at the satellite.

This work focuses on the first phase, aiming to jointly optimise the sampling rate τ_m, bandwidth allocation, and task offloading decisions at UAVs to minimise a weighted objective of latency and bandwidth utilisation under practical system constraints.

B. Image Collection and Computational Workload Models

Each IoT camera captures image data that must be processed and transmitted through the SAGIN system. The size and complexity of the captured data are influenced by several device-specific parameters: the resolution of the image in pixels ($H_m \times W_m$), the bit-depth per pixel (d_m), and the applied compression ratio (η_m). Based on these parameters, the data size per frame generated by IoT device m can be calculated as

$$D_m = H_m \times W_m \times d_m \times (1 - \eta_m). \tag{1}$$

To process each captured frame, a certain number of CPU cycles is required, depending on the complexity of the task. This requirement is captured by the computational workload per frame, which is given by $C_m = \gamma_m \times D_m$, where γ_m is the complexity factor of the task from the device m in terms of CPU cycles per bit.

C. Wireless Transmission Models

We consider a resource block (RB)-based orthogonal frequency division multiple access (OFDMA) scheme. The allocation of RBs directly affects the achievable transmission rate and consequently the latency of data transmission.

1) IoT-to-UAV transmissions (i2u): Since UAVs typically maintain a strong line-of-sight (LoS) link with IoT devices, the small-scale fading component for the IoT-to-UAV wireless channel is modeled using Rician fading as $h_{m,k} = \sqrt{\frac{K_R}{K_R+1}} h_{\text{LoS}} + \sqrt{\frac{1}{K_R+1}} h_{\text{NLoS}},$, where K_R denotes the Rician factor, h_{LoS} represents the deterministic LoS component, and h_{NLoS} is the non-line-of-sight complex Gaussian fading component. The wireless channel gain incorporating path-loss is defined as $g_{m,k} = |h_{m,k}|^2 / d_{m,k}^{\beta}$, where $d_{m,k}$ is the distance from IoT device m to UAV k, and β is the path-loss exponent.

The signal-to-noise ratio (SNR) for IoT-to-UAV communication is expressed as

$$\mathsf{SNR}_{m,k} = \frac{P_m g_{m,k}}{N_{m,k} B_0 N_0}, \tag{2}$$

where P_m is the transmit power of IoT camera m, $N_{m,k}$ is the number of RBs allocated to the IoT-to-UAV link, B_0 is the bandwidth per RB, and N_0 is the noise spectral density. As a result, the achievable transmission rate for the IoT-to-UAV link is calculated as

$$R_{m,k} = N_{m,k} B_0 \log_2(1 + \mathsf{SNR}_{m,k}), \tag{3}$$

leading to a transmission latency given by

$$T_{m,k}^{\text{i2u}} = \frac{D_m}{R_{m,k}}, \tag{4}$$

where D_m is the data size generated per frame by camera m.

2) UAV-to-Satellite transmissions (u2s): Each UAV forwards offloaded tasks from IoT devices to the satellite. The channel gain for the UAV-to-satellite link adopts the free-space path loss (FSPL) model with atmospheric attenuation effects:

$$h_k^{\mathsf{S}} = \left(\frac{c}{4\pi f_c d_{\text{ks}}} \right)^2 A_{\text{atm}}, \tag{5}$$

where c is the speed of light, f_c is the carrier frequency, d_{ks} represents the UAV-to-satellite distance, and A_{atm} accounts for atmospheric attenuation. The corresponding transmission rate from UAV k to the satellite is expressed as

$$R_k^{\mathsf{S}} = N_k^{\mathsf{S}} B_0 \log_2 \left(1 + \frac{P_k h_k^{\mathsf{S}}}{N_k^{\mathsf{S}} B_0 N_0} \right), \tag{6}$$

and the associated latency for each offloaded task from camera m is calculated as

$$T_m^{\text{u2s}} = \frac{D_m}{R_k^{\mathsf{S}}}. \tag{7}$$

D. Queuing-Aware Computation Models

Due to the limitations of computing capacity and energy budget of UAV, the UAV k have to decide if process the task m locally or offload it to satellite. To handle this, we introduce binary task offloading decision variables, $\boldsymbol{\alpha} = \{\alpha_{m,k}\}_{\forall m,k} \in \{0,1\}$ as follows:

$$\alpha_{m,k} = \begin{cases} 1, & \text{the task is processed at the } k\text{-th UAV}, \\ 0, & \text{the task is offloaded the the satellite.} \end{cases} \tag{8}$$

1) Queuing delay of task processing at UAVs: We apply M/M/1 queuing model to calculate the processing delay of the offloaded tasks at UAVs, which can be modelled as

$$T_k^{\mathsf{q}} = \frac{1}{\mu_k - \lambda_k}, \tag{9}$$

where $\lambda_k = \sum_{m \in \mathcal{M}_k} \tau_m$ is the task arrival rate and $\mu_k = f_k / \sum_{m \in \mathcal{M}_k} \alpha_{m,k} C_m$ is the UAV processing rate to execute all tasks in its network with respect to the offloading

decision α_m.

2) Queuing delay of task processing at the satellite: The queuing delay of task processing at the satellite is also modelled similarly as at the UAVs, which is expressed as

$$T_S^{\mathsf{q}} = \frac{1}{\mu_S - \lambda_S}, \tag{10}$$

where $\lambda_S = \sum_k \sum_{m \in \mathcal{M}_k} (1 - \alpha_{m,k}) \tau_m$ is the task arrival rate at the satellite and $\mu_S = f_S \big/ \sum_k \sum_{m \in \mathcal{M}_k} (1 - \alpha_{m,k}) C_m$ is the satellite processing rate to execute all the offloaded tasks.

E. Latency Model

The total latency of a task coming from the camera m consists of image collection time, transmission time from the camera m to the UAV k, queuing delay for processing at the UAV k or offloading delay from the UAVs to satellite and satellite's processing delay. As a result, the total delay for data collection and processing phase considered in this system model is calculated as

$$T_m^{\text{tot}} = T_m^{\text{col}} + T_{m,k}^{\text{i2u}} + T_k^{\mathsf{q}} + (1 - \alpha_{m,k}) \big(T_{m,k,S}^{\text{u2s}} + T_S^{\mathsf{q}} \big) \tag{11}$$

where $T_m^{\text{col}} = 1/\tau_m$ is the image collection delay and other delay components are calculated as in (4), (9), (7), and (10).

F. UAV Energy Consumption Model

To handle the computational tasks generated by IoT cameras, UAVs consume energy for both wireless transmission and local task execution, depending on the offloading decisions. Accordingly, the total energy consumption of UAV k is modelled as $E_k = E_k^{\text{comm}} + E_k^{\text{comp}}$ where $E_k^{\text{comm}} = P_k^{\text{u2s}} \sum_m^{\mathcal{M}_k} \alpha_{m,k} T_{m,k,S}^{\text{u2s}}$ is the energy for wireless transmissions and $E_k^{\text{comp}} = \kappa f_k^2 \sum_m^{\mathcal{M}_k} \alpha_{m,k} C_m$ is the energy for task execution; here, κ is the parameter of execution energy. We note that, in this work, the energy consumed for UAV flight and hovering is not considered, as our focus is on communication and computation-related energy costs.

G. Optimisation Problem Formulation

The addressed optimisation problem of this paper is formulated as follows:

$$\min_{\boldsymbol{\alpha}, \mathbf{N}, \boldsymbol{\tau}} \quad w_1 \sum_m T_m^{\text{tot}} + w_2 \sum_k \left(\frac{\sum_m N_{m,k} + N_k^S}{N_{\text{tot}}} \right) \tag{12a}$$

$$\text{s.t.} \sum_{m \in \mathcal{M}_k} N_{m,k} + N_k^S \le N_{\text{tot}}, \quad \forall k, \tag{12b}$$

$$E_k \le E_k^{\max}, \quad \forall k, \tag{12c}$$

$$\lambda_k < \mu_k, \quad \forall k, \tag{12d}$$

$$\lambda_S < \mu_S, \tag{12e}$$

$$\boldsymbol{\alpha} \in \mathcal{A}, \mathbf{N} \in \mathcal{N}, \boldsymbol{\tau} \in \mathcal{T}. \tag{12f}$$

In (12), we aim at minimising the total delay of all tasks generated by IoT cameras and the bandwidth utilisation ratio (12a). Constraint (12b) shows the RB allocation constraint of each UAV. Constraint (12c) reflects the energy budget of UAV k. Constraints (12d) and (12e) represent the queuing stability requirements for task processing. Lastly, constraint (12f) gives the discrete selection options of the optimisation variables.

II. Proposed Solution

As presented in (12), the formulated problem is a mixed-integer non-linear programming (MINLP) problem, which is NP-hard and computationally intractable using conventional optimisation techniques. Leveraging advanced reinforcement learning algorithms, such problems can be addressed more efficiently through *data-driven exploration* and decision-making. Furthermore, quantum circuits, with their inherent *expressibility*, are well-suited to capturing the complex non-linear patterns characteristic of MINLP problems. Motivated by these capabilities, we propose a quantum deep Q-network (DQN) solution to tackle the formulated problem. To enable this, we first reformulate the original problem (12) into a reinforcement learning framework, as described in the following subsection.

A. Reinforcement Learning Transformation

1) State space: The RL agent observes the system through an aggregated state vector $\mathbf{x} \in \mathbb{R}^d$ (with $d = 3$ in our design), defined as $\mathbf{x} = [x_1, x_2, x_3]$, where x_1 represents the total delay across all IoT cameras, which includes the image collection, transmission, queuing, and offloading delays as defined in (11); x_2 denotes the bandwidth usage across UAVs, computed as the ratio of the sum of allocated resource blocks (RBs) to the total available RBs; x_3 captures the total energy consumption across all UAVs.

2) Action space: The action space is formulated as a multi-discrete set to control the decisions for both cameras and UAVs. We define the following discrete option sets:

- $\mathcal{T}$: the set of available sampling rates for each camera,
- $\mathcal{A}$: the set of offloading decisions,
- $\mathcal{N}_{\text{IoT}}$: the set of RB allocation options for the IoT-to-UAV link for each camera,
- $\mathcal{N}_{\text{UAV}}$: the set of RB allocation options for the UAV-to-satellite link for each UAV.

Then, for each camera m and each UAV k, an action is given by $a = \big(a_m^\tau, a_m^\alpha, a_m^N, a_k^N\big)$, with $a_m^\tau \in \mathcal{T},\ a_m^\alpha \in \mathcal{A},\ a_m^N \in \mathcal{N}_{\text{IoT}},\ a_k^N \in \mathcal{N}_{\text{UAV}}$, which is presented in (12f).

3) Reward design: The reward function is designed to encourage policies that minimise both the total delay and the resource usage while penalising any violation of system constraints. Mathematically, the reward is defined as

$$r = -\Big(w_1\, x_1 + w_2\, x_2 + P(x_1, x_2, x_3) \Big), \tag{13}$$

where w_1 and w_2 are weighting coefficients that balance the relative importance of delay and bandwidth usage; x_1 is the total delay, and x_2 is the normalised bandwidth usage, $P(x_1, x_2, x_3)$ is a penalty function that accounts for violations of system constraints in (12).

For example, consider the bandwidth constraint for each UAV k, i.e., (12b), which requires

$$\sum_{m \in M_k} a_m^N + a_k^N \le N_{\text{tot}},\ \forall k, \tag{14}$$

where N_{tot} denotes the total available RBs per UAV. If this constraint is violated, we impose a penalty

$$P_{\text{bw}} = \lambda_{\text{bw}} \sum_{k} \max\Big\{0, \Big(\sum_{m \in M_k} a_m^N + a_k^N - N_{\text{tot}} \Big) \Big\}, \quad (15)$$

with λ_{band} as the penalty coefficient. Penalty terms for energy or latency violations are conducted similarly.

B. Proposed Quantum Deep Q-Network (QDQN) Solution

1) Quantum state encoding via the parameterised quantum circuit (PQC): Given a classical state $\mathbf{x} \in \mathbb{R}^d$ (with $d = 3$ in our case), our PQC acts on Q qubits and is structured in L layers. To enhance the data encoding step, we adopt data-reuploading and trainable scaling techniques for data encoding in the PQC [10]. As a result, the quantum embedding operation of a single layer in the PQC is expressed as

$$U_{\text{Q-DQN}}^{(l)}(\mathbf{x}) = \underbrace{\bigotimes_{q=l}^{Q} U_{\text{encode}}^{(l,q)}\big(\mathbf{x};\, \theta_{l,q}^{(y)},\, \theta_{l,q}^{(z)},\, \beta_q\big)}_{\text{Data Encoding \& Trainable Rotations}} \times \underbrace{\left(\prod_{q=1}^{Q-1} \text{CZ}\big(q,\, q+1\big) \right)}_{\text{Daisy-Chain Entangling}}. \quad (16)$$

where the encoding operation on the q-th qubit is defined as a unitary operation, given by

$$\begin{aligned} &U_{\text{encode}}^{(l,q)}\big(\mathbf{x};\, \theta_{l,q}^{(y)},\, \theta_{l,q}^{(z)},\, \beta_q\big) \\ &\quad = \underbrace{R_y\Big(\arctan\big(x_q\, \beta_q\big)\Big) R_z\Big(\arctan\big(x_q\, \beta_q\big)\Big)}_{\text{Data Encoding}} \\ &\quad \times \underbrace{R_y\big(\theta_{l,q}^{(y)}\big) R_z\big(\theta_{l,q}^{(z)}\big)}_{\text{Trainable Rotations}}, \end{aligned} \quad (17)$$

where β_q is the trainable scaling parameter. By applying the encoding scheme as (17), we achieve a *universal single-qubit rotation* that injects the classical data $\mathbf{x}$ (scaled by β_q) into the qubit and then applies additional trainable rotations. This structure is fundamental to *data re-uploading*, where each qubit repeatedly encodes classical information throughout multiple layers or at multiple points in the circuit, helping the quantum model learn richer representations. For multiple layers, the overall PQC is expressed as $U_{\text{Q-DQN}}(\mathbf{x}) = \prod_{l=1}^{L} U_{\text{Q-DQN}}^{(l)}(\mathbf{x})$, with $U_{\text{Q-DQN}}^{(l)}(\mathbf{x}) = U_{\text{ent}}^{(l)} \cdot U_{\text{encode}}^{(l)}(\mathbf{x})$, and $U_{\text{ent}}^{(l)} = \prod_{q=1}^{Q-1} \text{CZ}(q,\, q+1)$. is the entangling operation in each layer.

2) Measurement and post-processing: After the application of the PQC, the quantum state is measured in the computational (Z) basis, yielding a measurement outcome given by

$$\mathbf{z} \in \{-1, +1\}^Q. \quad (18)$$

This measurement vector is then processed by a classical fully connected (FC) layer:

$$\mathbf{o} = W\, \mathbf{z} + b, \quad (19)$$

where $W \in \mathbb{R}^{T \times Q}$ and $b \in \mathbb{R}^T$. The total output dimension is defined as

$$T = \sum_{i=1}^{H} \Big(n_{\text{decisions}}^{(i)} \times n_{\text{options}}^{(i)} \Big). \quad (20)$$

The output vector $\mathbf{o}$ is partitioned into H segments. For each head i, the corresponding segment is reshaped into a matrix:

$$\mathbf{o}^{(i)} \in \mathbb{R}^{n_{\text{decisions}}^{(i)} \times n_{\text{options}}^{(i)}}, \quad (21)$$

where each element $\mathbf{o}_{j,k}^{(i)}$ represents the estimated Q-value for the jth decision taking the kth option in head i.

3) Q-Value computation: For a composite action

$$a = \{a^{(i)} \mid i = 1, \ldots, H\}, \quad (22)$$

where for each head i the decision j selects an option $a_j^{(i)} \in \{1, \ldots, n_{\text{options}}^{(i)}\}$, the overall Q-value is computed as

$$Q(\mathbf{x}, a) = \sum_{i=1}^{H} \sum_{j=1}^{n_{\text{decisions}}^{(i)}} \mathbf{o}_{j,\, a_j^{(i)}}^{(i)}. \quad (23)$$

4) Training via temporal difference (TD) learning: For a transition tuple $(\mathbf{x}, a, r, \mathbf{x}', d)$ from the environment (with $d \in \{0, 1\}$ indicating whether $\mathbf{x}'$ is terminal), the TD target is defined using a target network Q_{target} as

$$y = r + \gamma\, (1 - d)\, \max_{a'} Q_{\text{target}}(\mathbf{x}', a'), \quad (24)$$

where $\gamma \in [0, 1)$ is the discount factor.

The loss function is then given by the mean-squared error (MSE) between the predicted Q-value and the target:

$$\mathcal{L}(\theta) = \mathbb{E}_{(\mathbf{x}, a, r, \mathbf{x}', d)} \left[\Big(Q(\mathbf{x}, a; \theta) - y \Big)^2 \right], \quad (25)$$

where the parameter set θ includes the PQC parameters i.e., the trainable input scaling factors $\{\beta_q\}$ and rotation angles $\{\theta_{l,q}^{(y)},\, \theta_{l,q}^{(z)}\}$, and the classical FC layer parameters: W and b for post-processing. These parameters are updated via gradient descent (using, e.g., the Adam optimiser) to minimise $\mathcal{L}(\theta)$, and the target network is updated periodically to stabilize the training process.

C. Proposed Algorithm

Based on above development, the proposed the QDQN algorithm to solve the problem (12) is summarised as presented in Algorithm 1.

III. Simulation Results and Discussions

A. Simulation Settings

To evaluate the performance of the proposed solution, we conduct simulations using the parameter settings as follows [3], [4], [9]. The system consists of $M = 10$ IoT cameras and $K = 2$ UAVs. Each IoT device captures images at a sampling rate selected from the set $\mathcal{T} = \{20, 25, 30\}$ fps. The task offloading decision is binary, represented by $\mathcal{A} = \{0, 1\}$,

Algorithm 1 : Proposed QDQN algorithm for solving (12).

Input: Environment Env, QDQN network $Q(\mathbf{x}, a; \theta)$, target network $Q_{\text{target}}(\mathbf{x}, a; \theta^-)$, replay buffer $\mathcal{D}$, discount factor γ, exploration parameters ϵ, batch size, target update frequency.
Initialise parameters θ, set target parameters $\theta^- \leftarrow \theta$
for each episode **do**
  Initialise State $\mathbf{x} \leftarrow$Env.reset()
  while not terminal **do**
    Quantum Encoding and Measurement:
      Compute measurement:

$$\mathbf{z} \leftarrow \text{Measure}\Big(\prod_{l=1}^{L} U^{(l)}_{\text{Q-DQN}}(\mathbf{x})\,|0\rangle^{\otimes Q}\Big) \in \{-1, +1\}^Q$$

    Post-Processing:
      Compute output:

$$\mathbf{o} \leftarrow W\,\mathbf{z} + b$$

      Partition $\mathbf{o}$ into segments $\{\mathbf{o}^{(i)}\}_{i=1}^{H}$ to obtain Q-value estimates $Q(\mathbf{x}, a; \theta)$
    Action Selection: Choose action a using an ϵ-greedy policy based on $Q(\mathbf{x}, a; \theta)$
    Execute action a in Env, observe reward r, next State $\mathbf{x}'$, and terminal flag d
    Store transition $(\mathbf{x}, a, r, \mathbf{x}', d)$ in $\mathcal{D}$
    Update State: $\mathbf{x} \leftarrow \mathbf{x}'$
    if size of $\mathcal{D} >$ batch size **then**
      Sample minibatch $\{(\mathbf{x}_i, a_i, r_i, \mathbf{x}'_i, d_i)\}$ from $\mathcal{D}$
      Compute current Q-values: $Q(\mathbf{x}_i, a_i; \theta)$
      Compute target using the target network:

$$y_i = r_i + \gamma\,(1 - d_i)\,\max_{a'} Q_{\text{target}}(\mathbf{x}'_i, a')$$

      Update parameters θ by minimizing the loss:

$$\mathcal{L}(\theta) = \mathbb{E}\Big[\big(Q(\mathbf{x}_i, a_i; \theta) - y_i\big)^2\Big]$$

    end if
  end while
  if episode mod (target update frequency) $= 0$ **then**
    Update target network: $\theta^- \leftarrow \theta$
  end if
end for
Return: Trained QDQN model with optimal actions.

where 1 indicates local processing at the UAV and 0 indicates offloading to the satellite. Resource block (RB) allocation between IoT-to-UAV and UAV-to-satellite links is selected from $\mathcal{N} = \{1, 2, 3, 4, 5\}$, subject to a maximum of $N_{\text{total}} = 10$ RBs per UAV. Each RB has a bandwidth of $B_{\text{RB}} = 180$ kHz, and the noise spectral density is set to $N_0 = -174$ dBm/Hz. Both IoT devices and UAVs transmit with a power of 30 dBm. The processing frequencies of the UAVs and the satellite are set to $f_{\text{uav}} = 4 \times 10^9$ cycles/s and $f_{\text{sat}} = 20 \times 10^9$ cycles/s, respectively. The energy consumption model uses an energy coefficient $\kappa = 10^{-27}$. The UAV-to-satellite distance is fixed at $d_{ks} = 500$ km, and the wireless channel model incorporates a path loss exponent $\beta = 3.0$ and a Rician factor $K_R = 3.0$, which reflects the presence of line-of-sight components in the IoT-to-UAV communication links.

Regarding model training, we adopt standard hyperparameter settings commonly used in the DQN algorithm, as summarised in Table I. It is important to note that the same configuration is applied to both the quantum and classical approaches to ensure a fair comparison of training performance—this includes parameters such as the maximum number of steps, epsilon decay schedule, replay buffer capacity, and discount factor. The implementation of the proposed solution and the simulations are conducted in a Python environment, using well-known libraries such as `PyTorch`, `NumPy`, `Gymnasium`, and `TorchQuantum` [11].

TABLE I
HYPERPARAMETER SETTINGS.

Parameters	Value
Max steps	100
Batch size	64
Discount factor	0.99
Learning rate	0.001
Update target model	10
Initial ϵ	1
Final ϵ	0.01
Decay of ϵ	0.99
Replay capacity	10000

B. Numerical Results

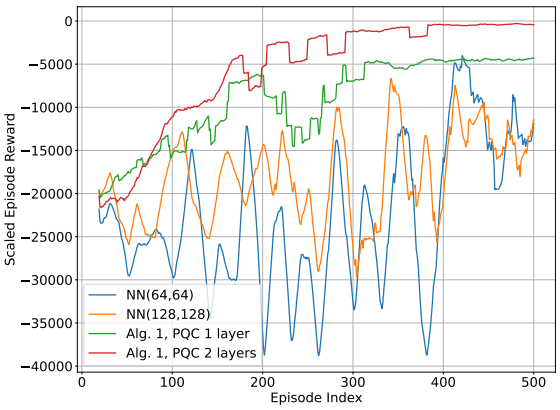

Fig. 2. Training performance of the proposed algorithm compared to the classical DQN algorithm with varying network size.

1) Training performance of the proposed algorithm: Figure 2 illustrates the training performance of the proposed quantum deep Q-learning algorithm compared to classical DQN models with different network sizes. As shown, the quantum-enhanced agents achieve significantly better stability and faster

TABLE II
COMPARISON OF TRAINABLE PARAMETERS.

Architecture	Number of Parameters
Classical DQN (64,64)	$11,566$
Classical DQN (128,128)	$31,214$
Quantum DQN with 1 Layer	449
Quantum DQN with 2 Layers	455

convergence. In particular, the quantum DQN with two PQC layers consistently outperforms all baselines, achieving the highest average reward and the most stable training behaviour. Even with a single PQC layer, the quantum model demonstrates superior convergence compared to classical DQNs. This improvement is especially notable given the number of trainable parameters reported in Table II, where the quantum models have dramatically fewer parameters—only 449 and 455—compared to $11,566$ and $31,214$ for classical DQNs with $(64, 64)$ and $(128, 128)$ hidden layers, respectively. These results highlight the representational efficiency and learning effectiveness of quantum models, which are able to explore and exploit the solution space more effectively with fewer resources.

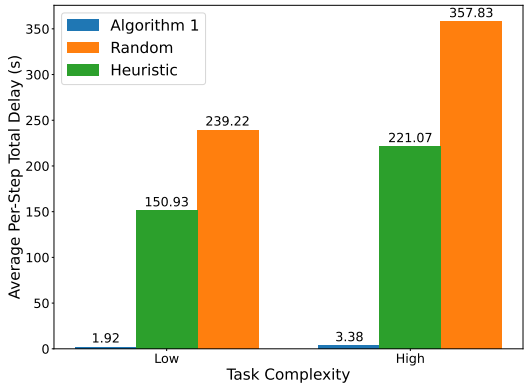

Fig. 3. The comparison of total delay achieved by Algorithm 1 and other conventional benchmarks with different settings of the task complexity.

2) Average delay comparison: Figure 3 presents the average per-step total delay under different task complexity levels for three approaches: the proposed Algorithm 1, a heuristic baseline, and a random policy. Task complexity is controlled by varying the image resolution and computational workload, where low complexity corresponds to a resolution range of 400×400 to 500×500 pixels and a processing intensity γ in the range of $[0.5, 1.0]$, while high complexity increases these ranges to 500×500 to 600×600 pixels and $\gamma \in [1.0, 1.5]$. Across both low and high complexity settings, the quantum-based Algorithm 1 consistently achieves significantly lower delays compared to the baseline methods. Under high complexity, Algorithm 1 maintains its advantage, achieving delays that are about 100 times lower than random and over 65 times lower than heuristic. These findings highlight the effectiveness and robustness of Algorithm 1 in handling varying levels of task complexity, thanks to its ability to adaptively allocate resources and minimise queuing and transmission latency.

IV. CONCLUSION

In this paper, we have investigated a QDQN solution for the optimal design of image sampling rates, task offloading decisions, and bandwidth allocation in SAGIN-enabled AR systems. By leveraging the expressive power of PQC and an efficient reinforcement learning algorithm, the proposed solution achieved significantly faster convergence and effectively minimised total delay in the considered system. Looking ahead, jointly optimising UAV path planning with resource control represents a promising direction for developing optimal designs in dynamic environments.

ACKNOWLEDGMENTS

This work 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, and in part by the NSERC CREATE program (Grant number 596205-2025)

REFERENCES

[1] Q. Chen, Z. Guo, W. Meng, S. Han, C. Li, and T. Q. S. Quek, "A survey on resource management in joint communication and computing-embedded SAGIN," *IEEE Commun. Surveys Tuts.*, vol. 27, no. 3, pp. 1911–1954, 2025.

[2] Y. Xiao, Z. Ye, M. Wu, H. Li, M. Xiao, M.-S. Alouini, A. Al-Hourani, and S. Cioni, "Space-air-ground integrated wireless networks for 6 : Basics, key technologies, and future trends," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 12, pp. 3327–3354, Dec. 2024.

[3] J. He, N. Cheng, Z. Yin, C. Zhou, H. Zhou, W. Quan, and X.-H. Lin, "Service-oriented network resource orchestration in space-air-ground integrated network," *IEEE Trans. Veh. Technol.*, vol. 73, no. 1, pp. 1162–1174, Jan. 2024.

[4] Y. Gao, Z. Ye, and H. Yu, "Cost-efficient computation offloading in SAGIN: A deep reinforcement learning and perception-aided approach," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 12, pp. 3462–3476, Dec. 2024.

[5] Z. Wang, L. Zhang, D. Feng, G. Wu, and L. Yang, "Intelligent cloud-edge collaborations for energy-efficient user association and power allocation in space-air-ground integrated networks," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 12, pp. 3659–3673, Dec. 2024.

[6] P. Zhang, N. Chen, S. Shen, S. Yu, N. Kumar, and C.-H. Hsu, "Ai-enabled space-air-ground integrated networks: Management and optimization," *IEEE Netw.*, vol. 38, no. 2, pp. 186–192, Apr. 2024.

[7] S. Yoo, S. Jeong, J. Kim, and J. Kang, "Cache-assisted mobile-edge computing over space–air–ground integrated networks for extended reality applications," *IEEE Internet of Things J.*, vol. 11, no. 10, pp. 18 306–18 319, May 2024.

[8] J. Feng and J. Zhao, "Resource allocation for augmented reality empowered vehicular edge metaverse," *IEEE Trans. Commun.*, vol. 73, no. 3, pp. 1987–2001, Mar. 2025.

[9] W. Qian and R. W. L. Coutinho, "Load-aware orchestrator for edge-computing-aided wireless augmented reality," *IEEE Internet of Things J.*, vol. 12, no. 6, pp. 6595–6606, Mar. 2025.

[10] A. Skolik, S. Jerbi, and V. Dunjko, "Quantum agents in the gym: A variational quantum algorithm for deep Q-learning," *Quantum*, vol. 6, p. 720, 2022.

[11] H. Wang, Y. Ding, J. Gu, Z. Li, Y. Lin, D. Z. Pan, F. T. Chong, and S. Han, "Quantumnas: Noise-adaptive search for robust quantum circuits," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Seoul, Korea, 2022.