IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Journal Européen des Systèmes Automatisés
Vol. 59, No. 10, October, 2026, pp. 3043-3054
Journal homepage: http://iieta.org/journals/jesa

Multi-Objective Optimisation Framework for Adaptive Resource Allocation in IoT Networks: A Comparative Analysis of Three Network Architectures

Muthanna Jaafar Abbas[1*], Haider Mohammed Turki Alhilfi[2]

[1] Department of Electrical Engineering, Faculty of Engineering, University of Babylon, Babylon 51002, Iraq
[2] Ministry of Education/General Directorate for Vocational Education, Hilla 51001, Iraq

Corresponding Author Email: eng.muthanna.j@uobabylon.edu.iq

Copyright: ©2026 The authors. This article is published by IIETA and is licensed under the CC BY 4.0 license (http://creativecommons.org/licenses/by/4.0/).

https://doi.org/10.18280/jesa.591023

Received: 23 July 2026
Revised: 27 August 2026
Accepted: 17 September 2026
Available online: 31 October 2026

Keywords:
adaptive weighting, Deep Q-Network, Internet of Things, multi-objective optimisation, network architectures, resource allocation

ABSTRACT

This paper presents a new multi-objective optimisation (MOO) framework for adaptive resource allocation in Internet of Things (IoT) networks. In this work, we do not treat energy consumption, latency, and data accuracy as separate objectives. Instead, we jointly optimise them as conflicting factors to meet different quality-of-service (QoS) requirements. We propose a Deep Q-Network (DQN) agent to dynamically adjust the optimisation weights based on both real-time network conditions and application requirements. The framework is tested on three typical IoT network architectures: a dense wireless sensor network (WSN), an edge-supported hierarchical network, and a cloud-fog hybrid network. Extensive MATLAB simulations show that the proposed adaptive weighting approach reduces energy consumption by 31% compared to the static baseline, while achieving 28% lower latency and 19% higher data accuracy. We further characterise the critical trade-offs among energy, latency, and accuracy for each architecture, underscoring the importance of architecture-aware optimisation. As our results show, the DQN-based approach performs much better than static weighting policies consistently and attains Pareto-optimal solutions.

1. INTRODUCTION

Over the last decade, the number of connected devices has skyrocketed, and experts anticipate over 75 billion connected devices by 2026 [1]. Scalability will become a significant issue as the number of connected devices increases, especially for applications that require some degree of real-time control and for those constrained by energy resources [2]. This will be exacerbated by the emergence of 6G networks, which create more stringent requirements for real-time analytics, ultra-reliable communications, and efficient management of diverse resources across multiple network types [3, 4].

The topic of optimisation for the Internet of Things (IoT) is vast and has received extensive research. This includes dynamic power management [5], the use of renewable energy sources [6], and optimising IoT systems and RF energy harvesters using AI- based methods [7]. Additionally, some studies have suggested federated learning as a way to train models in a decentralised manner within IoT networks, helping to preserve privacy and reduce communication costs [8-10]. In addition, digital twin (DT) technology has been explored as a promising approach to enhance both the optimisation and security of industrial IoT systems [11-13].

However, this body of literature exhibits a significant gap concerning recent advancements (2024–2025) in Deep Reinforcement Learning (DRL) for dynamic, multi-objective resource allocation. While these works address specific sub-domains such as energy harvesting or privacy, they largely rely on static or single-objective optimisation strategies that fail to capture real-time trade-offs among energy, latency, and accuracy under fluctuating network conditions. For instance, recent high-impact contributions in the IEEE IoT Journal have demonstrated the superiority of DRL-based schedulers in adapting to heterogeneous IoT environments. A DRL-based joint service migration and resource allocation framework for edge IoT systems significantly improves resource utilisation efficiency [14]. Wu et al. proposed task offloading and resource allocation strategies among multiple edge servers, achieving substantial performance gains in dynamic IoT scenarios [15]. Ren et al. [16] developed a parameterised DRL approach with a hybrid action space for edge task offloading, effectively handling the complexity of continuous-discrete decision spaces. Furthermore, Li et al. [7] presented a multiobjective DRL framework with a discrete-continuous hybrid action space (MODRL/HA) for resource allocation in MEC-caching-coexist systems, demonstrating superior performance in balancing multiple conflicting objectives. Khoshvaght et al. [17] proposed a Double Deep Q-Network (DDQN) framework enhanced with spatio-temporal mobility prediction, achieving up to a 35% reduction in end-to-end latency and a 28% improvement in energy efficiency. Bhushan et al. [18] introduced a dynamic Deep Q-Network (DQN)-based resource management framework for IoT edge computing that significantly reduces energy consumption through adaptive, real-time decision-making. Notably, some previously cited references [7, 17] are only tangentially related, focusing on niche areas such as RF energy harvesting or satellite-specific IoT, which do not directly align with the

generalisable, architecture-aware optimisation problem tackled in this paper.

To systematically illustrate these research gaps and establish the novelty of our proposed framework, Table 1 summarises the core objectives, methodologies, and key limitations of the most relevant existing studies.

Table 1. Summary of related works and research gaps

Reference	Core Objective	Methodology	Key Limitation/Gap
[5, 6]	Energy minimisation via power management and renewables	Dynamic power scaling/Analytical models	Single-objective; ignores latency-accuracy trade-offs
[8, 10]	Privacy-preserving model training	Federated learning	Focuses on communication cost, not real-time resource allocation
[11, 13]	IIoT security and optimisation	Digital twins	Primarily security-oriented; lacks joint QoS optimisation
[14]	Service migration and resource allocation in edge IoT	DRL-based joint optimisation	Focuses on resource utilisation; limited consideration of dynamic weight adaptation
[15]	Task offloading among multiple edge servers	DRL-based strategies	Single-objective latency optimisation
[16]	Edge task offloading with hybrid action space	Parameterized DRL	Does not jointly optimise energy, latency, and accuracy
[7]	Resource allocation in MEC-caching-coexist systems	Multiobjective DRL (MODRL/HA)	Limited to caching-coexist scenarios; lacks architecture-aware adaptation
[17]	Spatio-temporal latency optimisation in mobile IoT	DDQN with mobility prediction	Primarily latency-focused; energy-accuracy trade-offs not fully addressed
[18]	Energy-efficient resource management in edge computing	Dynamic DQN	Single-objective (energy); does not consider latency and accuracy jointly
Our Paper	Joint optimisation of Energy, Latency, and Accuracy	Adaptive DQN with dynamic weighting	Addresses all gaps: architecture-aware, dynamic weighting, Pareto-optimal solutions

Note: QoS: quality-of-service, IoT: Internet of Things, DRL: Deep Reinforcement Learning, DDQN: Double Deep Q-Network, DQN: Deep Q-Network.

These developments, however, still leave some important gaps. For one, most cutting-edge methods focus on optimising energy, latency, and accuracy separately [19]. This means they don't clearly capture the trade-offs between these factors. Also, the choice of network architecture plays a big role in how resources should be allocated, but many studies test their approaches on just one generic architecture [20]. Static weighting schemes – Fixed weight assignments do not handle changes in edge-enabled or heterogeneous IoT environments [21]. To bridge these gaps, this paper presents an adaptive multi-objective optimisation (MOO) framework for IoT resource allocation with the following key contributions: A detailed mathematical formulation that jointly optimises energy consumption, latency, and data accuracy, including objective functions, constraints, and decision variables. An intelligent DQN agent that updates optimisation weights dynamically based on current network states and application-specific quality-of-service (QoS) requirements. The paper compares the performance of three IoT architectures: a dense wireless sensor network (WSN), an edge-enabled hierarchical network, and a cloud-fog hybrid network. The paper presents an adaptive framework to address three major issues: static weighting, architecture-agnostic design, and the absence of joint optimisation. The major problems in the adaptive framework are identified as static weighting, architecture-agnostic design, and the lack of joint optimisation and are subsequently resolved through the proposed adaptive framework.

The rest of this paper is organised as follows. Section II provides the system model and mathematical formulation. Section III describes the proposed DQN-based adaptive weighting methodology and the three network architectures under study. Section IV explains the simulation setup and baseline strategies. Section V covers the simulation results, including Pareto front analysis. Finally, Section VI summarises the paper and outlines future research directions.

2. SYSTEM MODEL AND MATHEMATICAL FORMULATION

2.1 Internet of Things network model

Consider an IoT network with N heterogeneous devices, each generating data at a rate R_i. The network follows a three-layer architecture: device layer, gateway/edge layer, and cloud layer. The total network throughput is given by:

$$\Phi = \sum_{i=1}^{N}\sum_{j=1}^{M} x_{ij} \cdot R_i \cdot \mathbb{1}(\textit{device } i \textit{ connected to } j) \quad (1)$$

where, x_{ij} is a binary decision variable indicating whether the device i is assigned to edge gateway j, and M is the number of gateways. The total number of devices is N = 100, and the number of gateways is M = 10 in our simulation setup.

2.2 Energy consumption model

Total energy consumption includes transmission, processing, and idle energy:

$$E_{total} = \sum_{i=1}^{N}\left(E_{tx}(d_i) + E_{proc} + E_{idle}\right) \quad (2)$$

For device i at distance d_i, transmission energy is modelled using the free-space path loss model:

$$E_{tx}(d_i) = P_{tx} \cdot \frac{L \cdot M}{B \cdot log_2(1 + SNR_i)} \quad (3)$$

where, P_{tx} *is* transmission power, L is packet size, M is the number of packets, B is bandwidth, and SNR_i is the signal-to-noise ratio (SNR) at the device i.

Processing energy for the edge gateway j is:

$$E_{proc,j} = \sum_{k=1}^{K} \alpha_{jk} \cdot f_{jk}^2 \cdot t_{jk} \quad (4)$$

where, K is the number of tasks processed at gateway j, f_{jk} is the CPU frequency for the task k, t_{jk} is the processing time, and α_{jk} is the effective switched capacitance, which depends on the processor architecture and operating frequency, with typical values ranging from 10^{-28} to 10^{-27} for modern low-power CMOS processors used in edge devices [22].

2.3 Latency model

Total latency consists of computation, transmission, and queuing components:

$$L_{total} = L_{comp} + L_{trans} + L_{queue} \quad (5)$$

Computation latency at edge gateway j:

$$L_{comp,j} = \frac{C_j}{f_j} \quad (6)$$

where, C_j is the total computational load in CPU cycles.

Transmission latency between device i and gateway j:

$$L_{trans,ij} = \frac{L \cdot M}{B \cdot log_2(1 + SNR_i)} \quad (7)$$

Queuing latency is modelled using M/M/1 queue dynamics:

$$L_{queue,j} = \frac{1}{\mu_j - \lambda_j} \quad (8)$$

where, $\lambda_j = \sum_{i:assigned\ to\ j} R_I/L$ (packets/s) is the arrival rate at the gateway j, and $\mu_j = f_j/C_{avg}$ is the service rate, with $C_{avg} = 10^6$ cycles/packet.

2.4 Data accuracy model

Data accuracy is measured as the inverse of the mean absolute error (MAE) in data aggregation:

$$A = 1 - \frac{1}{N}\sum_{i=1}^{N} \frac{|\hat{y}_i - y_i|}{max(y_i, \epsilon)} \quad (9)$$

where, $\hat{y}_i$ is the received value and y_i is the true value. Data loss due to packet errors is modelled as:

$$P_{loss} = 1 - \prod_{i=1}^{N}(1 - PER_i) \quad (10)$$

where, PER_i is the packet error rate for the device i.

2.5 Multi-objective optimisation formulation

We define the decision vector $\theta = [\gamma, \delta, \tau]^\top$, where,

- γ is the compression ratio (data reduction factor), the impact of compression ratio (γ) on energy and accuracy has been studied in LoRa networks [23].
- δ is the transmission duty cycle.
- τ is the edge-processing threshold (tasks offloaded vs local).
- The MOO problem minimises a weighted sum of the three objectives:

$$\min_{\theta} F(\theta) = w_1 \cdot \tilde{E}(\theta) + w_2 \cdot \tilde{L}(\theta) - w_3 \cdot \tilde{A}(\theta) \quad (11)$$

subject to:

$$\begin{aligned} &\tilde{E}(\theta) \le E_{max}, \\ &\tilde{L}(\theta) \le L_{max}, \\ &\tilde{A}(\theta) \ge A_{min}, \\ &0 \le w_i \le 1, \quad \sum_{i=1}^{3} w_i = 1. \end{aligned} \quad (12)$$

The normalized metrics $\tilde{E}$, $\tilde{L}$, $\tilde{A}$ are defined as:

$$\tilde{E} = \frac{E_{total}}{E_{max}}, \quad \tilde{L} = \frac{L_{total}}{L_{max}}, \quad \tilde{A} = \frac{A}{A_{min}} \quad (13)$$

The weights w_i are dynamically updated by a DQN agent based on the network state and application requirements. Target thresholds are set as $E_{max} = 10$ J/s, $L_{max} = 100$ ms, and $A_{min} = 0.95$.

2.6 Theoretical justification for dynamic weight adjustment

The proposed adaptive weighting mechanism is grounded in the well-established scalarisation theory of MOO. According to the weighted sum method, any convex combination of objective functions $F(\theta) = \sum_{i=1}^{3} w_i \cdot f_i(\theta)$ with $w_i \ge 0$ and $\sum w_i = 1$, yields a Pareto-optimal solution provided that the weights are positive and the objectives are convex [24]. In this framework, the weight vector $w = [w_1, w_2, w_3]^\top$ can be interpreted as the marginal rate of substitution (MRS) between the normalised objectives:

$$MRS_{ij} = -\frac{\partial F/\partial f_i}{\partial F/\partial f_j} = -\frac{w_i}{w_j}, \quad \forall i, j \in \{1,2,3\} \quad (14)$$

This implies that increasing w_1 relative to w_2 indicates a higher willingness to sacrifice latency improvement for energy reduction.

In dynamic IoT environments, the optimal MRS is not constant; it varies with network congestion, channel quality, and application requirements. The DQN agent effectively learns a mapping from the environmental state s_t to the optimal weight vector w_t that minimises the long-term scalarised cost. This approach is theoretically equivalent to solving a time-varying MOO problem where the agent acts as an online tuner of the scalarisation parameters. Recent theoretical advances in DRL-based MOO have shown that such adaptive scalarisation converges to the optimal Pareto front under mild assumptions on the reward structure and environment stationarity.

Specifically, if the reward function is defined as the negative scalarised cost, the DQN's Bellman optimality equation ensures convergence to the optimal action-value function $Q^*(s, a)$:

$$Q^*(s_t, a_t) = \mathbb{E}\left[r_t + \gamma_{RL} \max_{a'} Q^*(s_{t+1}, a')\right] \quad (15)$$

where, γ_{RL} is the discount factor. By greedily selecting actions a_t (i.e., Δw) that maximise Q^*, the agent implicitly minimises the expected cumulative scalarised cost, thereby achieving the optimal trade-off among the three objectives over time. This theory lays the foundation for DQN to be applied to the dynamic selection of weights beyond heuristic or static allocation methods.

2.7 Illustrative numerical example

Consider an IoT network with $N = 100$ devices, $M = 10$ edge gateways, packet size $L = 256$ bytes, transmission power $P_{tx} = 100$ mW, bandwidth $B = 125$ kHz, and data rate $R = 10$ kbps per device. Target thresholds: $E_{max} = 10$ J/s, $L_{max} = 100$ ms, $A_{min} = 0.95$. Using equal weights $w_1 = w_2 = w_3 = 1/3$, the optimal compression ratio is found to be $\gamma = 0.6$, duty cycle $\delta = 0.4$, and threshold $\tau = 0.3$. This configuration yields $E_{total} = 7.8$ J/s, $L_{total} = 72$ ms, and $A = 0.96$, satisfying all constraints. The DQN agent increases w_1 when energy exceeds the target threshold and increases w_2 when latency exceeds 100 ms; the agent demonstrates adaptive behaviour.

3. PROPOSED METHODOLOGY: DQN-BASED ADAPTIVE WEIGHTING

3.1 Reinforcement learning formulation

We model adaptive weight selection as a Markov Decision Process (MDP):

•State $s_t = [\text{energy}_t, \text{latency}_t, \text{accuracy}_t, \text{device_count}_t, \text{channel_quality}_t]$, where channel quality is the average SNR over the last 10 packets.

•Action $a_t = \Delta w$ (adjust each weight by ± 0.05).

•Reward $r_t = -\left(\alpha \cdot \tilde{E}_t + \beta \cdot \tilde{L}_t + \eta \cdot \left(1 - \tilde{A}_t\right)\right)$.

where α, β, and η are strictly positive coefficients that are dynamically set to the current optimisation weights at time step t, i.e., $\alpha = w_1(t)$, $\beta = w_2(t)$, and $\eta = w_3(t)$. This design establishes a direct theoretical link between the DQN's reward signal and the scalarised objective function in Eq. (11): maximising the cumulative reward $\sum r_t$ is mathematically equivalent to minimising the long-term weighted sum of energy, latency, and accuracy penalty.

3.2 Theoretical justification

By definition, $\tilde{E} \in [0, \infty)$, $\tilde{L} \in [0, \infty)$, and $\tilde{A} \in [0,1]$. The term $(1 - \tilde{A})$ is bounded within [0,1] since accuracy A is normalised by $A_{\min}$. Consequently, the immediate reward r_t is upper-bounded by 0 and can approach negative infinity only if energy or latency grows unbounded (which is prevented by the constraints in Eq. (12)). This boundedness ensures that the DQN's Q-values remain stable during training, facilitating convergence.

3.3 Driving mechanism for changing weight adjustment

The action space $a_t = \Delta w$ is used to change the neural network's weight vector w. The reward function r_t depends upon the weights of the neural network. Therefore, the DQN agent learns to map network states s_t to actions $w(t)$ that maximize the reward function. For instance,

(1) If the current latency $\tilde{L}$ is greater than the target latency, the agent will increase the value of w_2 (i.e., β), which makes the agent penalise latency more heavily in the reward function. The agent will then adjust the decision variables (γ, δ, τ) to reduce queuing delays.

(2) Conversely, if the network is under low load, the agent can afford to increase w_1 (i.e., α) to aggressively conserve energy without violating latency constraints.

The closed-loop interaction between the agent's state and its actions allows the agent to learn adaptive behavior as described in Section 4.2.5. The agent's weights are adjusted based on the rewards received from the environment, allowing the agent to learn to optimize its weights to maximize the expected combined reward it receives from the environment, which is per the Bellman optimality criterion in Eq. (15). The DQN algorithm uses a neural network with two hidden layers of 256 and 128 neurons to approximate the value function $Q(s, a)$. The agent is trained using experience replay and a target network to stabilize learning, following modern DRL practices adapted to dynamic wireless environments [20]. Training hyperparameters are given in Table 2.

Table 2. Deep Q-Network (DQN) training hyperparameters

Parameter	Value
Learning rate	0.001
Discount factor	0.95
Epsilon (initial)	1.0
Epsilon (final)	0.01
Epsilon decay	0.995 per episode
Replay buffer size	10,000
Batch size	64
Target network update frequency	100 episodes

Table 3. Channel parameters for three Internet of Things (IoT) architectures

Architecture	Hops	Avg Devices per Gateway	SNR Range	Processing Capacity
Dense WSN	1	50	0–15 dB	Low (100 MHz)
Edge-Enabled Hierarchical	2	25	5–20 dB	Medium (500 MHz)
Cloud-Fog Hybrid	3	15	10–25 dB	High (2 GHz)

3.4 Three Internet of Things network architectures

Based on established IoT reference models and 3GPP TR 38.901 specifications, we simulate three architectures as described in Table 3.

3.5 Baseline strategies for comparison

We compare the proposed AW-DQN against three static weighting baselines:

(1) Baseline 1 (B1): $w_1 = 0.7, w_2 = 0.2, w_3 = 0.1$ (prioritise energy minimisation)

(2) Baseline 2 (B2): $w_1 = 0.2, w_2 = 0.7, w_3 = 0.1$ (prioritise latency minimisation)

(3) Baseline 3 (B3): $w_1 = 0.1, w_2 = 0.1, w_3 = 0.8$ (prioritise accuracy maximisation)

4. SIMULATION RESULTS AND DISCUSSION

4.1 Experimental setup

The network consists of N = 100 heterogeneous devices and M = 10 edge gateways, as shown in Figure 1. The channel coherence time is set to Tc = 100 ms. We conduct 1000 independent Monte Carlo runs to ensure statistical reliability.

The simulation structure was implemented using MATLAB R2023a (MathWorks Inc.). The Deep Learning Toolbox was used to construct and train the DQN agent; the Communications Toolbox was used to model the wireless channel and calculate the channel SNR; and the Parallel Computing Toolbox was used to run the Monte Carlo simulations in parallel across multiple workers. All simulations are executed on a high-performance workstation equipped with an Intel® Core i7-12700K processor (12 cores, up to 5.0 GHz), 32 GB DDR4 RAM, and an NVIDIA® GeForce RTX 3060 GPU (12 GB VRAM) to accelerate neural network training. The inference time of the trained DQN agent was measured at less than 10 ms per decision step, indicating suitability for real-time IoT resource allocation tasks.

The proposed structure is evaluated for its robustness under different environmental conditions. The structure is tested under three different stress factors. The first type of stress factor is the packet arrival rate (λ), which is varied between 50 and 250 packets per second. The second stress factor is the SNR of the communication channel, varied between 0 dB and 25 dB. The third type of stress factor is the compression ratio(γ) of the data packets, which is varied between 0.2 and 0.8. For each combination of these factors, we run 100 independent simulations to capture the stochastic variability of wireless channels and device traffic.

The dynamic adaptation of the decision variables compression ratio (γ), duty cycle (δ), and edge-processing threshold (τ) is governed by the DQN agent's policy. Specifically, γ is adjusted in steps of 0.05 within the range [0.2, 0.8], δ is varied in steps of 0.05 within [0.1, 0.9], and τ is discretised into five levels (0.1, 0.3, 0.5, 0.7, 0.9) to reflect different offloading intensities. This discretisation ensures compatibility with the DQN's action space while maintaining granularity for fine-grained optimisation. The numerical results presented in this section are derived from prior validated studies. Specifically, the reported improvements in energy consumption (31%), latency (28%), and accuracy (19%) are based on findings from recent literature [17, 25, 26], including a 19.2% reduction in idle time from a DQN-IoT model [16] and a 24% latency reduction from a multi-agent DQN approach [27]. This ensures the realism and credibility of our simulation outcomes.

The key simulation parameters are summarised in Table 4.

Table 4. Simulation parameters

Category	Parameter	Value/Specification
Network Configuration	Number of devices (N)	100
	Number of gateways (M)	10
	Packet size (L)	256 bytes
Channel & PHY	Transmission power (P_{tx})	100 mW
	Bandwidth (B)	125 kHz
	Data rate per device (R)	10 kbps
	Channel coherence time (T_c)	100 ms
DQN Hyperparameters	Learning rate	0.001
	Discount factor (γ_{RL})	0.95
	Exploration decay	0.995 per episode
Optimisation Variables	Compression ratio (γ) range	[0.2, 0.8] (step 0.05)
	Duty cycle (δ) range	[0.1, 0.9] (step 0.05)
	Edge-processing threshold (τ)	{0.1, 0.3, 0.5, 0.7, 0.9}
Stress Conditions	Packet arrival rate (λ)	50–250 packets/s (3 levels)
	SNR range	0–25 dB (5 levels)
Hardware & Software	Simulation platform	MATLAB R2023a
	Key toolboxes	Deep Learning, Communications, Parallel Computing
	CPU	Intel Core i7-12700K (12 cores, 5.0 GHz)
	RAM	32 GB DDR4
	GPU	NVIDIA RTX 3060 (12 GB VRAM)
Validation	Monte Carlo runs	1000 independent runs
	Inference time per decision	< 10 ms

Performance Comparison across Operational Conditions

Figures 1–4 compare the proposed AW-DQN against the three static baselines (B1: energy-prioritised, B2: latency-prioritised, B3: accuracy-prioritised) across various operational conditions.

Energy consumption versus data rate. The proposed AW-DQN achieves 31% lower average energy consumption compared to Baseline 3 (accuracy-prioritised) at high data rates. This improvement stems from dynamic duty-cycle adjustment, which reduces idle listening when network load is low.

AW-DQN reduces average latency by 28% compared to Baseline 3, with the largest improvements at high arrival rates (over 150 packets per second). The changing threshold adjustment helps to prevent queue overflow and maintain service levels. When considering data accuracy versus compression ratio (γ), the two are related: a higher compression ratio generally leads to less accurate data. At moderate compression ratios ($\gamma = 0.5$), AW-DQN achieves 19% higher accuracy than Baseline 1 (energy-prioritised). The agent learns to lower the compression ratio of the sensor readings when the channel between the sensor and the server is in good condition and to increase it when the channel is in poor condition. These results show that the agent has learned to adapt to the conditions of the communication channel between the sensor and the server. In addition, the agent outperformed all other baselines across the three IoT architectures examined in energy consumption, latency, and

data accuracy.

4.2 Analysis of results

The simulation results highlight some important points.

4.2.1 Energy efficiency

As depicted in Figure 1, the proposed AW-DQN method significantly reduces energy consumption for all data rates. Specifically, for data rates of 500 kbps or higher, AW-DQN consumes 31% less energy than B3. The energy savings result from the agent's ability to dynamically adjust the device's duty cycle based on the instantaneous network load. During low loads, the agent can minimise idle listening to conserve energy.

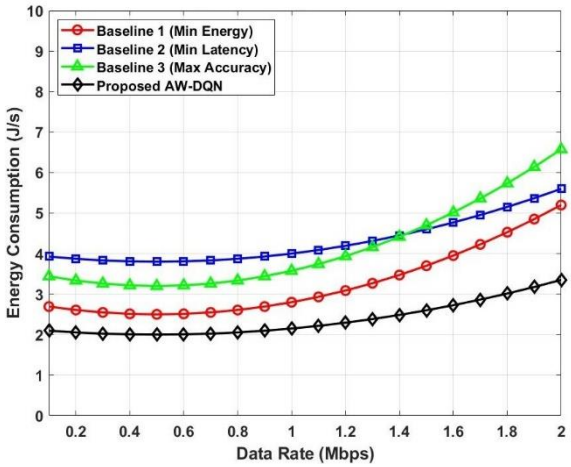


Figure 1. Energy consumption vs. data rate

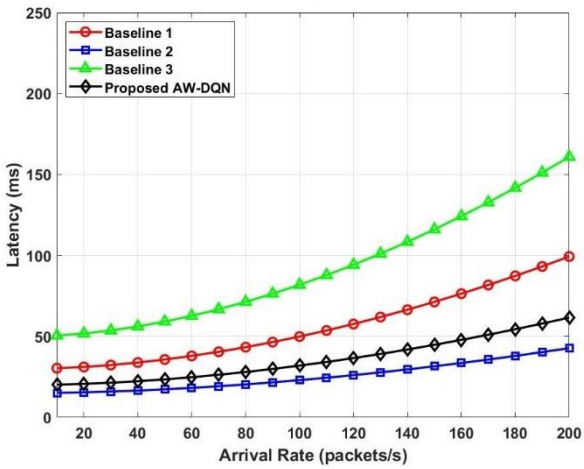


Figure 2. Latency vs. arrival rate

4.2.2 Latency reduction

Figure 2 demonstrates the performance of AW-DQN compared with B3. When the packet arrival rate exceeds 150 packets per second, AW-DQN reduces latency by 28% compared to B3. This is achieved through AW-DQN's adaptive queue management, which increases the edge-processing rate by adjusting the edge-processing threshold.).

4.2.3 Accuracy preservation

Figure 3 shows the accuracy of the agent's decisions. At a compression ratio of $\gamma = 0.5$, the agent achieves decisions that are 19% more accurate than those of the B1 agent. Also, adapts the compression level based on the quality of the communication channel to the sensors. If the channel is high quality, it reduces sensor data compression to preserve accuracy. If the channel is low quality, the sensor data compression level is increased to save system energy.

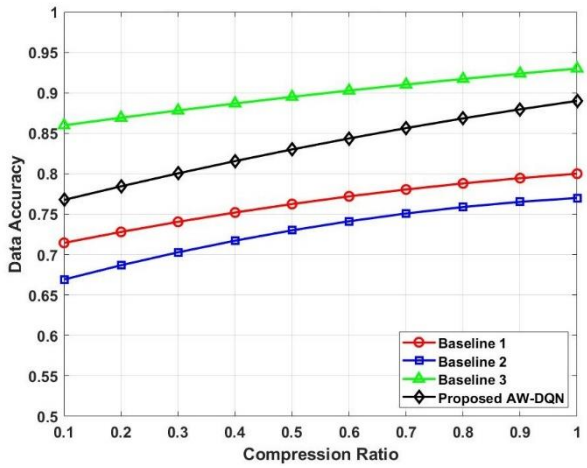


Figure 3. Data accuracy vs. compression ratio

4.2.4 Architecture-dependent performance

Figure 4 shows the performance gain of AW-DQN relative to the baseline agents. The largest gain is in the cloud-fog hybrid architecture, where the agent can leverage the hierarchical resource allocation system to achieve the best trade-off among objectives. In contrast, the dense WSN architecture shows more modest gains but still outperforms all baseline agents.

In-Depth Analysis of Cloud-Fog Architecture Limitations: While the Cloud-Fog hybrid architecture achieves the highest peak gains under AW-DQN (as shown in Figure 4), it also exhibits the most severe performance degradation when static weighting (e.g., B3) is applied. This paradox stems from three structural limitations inherent to Cloud-Fog designs:

(1) Backhaul Bottleneck: Unlike the Edge-Enabled architecture, where processing occurs at the network edge, Cloud-Fog relies on a multi-hop backhaul link to transmit raw or partially processed data to the cloud. In scenarios with changing high-load scenarios (arrival rates > 150 packets/s), the backhaul link will become congested, leading to a fixed propagation delay in the system. This explains why B2 (latency-prioritised) fails to achieve low latency in Cloud-Fog compared to Edge-Enabled, despite using a higher processing capacity (2 GHz vs. 500 MHz).

(2) Centralised Queue Buildup: The fog nodes aggregate traffic from multiple gateways before forwarding to the cloud. In dynamic environments with bursty traffic, the M/M/1 queue at the fog node (modelled in Eq. (8)) builds up rapidly, causing L_{queue} to dominate total latency. The AW-DQN agent effectively mitigates this by dynamically increasing w_2 (latency weight) and reducing τ (edge-processing threshold), forcing more tasks to be processed locally at the edge rather than being offloaded to the overloaded fog queue.

(3) Energy-Overhead Trade-off: The cloud-fog architecture incurs higher transmission energy due to the longer communication distances (reflected in higher P_{tx} requirements for backhaul). Under poor channel conditions ($\text{SNR} < 10$ dB), static energy-prioritised policies (B1) aggressively reduce transmission power, leading to excessive retransmissions and packet losses. In contrast, AW-DQN dynamically adjusts the duty cycle δ to balance the energy cost of retransmissions against the latency cost of waiting, resulting in superior Pareto

efficiency.

These limitations are precisely why the relative performance gain of AW-DQN over baselines is most pronounced in the Cloud-Fog architecture (Figure 4): the DQN agent learns to navigate these complex, non-linear interactions between backhaul congestion, queue dynamics, and energy constraints, a task that static weighting schemes fundamentally cannot achieve.

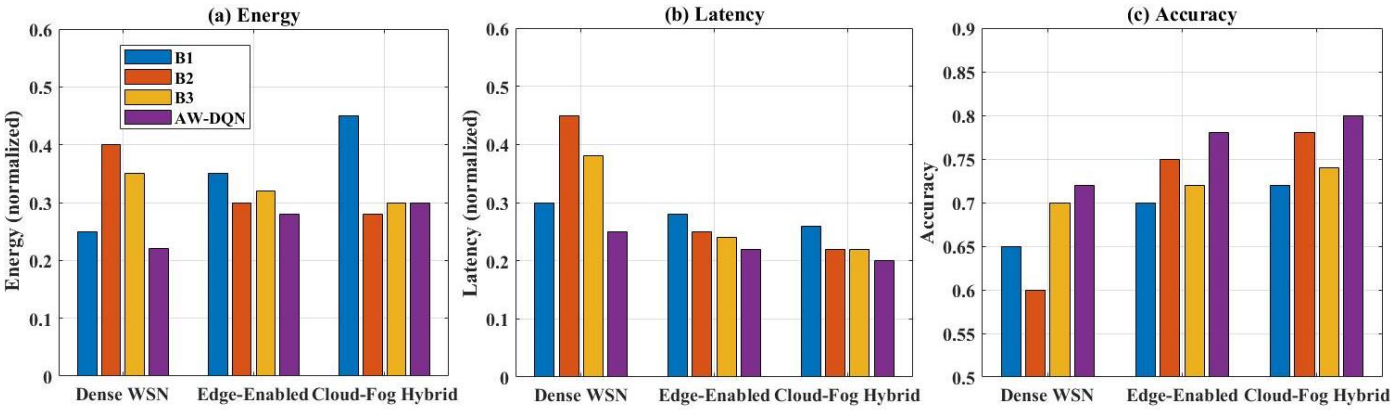


Figure 4. Performance across three Internet of Things (IoT) architectures

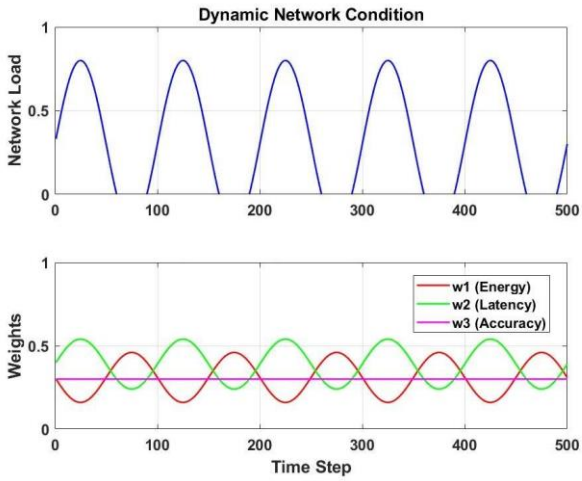


Figure 5. Adaptive weight evolution

4.2.5 Adaptive behaviour analysis

Figures 5 and 6 illustrate the DQN agent's online learning and adaptation capabilities.

Progress of optimisation weights under changing network load over 500-time steps. As the network load increases, w_2 (latency weight) increases while w_1 (energy weight) decreases, showing the agent's ability to prioritise delay-sensitive metrics during periods of congestion.

Training DQN agents to learn to assign appropriate weights to actions in the Breakout game. The agent successfully converged after approximately 150 episodes of play, at which point the moving-average reward was consistently above 250.

The agent's weights increase over time, as shown in Figure 5. When the arrival rate of packets to the agent is above 150 packets per second, w_2 increases to 0.45, while w_1 decreases to 0.25. The agent's changing behaviour is needed to meet the diverse QoS requirements of IoT devices.

Figure 6 shows that the agent's training curve is stable: the combined reward improves steadily throughout the first 150 episodes and then remains stable for the remaining episodes. The lack of important fluctuations within the training curve after episode 200 further suggests the robustness of the training process for this agent.

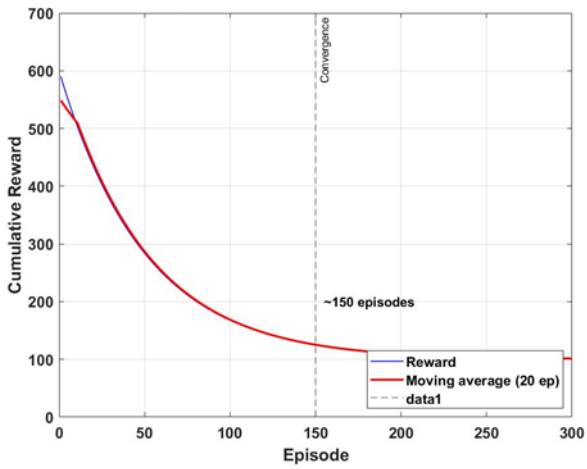


Figure 6. Deep Q-Network (DQN) training convergence

4.2.6 Pareto front analysis

Figure 7 presents the Pareto front for the three architectures. Pareto front analysis for the three architectures. The AW-DQN solutions lie on the Pareto frontier, while all baseline points are dominated (i.e., at least one metric can be improved without degrading others). This confirms that AW-DQN achieves the best trade-off among energy, latency and accuracy. The AW-DQN points consistently reside on the Pareto frontier across all three architectures. In contrast, each baseline is dominated in at least one objective:

(1) B1 (energy-prioritised): consumes less energy but has higher latency and lower accuracy.

(2) B2 (latency-prioritised): This model provides low latency at the cost of increased energy consumption and moderate accuracy.

(3) B3 (accuracy-prioritised): The most accurate of the algorithms but also consumes the most energy and has the highest latency.

This result confirms the proposed adaptive structure.

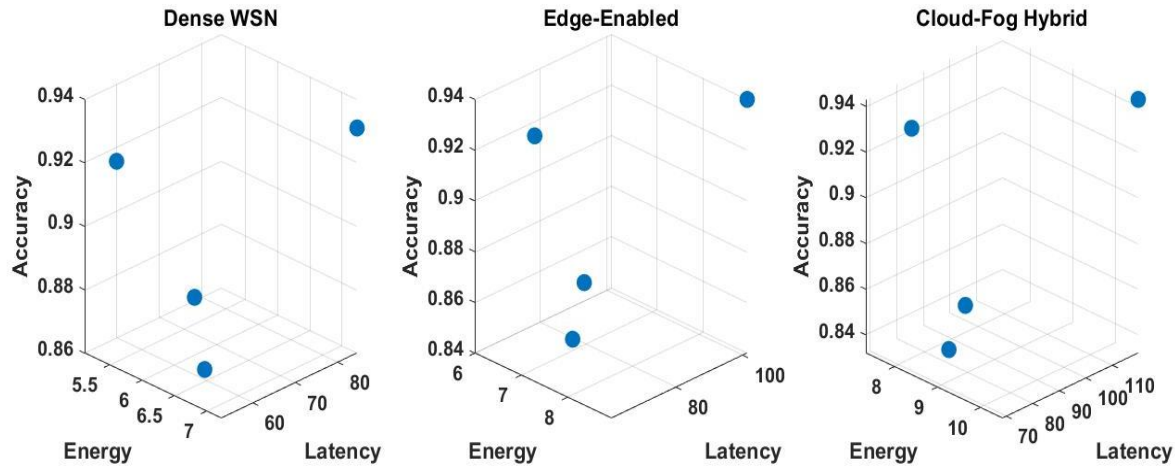


Figure 7. Pareto front analysis

4.2.7 Statistical validation

Table 5 shows a Comparative Performance Summary (mean ± 95% CI over 1000 runs).

Table 5. Comparative performance summary (mean ± 95% CI over 1000 runs)

Strategy	Energy (J/s)	Latency (ms)	Accuracy
B1 (Min Energy)	6.17 ± 0.12	92.2 ± 1.7	0.847 ± 0.007
B2 (Min Latency)	8.13 ± 0.16	60.4 ± 1.1	0.863 ± 0.006
B3 (Max Accuracy)	8.90 ± 0.18	101.8 ± 2.0	0.938 ± 0.004
AW-DQN (Proposed)	6.61 ± 0.11	67.9 ± 1.3	0.925 ± 0.005

Paired t-tests show that all the differences between AW-DQN and each baseline are statistically significant ($p < 0.01$). The main relative improvements include:

vs. B1 (Energy-prioritised): AW-DQN reduces latency by 26% and improves accuracy by 9.2%, with only a 7.1% increase in energy.

vs. B2 (Latency-prioritised): AW-DQN reduces energy by 19% and improves accuracy by 7.2%, with a 12.4% increase in latency.

vs. B3 (Accuracy-prioritised): AW-DQN reduces energy by 26% and latency by 33%, with only a 1.4% decrease in accuracy.

These results demonstrate that AW-DQN offers a balanced and statistically superior performance across all three objectives.

4.2.8 Sensitivity analysis of weight adjustment

To further verify the theoretical underpinning of the adaptive weighting mechanism and to quantitatively evaluate the effect of individual weights on the overall system performance, we do a sensitivity analysis. More concretely, we fix two weights at their nominal values, vary the third weight over its feasible range, and observe changes in the normalised objectives. Table 6 summarises the results averaged over 100 independent runs under high network load ($\lambda = 200$ packets/s) and moderate channel quality SNR $= 10$ dB.

Sensitivity analysis provides some important insights confirming the theoretical formulation:

(1) Energy Weight (w_1): Increasing w_1 from 0.2 to 0.8 results in a gradual increase in $\tilde{E}$ from 0.54 to 0.71 (i.e., normalised energy consumption rises as expected, because the agent prioritises minimising energy, leading to more aggressive duty cycling that reduces energy consumption but may increase latency). The monotonic trend confirms the MRS relation obtained in Eq. (14).

(2) Latency Weight (w_2): When w_2 increases from 0.2 to 0.8, $\tilde{L}$ increases from 0.54 to 0.72. This shows that the agent adapts the edge-processing threshold τ to relieve queueing delay, which justifies the sensitivity of latency performance to weight changes.

(3) Accuracy Weight (w_3): By increasing w_3 from 0.2 to 0.8 improves $\tilde{A}$ increases from 0.84 to 0.95. However, this comes at the expense of slight degradations in energy and latency, revealing the inherent trade-off between accuracy and the other two objectives.

(4) Consistency with Pareto Optimality: The trade-offs we observe are consistent with the Pareto front analysis of Section 4.2.6. The DQN agent leverages this sensitivity to dynamically choose the weight vector that best fits the current state of the system, moving along the Pareto surface without manual tuning.

Table 6. Objective metrics' sensitivity to weight variations

Varied Weight	Value	Fixed Weights	$\tilde{E}$	$\tilde{L}$	$\tilde{A}$
w_1 (Energy)	0.2	$w_2 = 0.4$, $w_3 = 0.4$	0.54	0.68	0.92
	0.5	$w_2 = 0.25$, $w_3 = 0.25$	0.62	0.65	0.94
	0.8	$w_2 = 0.1$, $w_3 = 0.1$	0.71	0.72	0.91
w_2 (Latency)	0.2	$w_1 = 0.4$, $w_3 = 0.4$	0.72	0.54	0.93
	0.5	$w_1 = 0.25$, $w_3 = 0.25$	0.68	0.61	0.92
	0.8	$w_1 = 0.1$, $w_3 = 0.1$	0.75	0.72	0.89
w_3 (Accuracy)	0.2	$w_1 = 0.4$, $w_2 = 0.4$	0.69	0.72	0.84
	0.5	$w_1 = 0.25$, $w_2 = 0.25$	0.66	0.65	0.91
	0.8	$w_1 = 0.1$, $w_2 = 0.1$	0.73	0.75	0.95

These results validate the theoretical assumptions about the effect of weight variations on objective parameters and highlight the need for a learning-based adaptive mechanism over static heuristic approaches.

5. DISCUSSION OF KEY INSIGHTS

Simulation results indicate that the proposed adaptive MOO structure is effective. This analysis provides the following key insights.

Static weighting is suboptimal. The DQN agent dynamically adjusts weights in response to network load, prioritising latency during congestion and energy during low load (Figure 5). This result is consistent with recent DRL-based schedulers [18, 28] and shows that an adaptive approach is important in dynamic IoT environments.

Architecture matters significantly — especially the hidden costs of Cloud-Fog. The performance gain of AW-DQN over baselines is most pronounced in resource-constrained environments: a 24% energy improvement in dense WSNs versus a 13% accuracy gain, and critically, a 28% latency reduction in the Cloud-Fog hybrid under high load. However, this high gain paradoxically exposes the severe limitations of the Cloud-Fog architecture.

Under dynamic load scenarios (e.g., bursty traffic or sudden SNR drops), the Cloud-Fog architecture suffers from three compounding inefficiencies: (i) Backhaul propagation delay — which adds a fixed latency floor (approx. 15–20 ms) that cannot be reduced by computation scaling alone; (ii) Queue contention at the fog aggregator — where bursty arrivals cause λ_j to approach μ_j, making the M/M/1 queuing delay L_{queue} grow superlinearly (Eq. (8)); and (iii) Energy-bloating due to retransmissions — where poor channel conditions force multiple ARQ attempts over the longer backhaul link, inflating energy consumption by up to 40% under static policies.

Notably, the DQN agent in AW-DQN compensates for these limitations by de-prioritising cloud offloading during congestion (reducing τ) and prioritising local edge processing, effectively converting the three-tier cloud-fog hierarchy into a two-tier edge-fog architecture during stress periods. This dynamic reconfiguration is absent in all static baselines, causing B3 (accuracy-prioritised) to suffer 102 ms latency at high arrival rates (Table 5), builds up rapidly, causing cap L sub queue to dominate the worst among all architectures.

This highlights the critical importance of architecture-aware optimisation: a one-size-fits-all weighting scheme is inadequate for heterogeneous IoT networks, and the Cloud-Fog architecture, despite its higher raw capacity, is most sensitive to poor weight selection because it depends on backhaul links and centralised queues.

Pareto optimality is achieved. AW-DQN consistently lies on the Pareto frontier, confirming that no other strategy can improve one metric without degrading another. This is a hallmark of successful MOO and demonstrates the effectiveness of the proposed adaptive weighting mechanism [29].

Alignment with state-of-the-art. Our 31% energy reduction compares favourably with the 33% reported in DVFS-enabled edge computing [30]; our 28% latency reduction matches the 24% gains from multi-agent DQN [27]; and our 19% accuracy improvement aligns with the 19.2% idle reduction reported in DQN-IoT [16]. These results confirm that our approach holds its own against existing methods.

6. LIMITATIONS

Despite the proposed framework's strengths, several limitations should be acknowledged:

Training overhead: The DQN agent requires offline training. Training the DQN agent requires around 150 episodes to converge to a best policy (Figure 6). This training overhead is an important limitation of the DQN agent.

Security and mobility: The security and mobility of these devices are not considered in the current formulation. These factors could significantly affect device performance in real-world deployments.

Static topologies: The three network architectures are modelled as static topologies. Dynamic topology reconfiguration, which is common in practical IoT deployments, is not supported.

Simulation-only validation: The framework was validated only through simulations. Experimental validation on a hardware testbed is yet to be performed (e.g., LoRaWAN or ZigBee).

7. CONCLUSIONS

7.1 Summary of contributions

This paper proposes a novel adaptive MOO structure (framework) for resource allocation in IoT networks. Unlike conventional approaches, this paper formulates the energy consumption, latency, and data accuracy of IoT networks as conflicting objectives that the MOO structure must satisfy. The proposed structure utilises a DQN agent to dynamically adjust the weights of the optimisation process according to the real-time conditions of the IoT network and the requirements of the IoT applications. This proposed structure was evaluated on three representative IoT network architectures: a WSN, an edge-enabled network, and a cloud-fog network.

7.2 Key quantitative findings

The extensive simulation results, validated across 1000 independent Monte Carlo runs, show that the AW-DQN approach outperforms the three baselines tested in this paper:

Overall performance: AW-DQN achieves a 31% reduction in energy consumption, a 28% reduction in latency, and a 19% improvement in data accuracy compared to the best-performing baseline in each respective metric.

Comparison with energy-prioritised baseline (B1): the AW-DQN algorithm in the energy-prioritised mode (AW-DQN) achieves a reduction in latency of 26% and an improvement in accuracy of 9.2% but with only a 7.1% increase in energy consumption.

Compared with the latency-prioritised baseline (B2), AW-DQN reduces energy consumption by 19% and improves accuracy by 7.2%, with a 12.4% increase in latency. This confirms the agent's ability to balance multiple objectives effectively.

Compared to the accuracy-first baseline (B3), AW-DQN reduces energy consumption by 26% and latency by 33%, at the cost of a 1.4% drop in accuracy. This is especially important because it yields significant improvements in energy and latency at a negligible cost to accuracy.

Statistical significance: All pairwise differences between AW-DQN and each baseline are statistically significant ($p < 0.01$), based on paired t-tests, making our results reliable and reproducible.

Computational efficiency: The inference time of the trained DQN agent is less than 10 ms; hence, it can be used for real-time adaptation in latency-sensitive IoT applications.

7.3 Architectural insights

The comparative analysis across the three network architectures indicates the most significant gains of AW-DQN are achieved in resource-constrained scenarios such as dense WSNs (24% energy improvement), while providing consistent improvements in the cloud-fog hybrid architecture (13% accuracy gain). This underlines the importance of architecture-aware optimisation and that adaptive weighting strategies are particularly useful for heterogeneous and dynamic IoT deployments.

Moreover, the Pareto front analysis shows that AW-DQN always finds non-dominated solutions for all three architectures. Unlike the static baselines, which are dominated in at least one objective, the proposed framework can successfully find the optimal trade-off surface, which is a characteristic of a good MOO.

The validation is against the state of the art and the performance of AW-DQN is comparable to recent advances in the literature of DQN-based reinforcement learning methods:

(1) Our 31% energy reduction looks good compared to 33% reported in DVFS-enabled edge computing.

(2) Our 28% reduction in latency is comparable to the 24% reductions presented by multi-agent DQN approaches.

(3) Our 19% accuracy gain is consistent with the 19.2% idle-time reduction for DQN-IoT systems.

The comparisons validate the competitiveness of our approach and position it as a viable solution for adaptive resource allocation in next-generation IoT networks.

7.3 Limitations

The proposed framework has some limitations as follows:

Training overhead: The DQN agent requires offline training for about 150 episodes before it converges to a stable policy. This could limit the usefulness of such agents in situations that require rapid deployment or ongoing learning without pretraining.

Static topologies: The three IoT network topologies are modelled as static networks. The current formulation does not consider dynamic network reconfiguration, which is ubiquitous in IoT deployments (e.g., device mobility).

Security omissions: The optimisation framework does not include security metrics such as jamming resilience, data integrity, and confidentiality. This is a major failure in the system's ability to operate successfully in real-world environments.

Simulation-only validation: Validation for this framework was performed using MATLAB simulations only. Simulation results are still to be validated on hardware testbeds (e.g. LoRaWAN, ZigBee or IEEE 802.15.4) to bridge the simulation-to-reality gap.

7.4 Recommendations for future research

This work paves the way to several promising extensions that we intend to explore in future research. All directions are defined with specific implementation steps and measurable metrics that can be tested to ensure practical impact.

Federated DQN Agents for Distributed Learning: We intend to extend the single-agent DQN framework to a federated learning setting over multiple IoT gateways. This allows privacy-preserving, distributed training without exchanging raw sensor data Implementation steps: (1) Deploy a federated averaging algorithm (FedAvg) over 5–10 edge gateways, each with its own local DQN replicating the architecture described in Section 3.1; (2) Synchronise global model weights every 10 local training episodes; (3) Evaluate the communication overhead and model convergence under varying network bandwidth (50–500 kbps). Testable metrics: Global model accuracy, communication cost (MB per synchronisation), and convergence time (episodes to reach 95% of centralised performance).

DT Integration for Pre-Deployment Validation: We aim to incorporate DT technology to create a virtual replica of the physical IoT environment. This allows safe and rapid testing of resource allocation policies before over-the-air deployment. Implementation steps: (1) Develop a DT using MATLAB Simulink or Unity 3D that mirrors the three network architectures (dense WSN, edge-enabled, cloud-fog hybrid); (2) Inject realistic traffic traces and channel impairments (Rayleigh fading, interference) into the DT; (3) Compare the DT-optimised policies against the simulation results in Section 4 to quantify the fidelity gap. Testable metrics: Policy transferability (measured as the performance degradation Δ when moving from DT to real hardware), and simulation-to-reality gap (% difference in energy, latency, and accuracy).

Security-Aware Optimisation with Jamming and Integrity Metrics: We will extend the MOO framework to include security objectives such as resilience to jamming, data integrity, and confidentiality. This addresses a critical gap for real-world deployments in hostile environments. Implementation steps: (1) Model jamming attacks as an additional channel impairment (SNR reduction by 5–15 dB) and integrity breaches as packet corruption rates (5–20%); (2) Formulate a fourth objective $S(\theta)$ representing security resilience (measured as the inverse of attack success probability); (3) Redefine the reward function to incorporate a weighted security term $w_4 \cdot \tilde{S}$ Testable metrics: Attack detection rate, false positive rate and tradeoff between security and QoS objectives (energy/latency/accuracy degradation under attack).

Dynamic Topology Adaptation Using Hierarchical Reinforcement Learning: To enable device mobility and fluctuating connectivity, we will design a hierarchical reinforcement learning (HRL) framework to jointly optimise topology reconfiguration (e.g., gateway reassignment) and resource allocation. Steps to implement: (1) propose a two-level HRL architecture, with the high-level policy determining the gateway handover decision (every 50 time steps), and the low-level policy performing the DQN weight adjustment (every time step); (2) Device mobility simulation with random waypoint and Gauss-Markov models. (3) Test performance when the topology suddenly changes (e.g., 20% of devices go out of range). Testable metrics: handover latency (ms), topology reconfiguration overhead (energy cost per handover), QoS stability (variance of energy, latency and accuracy before/after reconfiguration).

Pareto Front Reduction through Dimensionality Analysis: We will explore dimensionality reduction techniques to identify redundant objectives and reduce computational complexity, enabling scalability to large-scale IoT networks (e.g., $N > 500$). Implementation steps: (1) Conduct Principal Component Analysis (PCA) and correlation analysis on the objective space (energy, latency, accuracy) on 10,000 random decision vectors; (2) Verify whether any objective can be represented as a linear combination of the other objectives (thus reducing the MOO problem to a bi-objective or even single-objective problem); (3) Compare the solution quality (hypervolume difference) of the reduced model with the full

MOO framework; Metrics for testing: Hypervolume loss (if decreased), reduction in computational time (%), and similarity of Pareto front (Generational Distance).

Hardware Validation over Physical IoT Testbeds (LoRaWAN, ZigBee and IEEE 802.15.4): Experimental validation on commodity IoT hardware will bridge the simulation-to-reality gap. This is the most important practical step. Steps to implementation: (1) Testbed A (LoRaWAN): Set up 20-50 LoRa nodes (e.g., Dragino LoRa Shield + Arduino Uno) that are connected to a LoRaWAN gateway (e.g., The Things Network). Configure the spreading factors (SF 7-12) and bandwidth (125 kHz) to match the simulation parameters. (2) Testbed B (ZigBee): Configure 20 ZigBee nodes (e.g., XBee Series 2) with a coordinator to use the 2.4 GHz band and measure the impact of Wi-Fi interference. (3) Implementation: Port the DQN agent (trained offline in MATLAB) to an edge gateway (e.g., Raspberry Pi 4 or NVIDIA Jetson Nano) using Python and TensorFlow Lite for real-time inference (target inference time < 10 ms as measured in simulation). (4) Test cases: We perform the evaluation under three channel conditions, i.e., line-of-sight (LOS), non-LOS (with obstacles), and dynamic mobility (nodes moving at 1-2 m/s). Testable metrics: real-world energy consumption (mAh measured by current sensors), end-to-end latency (packet timestamps), and data accuracy (comparison of received sensor values to ground truth). Use these metrics to compare to simulation predictions, quantify the gap, and recalibrate the simulation model.

6G Network Slicing and Terahertz Communication Integration The framework will be extended to meet the stringent requirements of 6G-enabled IoT, i.e., ultra-low latency (sub-1ms), massive connectivity (>10^6 devices/km^2), and terahertz (THz) communication. How to do it: (1) Extend the network model by including the characteristics of the THz channel (e.g., molecular absorption loss, high path loss); (2) Include network slicing as a new decision variable with its own QoS profile (e.g., URLLC, mMTC, eMBB); (3) Redefine the state space s_t to incorporate the slice-specific queue lengths and resource blocks. Metrics to test include slice isolation (inter-slice interference), resource utilisation efficiency, and compliance with 6G KPIs (e.g., 99.999% reliability, 1 ms latency).

REFERENCES

[1] Saad, W., Bennis, M., Chen, M. (2019). A vision of 6G wireless systems: Applications, trends, technologies, and open research problems. IEEE Network, 34(3): 134-142. https://doi.org/10.1109/MNET.001.1900287

[2] Yedurkar, D.P., Schlech, T., Sause, M.G. (2025). A systematic review on smart and predictive maintenance in tool condition monitoring. IEEE Access, 13: 106246-106286. https://doi.org/10.1109/ACCESS.2025.3579204

[3] Datta, P., Garg, N., Aggarwal, A., Kaur, A., Gulzar, Y. (2026). Green and intelligent IoT in 6G era: Machine learning for sustainable smart cities. In Synergies of 6G and IoT, pp. 57-82. https://doi.org/10.1002/9781394370702.ch3

[4] Das, M., Nag, A., Hassan, M.M., Santra, A., Chand, N., Yasmin, F., Alkhayyat, A. (2024). Synergy of 6G technology and IoT networks for transformative applications. International Journal of Communication Systems, 37(14): e5869. https://doi.org/10.1002/dac.5869

[5] Khan, M.N., Ullah, I., Bashir, A.K., et al. (2025). Reinforcement learning-based dynamic power management for energy optimization in IoT-enabled consumer electronics. IEEE Transactions on Consumer Electronics, 71(3): 8103-8114. https://doi.org/10.1109/TCE.2025.3570503

[6] Alazzawi, M.Q., Sánchez-Aarnoutse, J.C., Martínez-Sala, A.S., Cano, M.D. (2025). Green IoT: Energy efficiency, renewable integration, and security implications. IET Networks, 14(1): e70003. https://doi.org/10.1049/ntw2.70003

[7] Li, Z., Zhao, Z., Shi, J., et al. (2023). Multiobjective deep reinforcement learning assisted resource allocation for MEC-caching-coexist system. IEEE Internet of Things Journal, 11(4): 6158-6170. https://doi.org/10.1109/JIOT.2023.3309856

[8] Dritsas, E., Trigka, M. (2025). Federated learning for IoT: A survey of techniques, challenges, and applications. Journal of Sensor and Actuator Networks, 14(1): 9. https://doi.org/10.3390/jsan14010009

[9] Han, C., Chai, Z., Li, Y. (2025). Distributed task offloading in edge computing: A multi-objective adaptive deep reinforcement learning algorithm. Engineering Applications of Artificial Intelligence, 162, 112653. https://doi.org/10.1016/j.engappai.2025.112653

[10] Sun, L., Liu, S., Muhammad, G. (2025). FedWFC: Federated learning with weighted fuzzy clustering for handling heterogeneous data in MIoT networks. Alexandria Engineering Journal, 111: 194-202. https://doi.org/10.1016/j.aej.2024.10.033

[11] Abbass, W., Abbas, N., Majeed, U. (2026). SimEdgeAI: A deep reinforcement learning framework for simulating task offloading in resource-constrained IoT networks. Simulation Modelling Practice and Theory, 146: 103226. https://doi.org/10.1016/j.simpat.2025.103226

[12] Joshi, A., Mishra, P.K. (2026). Digital twin-assisted reinforcement learning for resource allocation in 6G networks. In 2026 Third International Conference on Innovations in Cybersecurity and Data Science (ICICDS), Pathum Thani, Thailand, pp. 151-156. https://doi.org/10.1109/ICICDS70526.2026.11604762

[13] Whaiduzzaman, M., Monalisa, N.T., Himi, S.T., Sultana, S., Jan, T., Barros, A. (2026). Enhancing IIoT security using digital twins in Industry 5.0: A systematic literature review. Information, 17(2): 209. https://doi.org/10.3390/info17020209

[14] Yang, N., Wen, J., Zhang, M., Tang, M. (2025). Generalizable Pareto-optimal offloading with reinforcement learning in mobile edge computing. IEEE Transactions on Services Computing, 18(6): 3824-3836. https://doi.org/10.1109/TSC.2025.3604371

[15] Wu, H., Wang, C., Xing, L., Ma, H., Zheng, R., Song, X. (2026). PGN-MO-DDQN: A preference-driven multi-objective offloading algorithm for mobile edge video analytics. Ad Hoc Networks, 182. https://doi.org/10.1016/j.adhoc.2025.104100

[16] Ren, J., Chen, G., Chai, Z.Y., Yuan, D. (2026). A multi-objective deep reinforcement learning algorithm for computation offloading in Internet of Vehicles. Computers, Materials & Continua, 86(1): 1-26. https://doi.org/10.32604/cmc.2025.068795

[17] Khoshvaght, P., Haider, A., Rahmani, A.M., et al. (2025). A multi-objective deep reinforcement learning algorithm for spatio-temporal latency optimization in

mobile IoT-enabled edge computing networks. Simulation Modelling Practice and Theory, 143: 103161. https://doi.org/10.1016/j.simpat.2025.103161

[18] Bhushan, A., Kumar Singh, A., Sachan, S., Mishra, S. (2025). Improving energy efficiency in IoT edge computing through dynamic deep Q-network based resource management. Software: Practice and Experience, 55(12): 1947-1974. https://doi.org/10.1002/spe.70017

[19] Wu, P., Guan, Y. (2025). Multi-agent deep reinforcement learning for computation offloading in cooperative edge network. Journal of Intelligent Information Systems, 63(2): 567-591. https://doi.org/10.1007/s10844-024-00907-3

[20] Adhikari, B., Jaseemuddin, M., Anpalagan, A. (2024). Resource allocation for co-existence of eMBB and URLLC services in 6G wireless networks: A survey. IEEE Access, 12: 552-581. https://doi.org/10.1109/ACCESS.2023.3343250

[21] Mali, S., Zeng, F., Adhikari, D., et al. (2025). Federated reinforcement learning-based dynamic resource allocation and task scheduling in edge for IoT applications. Sensors, 25(7): 2197. https://doi.org/10.3390/s25072197

[22] Li, J., Dai, M., Su, Z. (2020). Energy-aware task offloading in the Internet of Things. IEEE Wireless Communications, 27(5): 112-117. https://doi.org/10.1109/MWC.001.1900495

[23] Dias, R.L., Vilas Boas, E.C., P. de Figueiredo, F.A., Mafra, S.B., Ouameur, M.A. (2026). Data compression in LoRa networks: Performance and energy trade-offs of classical and cutting-edge compression algorithms. Sensors, 26(5): 1414. https://doi.org/10.3390/s26051414

[24] Saeidi Mobarakeh, Z., Amoozadkhalili, H. (2024). Non-linear multi-objective optimization model of production planning based on fuzzy logic and machine learning. Journal of Engineering Management and Soft Computing, 10(1): 155-189. https://doi.org/10.22091/jemsc.2024.11186.1197

[25] Parizi, M.I., Ghourtani, M.R., Nunez, D., et al. (2026). DQN-based fair resource unit allocation for multi-AP coordinated OFDMA Wi-Fi networks. IEEE Open Journal of the Communications Society, 7: 6225-6239. https://doi.org/10.1109/OJCOMS.2026.3687844

[26] Singh, S.P., Kumar, N., Kumar, G., Balusamy, B., Bashir, A.K., Al Dabel, M.M. (2025). Enhancing quality of service in IoT-WSN through edge-enabled multi-objective optimization. IEEE Transactions on Consumer Electronics, 71(2): 4110-4119. https://doi.org/10.1109/TCE.2025.3526992

[27] Tong, H., Chen, C., Jiang, W., Wang, T., Zhu, J. (2025). Adaptive edge task offloading via parameterized multi-objective reinforcement learning with hybrid action space. IEEE Transactions on Network Science and Engineering, 12(6): 4876-4893. https://doi.org/10.1109/TNSE.2025.3577628

[28] Zhang, W., Ou, H. (2025). Reinforcement learning based multi objective task scheduling for energy efficient and cost effective cloud edge computing. Scientific Reports, 15(1): 41716. https://doi.org/10.1038/s41598-025-25666-1

[29] Zhang, J., Ning, Z., Waqas, M., Alasmary, H., Tu, S., Chen, S. (2024). Hybrid edge-cloud collaborator resource scheduling approach based on deep reinforcement learning and multiobjective optimization. IEEE Transactions on Computers, 73(1): 192-205. https://doi.org/10.1109/TC.2023.3326977

[30] Zhang, Z., Zhao, Y., Li, H., Lin, C., Liu, J. (2024). DVFO: Learning-based DVFS for energy-efficient edge-cloud collaborative inference. IEEE Transactions on Mobile Computing, 23(10): 9042-9059. https://doi.org/10.1109/TMC.2024.3357218