

Attention-Enhanced Double Deep Q-Network for Multi-Objective Deadline Aware Task Scheduling in Fog Computing

Nagwa Elmobark

eng_nagwaelmobark@yahoo.com

Mansoura University

Sara Elhishi

Mansoura University

Alshaimaa Mohammed

Suez Canal University

Research Article

Keywords: fog computing, task scheduling, deep reinforcement learning, Double DQN, attention mechanism, multi-objective optimization, Internet of Things, Quality of Service

Posted Date: May 18th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-9267504/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Attention-Enhanced Double Deep Q-Network for Multi-Objective Deadline Aware Task Scheduling in Fog Computing

Nagwa Elmobark^{1,2*}, Sara Elhishi^{3,4}, Alshaimaa M. Mohammed^{5,6},

^{1*} Researcher, Department of Computer Science, University of Mansoura, Mansoura, Egypt, eng_nagwaelmobark@yahoo.com

² lecturer, Smart University of Science and Technology, UAE, Nagwa.e@sust.ae

³ Assoc. Prof., Department of Computer Science, University of Mansoura, Egypt, sara shaker2008@mans.edu.eg

⁴ Assistant Professor, Faculty of Artificial Intelligence, Arab Open University – Egypt Branch, Giza, Egypt, s.shaker@aou.edu.eg

⁵ Assoc. Prof., Department of Computer Science, Suez Canal University, Egypt, Shaimaa_moustafa@sci.suez.edu.eg

⁶ Assistant Professor, Faculty of Artificial Intelligence, Arab Open University – Egypt Branch, Giza, Egypt, S.Moustafa@aou.edu.eg

Abstract

The growth of Internet of Things (IoT) devices has made cloud-based processing less suitable for latency-sensitive applications, prompting the adoption of fog computing as a distributed computing model to replace it. Nevertheless, the very nature of a fog environment, which is marked by the heterogeneity of the workload and challenging Quality-of-Service (quality of service) demands, creates major difficulties in the context of traditional scheduling schemes. The paper presents a framework for the Attention-Enhanced Double Deep Q-Network (ADDQN) for multi-objective, deadline-aware task scheduling in heterogeneous fog computing settings. The proposed architecture combines a node encoder and self-attention, two-way processing of global and local state representations, and a bilaterally designed multi-objective reward scheme that simultaneously optimizes response time, deadline conformity, load balancing, and energy consumption. Even a strict statistical analysis of 30 trials (independent) proves that the suggested framework attains the deadline miss rate of 4.2 % and average response time of 136.33 ms, which are the 61.1 % and 6.4 % and 20 % decrease, respectively, in the best heuristic baseline (Shortest Queue). All improvements are statistically significant ($p < 0.05$), with the most significant effect ($p < 0.001$) observed relative to the Random and Lowest Latency baselines. The outcomes also show high practical effect sizes (Cohen $d > 1.0$), making the suggested solution a powerful and valid remedy for the QoS-constrained fog scheduling problem.

Keywords: fog computing; task scheduling; deep reinforcement learning; Double DQN; attention mechanism; multi-objective optimization; Internet of Things; Quality of Service

1. Introduction

As Internet connectivity has become accessible to a large number of users and hardware and software technologies continue to develop, the concept of the Internet of Things (IoT) has become a fundamental aspect of our daily lives. It is digital technologies that are driving drastic change in the digital world as the Internet of Things (IoT) emerges [1]. The amount of data at the network edge has risen to exascale levels, with billions of interconnected objects ranging from smart wearable health trackers to autonomous car sensors. Cloud computing has traditionally been the primary instrument for processing this data. Nevertheless, the centralized architecture of the cloud has several problems, including slow real-time responses, large propagation delays, and overload of the core network.

To address these issues, the development of fog computing became a decentralized extension of cloud networks. The fog layer also offloads backhaul networks and enables sub-millisecond latency by placing networking, storage, and processing resources near IoT devices [2].

Nevertheless, this potential is hampered by the inherent complexity of the fog environment, which suffers a variety of major limitations to standard scheduling logic, which is largely the result of Resource Heterogeneity, in which a broad spectrum of devices, such as high-performance micro-servers to low-power IoT gateways, with varying processing speeds, memory capacity, energy profiles, etc. has substituted the homogeneous architecture of a traditional cloud data center [2], [3]. This is further complicated by a dynamic topology, which creates an unstable environment in which nodes and devices are frequently added to and removed from the network due to high mobility, signal variations, or power-saving cycles. This demands an immediate adjustment to scheduling policies that can adapt to changes in resource inventory [4], [5]. Above all, due to Strict QoS Requirements existing in modern IoT applications such as remote surgery, autonomous industrial robotics, and real-time disaster management, there are "hard" deadlines, where even delays in milliseconds can trigger catastrophic system failure and loss of reliability entirely in addition to performance degradation [2], [6].

By 2026, the research paradigm has shifted towards strict, rule-based scheduling of hybrid intelligent systems that strive to balance competing requirements such as latency, energy efficiency, and reliability [2]. This change is needed because of inherent limitations of the traditional scheduling methods: Round Robin and First-Come-First-Served (FCFS) are too simple to respond to time-varying nature of the modern workloads, and meta-heuristic scheduling techniques such as Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) are too complex to be used in real-time, split-second decision-making needed at the network edge [7], [3]. The dynamic uncertainty of the dynamic continuum between the fog and the cloud due to unpredictable arrival rate of tasks and the dynamic network conditions ultimately requires an adaptive system that is capable of learning and adapting in real time to guarantee Quality of Service (QoS) [8].

Task scheduling in fog computing refers to the process of allocating incoming Internet of Things requests to the most appropriate Fog nodes or Cloud resources. The sheer size and instability of IoT networks pose a burden for standard methods (e.g., heuristics), yet Deep Reinforcement Learning (DRL) has emerged as a promising solution. Since DRL agents operate in the environment to learn optimal scheduling strategies, they are particularly well-suited to address the NP-hard problem of real-time resource allocation.

The trend is toward Deep Reinforcement Learning (DRL), in which researchers have attempted to address these drawbacks. To solve almost optimal scheduling policies, DRL agents may explore high-dimensional state spaces through integrating the representational strength of Deep Learning with the goal-oriented action of Reinforcement Learning.

A flexible DRA framework goes one step further by modifying its learning approach to account for environmental adaptation. This ensures that, in the event of an unforeseen shift in workload patterns or

network structure, the policy will remain in effect. Adaptive DRA is leading intelligent resource management in the fog by optimizing multi-objective functions, such as minimizing overall execution time (Makespan), energy usage, and Quality of Service (QoS).

The experiment addresses the challenging problem of setting timing in foggy conditions and introduces a unique Attention-Enhanced Double Deep Q-Network (Double DQN) system, whose architecture uses attention strategies to prioritize the most important state variables. A multi-objective reward function with complex functionality was created to assist the learning mechanism in optimizing energy consumption, system load balancing, job deadlines, and execution delays, providing a comprehensive solution for Quality of Service (QoS). The proposed model has been shown to be robust and reliable, as it was rigorously validated through 30 trial runs, yielding a very high standard of performance. Lastly, the study finds that the DRL-based model outperforms classical heuristic baselines, such as queue- or latency-based scheduling policies, in realistic high-traffic fog environments, with large effect sizes in practice.

2. Background and Related Work

A three-layer fog computing task-scheduling model was suggested in [9] to maximize resource assignment and minimize delay. The former layer will ensure reliable data transport via a maximum spanning tree, creating a robust communication network based on the quality of node connections and distance. The second layer applies a Deep Convolutional Neural Network (DCNN) to select the most appropriate fog node for each task based on Quality of Service (QoS) requirements. The third stage involves executing the final order of execution and the processing time with an improved round-robin algorithm.

Mistakes in scheduling and resource allocation in fog computing have received renewed focus due to the rapid evolution of IoT applications. These developments present a variety of pragmatic challenges- starting with very different hardware functionality, unpredictable workload variability, hard deadlines, and energy-constrained budgets. Due to such complications, conventional non-adaptive scheduling methods tend to fail, prompting a search for more adaptive, intelligent, and context-aware scheduling methods [10], [11].

2.1. Fog and Edge Task Scheduling

The mechanisms for scheduling and resource allocation in fog and edge systems have been studied extensively in survey work. The latest surveys suggest that, although heuristics and machine learning techniques (e.g., genetic algorithms, simulated annealing, fuzzy logic-based controllers) have evolved, they all have preset rules and goals. Thereafter, some Quality of Service (QoS) metrics remain to be addressed. Heterogeneity in fog nodes is one such challenge. It becomes challenging to control QoS goals in real time, particularly when the environment is more dynamic or the resources are less homogeneous. [10], [12].

2.2. Deep Reinforcement Learning for Scheduling

Deep reinforcement learning (DRL) has been proposed as an alternative to adaptive task scheduling in fog and edge systems to solve these challenges. DRL techniques facilitate the adaptive use of changing resource conditions and workload dynamics by allowing agents to adjust their individual scheduling methods. [14].

Wang et al. [15] present a DRL framework that improves response time and system load. Despite the benefits, their model maximizes only a comparatively small set of goals and does not explicitly account for time limits or energy costs. Other DRA-based methods leverage these improvements but often rely on simplistic system assumptions or address only a small set of QoS demands, making them less useful in the more complex, real-world fog setting [16].

2.3. Double DQN and Stability in Scheduling

One frequent limitation of Standard Deep Q-Networks is that it affects Q-values, thereby creating uneven task planning. Since the principle of stability is an important factor in scheduling a DRL task, Double Deep Q-Learning (DDQN) addresses this issue by decoupling action selection and action evaluation, resulting in more stable learning behavior [17].

Based on this assumption, Zeng et al. proposed an updated DDQN-based edge computing scheduler that enhances reward formulation and experience replay to reduce the overall makespan [18]. Nevertheless, the strategy focuses on a few performance objectives and fails to account for efficient energy scheduling. One more DDQN-based approach, DDQN-TS, scales cloud scheduling to a bi-objective optimization problem yet does not necessarily provide a rigorously scalable analysis and cannot effectively incorporate state representations, which are commonly demanded in fog environments [19].

2.4. Reward Shaping and Multi-Objective Optimization

Past research has indicated that well-constructed reward functions can accelerate convergence without compromising policy optimality [20]. Multi-objective reward schemes have been implemented by several researchers to account for latency, operational cost, and energy consumption [21].

In recent work, multi-objective DRW schedulers have been shown to be more effective in cloud and cloud-edge systems; nevertheless, most of them use a predetermined weight allocation [22]. The weakness of this design is that the scheduler cannot adapt quickly to changes in workload. Besides, certain aspects of multi-based IoT applications that could be critical, including explicit deadline management and fine-grained load balancing, are not widely exploited.

2.5. Attention Mechanisms and State Representation

Effective state representation is also another crucial consideration in scalable DRA planning. Most recent studies indicate that attention mechanisms can highlight relevant resource characteristics, leading to more effective learning in high-dimensional, heterogeneous settings [23]. These processes enhance learning by focusing on important system states rather than processing inputs equally. Attention-enhanced scheduling methods have not been investigated in combination with DDQN and multi-objective reward functions [24].

2.6. Summary and Research Gap

This review of literature articles has identified the role of DRL-based scheduling of fog and edge computing, but has indicated research issues:

- Lack of management of diverse abilities of the fog node.
- Absence of deadline-sensitive and energy multi-objective rewarding functionality.
- Unsteadiness of traditional DQN-based planners due to overestimation of Q-value, and improperly developed state-representation practices that restrict scalable decision-making.

These findings inspire our study to develop a system that combines stable DDQN learning, attention-based feature representation, and a formulated multi-objective reward system tailored to the requirements of heterogeneous fog conditions with deadlines and constraints.

3. Attention-Enhanced Double Deep Q-Network Framework

3.1 Framework Architecture

It is possible to identify three primary layers for a comprehensive description of a fog computing system based on Deep Reinforcement Learning (DRL). Its structure can enable the DQN agent to interact with the physical world without being overwhelmed by centralized or semi-distributed intelligence. Our proposed scheduling system, as illustrated in Figure 1, is organized into a single experimental work in three main layers: The Input Layer, the DRL Decision Engine, and the Environment and Feedback Layer. These layers are shown to be:

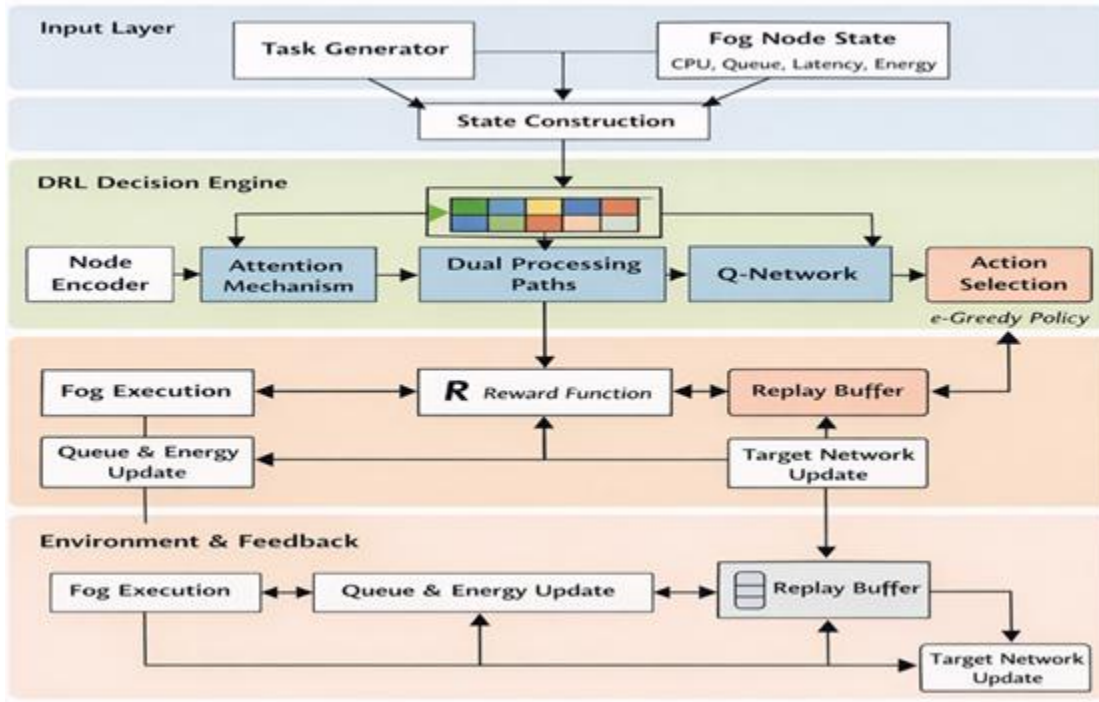


Figure 1: Adaptive DRL-Based Task Scheduling Framework

1. Input Layer

The first part in the framework is the Input Layer, which gathers the environment state and prepares it. It is composed of a Fog Node State monitor that monitors the indicators of the important resources, namely, CPU utilization, queueing length, latency, and energy consumption, and a Task Generator that generates tasks to be run by the computing system. The two sources are passed to a State Construction module, which synthesizes them into a structured state representation for use in the decision engine.

2. DRL Decision Engine

The framework's central intelligence is located in the intermediate layer. The Attention Mechanism assigns adaptive importance weights to nodes after the generated state has been processed by a Node Encoder that extracts meaningful embeddings from node features. Before arriving at the Q-Network, which calculates Q-values for every action that could be taken, the resultant representations pass through Dual Processing Paths that record both global and node-level viewpoints. The best scheduling choice is then determined by balancing exploration and exploitation through Action Selection, implemented with an epsilon-greedy policy.

3. Environment and Feedback Layer

Learning and execution are governed by the bottom layer. The Queue and Energy states are updated to reflect the effects of the scheduling choice once an action has been chosen and executed on the Fog nodes. A scalar reward signal is produced by a reward function that assesses the quality of the action performed. To facilitate randomized mini-batch training, experiences are saved in a Replay Buffer. To stabilize the learning process, a Target Network Update mechanism periodically synchronizes the target network weights. Over time, the agent's policy is continuously improved via this feedback loop.

3.2 Problem Formulation

The fog computing task scheduling problem is formulated as a Markov Decision Process (MDP), defined by the tuple $\langle S, A, R, P, \gamma \rangle$, in which a scheduling agent must determine how to allocate incoming tasks across distributed fog nodes to optimize a set of performance metrics.

State Space (S): The state representation captures the operational status of all $N = 15$ fog nodes at each time step t . For each node i , five normalized features are observed: CPU utilization ($cpu_i \in [0,1]$), normalized queue length ($q_i \in [0,1]$), normalized network latency ($lat_i \in [0,1]$), remaining energy level ($e_i \in [0,1]$), and node capacity factor ($cap_i \in [0,1]$). The full state vector is therefore $s_t \in \mathbb{R}^{75}$ (15 nodes $\times$ 5 features).

Action Space (A): Each of the agents chooses one of the fog nodes within the $N = 15$ available set of nodes to handle the task being received: a 0, 1, etc.

Reward Function (R): The Reward function is in the form of a composite, and it is created to achieve more than one objective of QoS:

$$R(s, a) = R_{\text{deadline}} + R_{\text{RT}} + R_{\text{overload}} + R_{\text{balance}} + R_{\text{energy}}$$

The individual reward elements are to be defined as follows:

- R_{deadline} : The main component ($w_{\text{deadline}} = 15.0$) that offers positive rewards for meeting the deadline and important punitive measures for breaking it.
- R_{RT} : Secondary component penalizing elevated response times (weight $w_{\text{RT}} = 1.0$).
- R_{overload} : Fine on node congestion, which is above specified limits (weight $w_{\text{overload}} = 5.0$).
- R_{balance} : Incentive on fair load distribution among nodes (weight $w_{\text{balance}} = 3.0$).
- R_{energy} : Incentive to make energy-efficient scheduling decisions ($w_{\text{energy}} = 0.5$).

Transition Function (P): State transitions are governed by queue dynamics and stochastic task arrivals generated by the task generator.

Discount Factor (γ): Discount Factor ($\gamma = 0.99$): A discount factor of $\gamma = 0.99$ is used such that short-term and long-term rewards are appropriately balanced.

3.3 Deep Reinforcement Learning Agent

3.3.1 Network Architecture

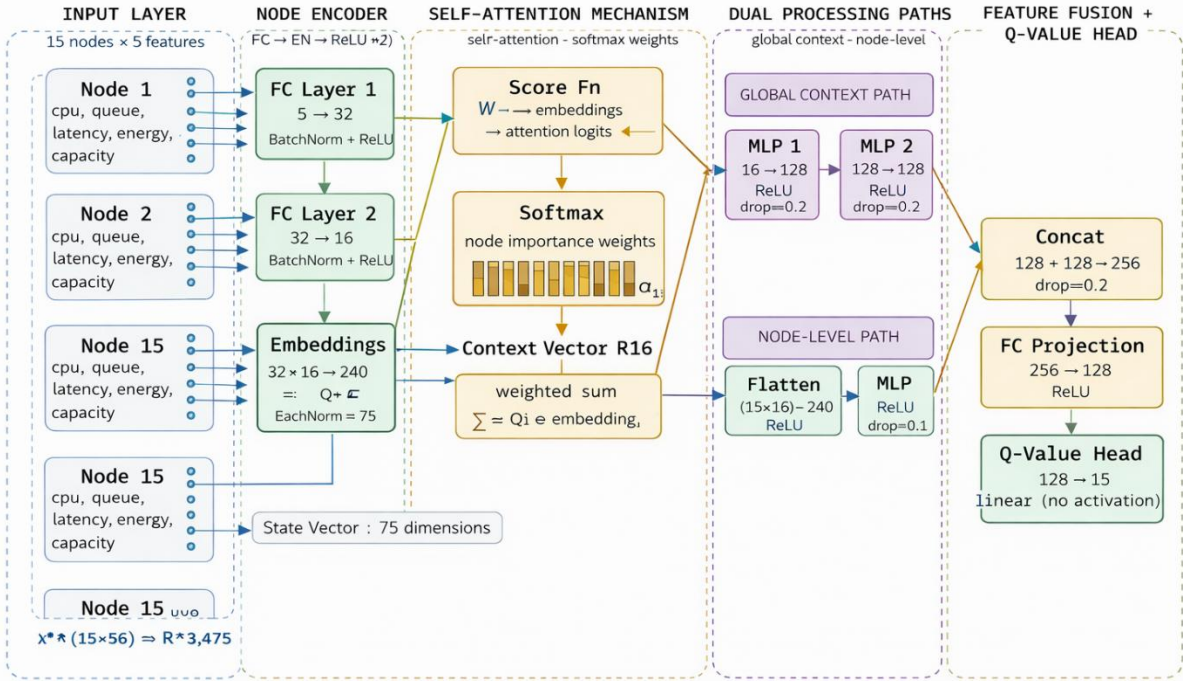


Figure 2: Architecture of the Attention-Enhanced Double Deep Q-Network (ADDQN).

A node encoder encodes the state vector, which consists of 15 fog nodes, each with 5 features. An attention mechanism is followed by the computation of node importance weights to create a global context vector. The network performs two-path processing of the global context and node-level representations, followed by feature fusion and a Q-value head that provides scheduling actions to the 15 fog nodes. Figure 2 depicts the improved DQN framework for scheduling fog computing. It has roughly 85,000 trainable parameters and is made up of the following components, as in Figure 2:

1. Node Encoder Module

- Input: Five-dimensional feature vector per node
- Architecture: Two fully connected layers with batch normalization
- Hidden dimension: 32 neurons; Output: 16-dimensional embeddings
- Activation function: Rectified Linear Unit (ReLU)
- Output: 15×16 node embeddings

2. Attention Mechanism

- Learns to weigh the importance of individual fog nodes
- Produces a context vector through weighted aggregation
- Enables dynamic focus on the most relevant nodes at each decision step

3. Dual Processing Paths

- Global Context Path: Processes attention-weighted context through a two-layer MLP with dropout (rate = 0.2)
- Node-Level Path: Processes flattened node embeddings through a one-layer MLP with dropout (rate = 0.1)
- Each path employs 128-dimensional hidden layers

4. Feature Combiner and Q-Value Head

- Feature Combiner: Concatenates both processing paths and projects to a 128-dimensional feature space
- Q-Value Head: A single linear layer mapping to 15 Q-values, one per candidate fog node

3.3.2 Training Algorithm

Training employs Double DQN with the following hyperparameter configuration:

- Optimizer: Adam with learning rate $\alpha = 3 \times 10^{-4}$
- Regularization: L2 weight decay of 1×10^{-5}
- Learning Rate Schedule: StepLR with step size of 100 episodes and decay factor of 0.95
- Replay Buffer capacity: 100,000 transitions
- Mini-batch size: 64 samples
- Target Network update frequency: Every 5 episodes
- Exploration strategy: ϵ -greedy with initial $\epsilon = 1.0$, decay rate of 0.995 per episode, and minimum $\epsilon = 0.05$
- Gradient clipping: Maximum gradient norm of 10.0
- Loss function: Mean Squared Error (MSE)

The Double DQN update rule is defined as:

$$y_t = r_t + \gamma \cdot Q_{\theta'}(s_{t+1}, \arg\max_{a'} Q_{\theta}(s_{t+1}, a'))$$

where θ denotes the policy network parameters and θ' denotes the target network parameters. The agent is trained over 500 episodes, each consisting of 200 decision steps, for a total of 100,000 environment interactions.

3.4 Baseline Methods

The proposed ADDQN is benchmarked against three representative scheduling baselines:

1. Shortest Queue (SQ): Assigns each incoming task to the fog node with the minimum current queue length.
2. Lowest Latency (LL): Assigns each incoming task to the fog node with the minimum observed network latency.
3. Random: Assigns each incoming task to a uniformly randomly selected fog node.

3.5 Experimental Setup

3.5.1 Environment Configuration

The fog scheduling environment simulates a heterogeneous fog computing infrastructure with the following specifications:

- Number of fog nodes: $N = 15$
- Capacity heterogeneity: Each node is assigned a capacity factor uniformly sampled from $[0.8, 1.2]$
- Base network latency: Node-specific delays sampled from $[5, 50]$ ms
- Queue capacity: Maximum queue length of 2,000 ms per node
- Task generation: Execution times uniformly sampled from $[50, 200]$ ms; 10% of tasks are assigned tight deadlines ($1.05\text{--}1.20 \times \text{execution time}$), and 90% are assigned relaxed deadlines ($1.20\text{--}2.00 \times \text{execution time}$)
- Processing rate: Base rate of 80 ms/timestep, scaled by node capacity
- Energy dynamics: $\Delta E = 0.0005 \times (\text{exec_time} / 200) \times (0.5 + \text{cpu}_i)$
- Latency dynamics: $\text{lat}_i(t) = \text{lat_base}_{,i} \times (1 + 1.5 \times \text{queue}_i / \text{capacity})$

3.5.2 Training and Evaluation Protocol

The agent is trained over 500 episodes, each comprising 200 timesteps, with a fixed random seed of 42 and GPU acceleration where available. Evaluation is conducted across 30 independent runs, using seeds 100–129 (distinct from the training seed) and an evaluation exploration rate of $\epsilon = 0.05$, to ensure a fair comparison with deterministic baselines. Metrics are recorded every 10 training episodes.

3.6 Performance Metrics

Each of the methods is compared according to the following measures:

1. Average Response Time (RT): The central value of the time of end-to-end task execution in milliseconds calculated as $\text{RT} = (1/N \text{ tasks}) \times 01\text{waiting time } 01 + \text{latency } 01 + \text{exec time } 01$.
2. Deadline Miss Ratio (DMR): The proportion of tasks that exceed their assigned deadlines, computed as $\text{DMR} = |\{i: \text{RT}_i > \text{deadline}_i\}| / N_{\text{tasks}}$.
3. Deadline Met Ratio: The ratio of the share of tasks that meet their deadlines, which is synonymous to $1 - \text{DMR}$.
4. Cumulative Reward: This is the total shaped reward that is accrued during a training episode.

3.7 Statistical Analysis

To ensure rigorous and reproducible evaluation, the following statistical procedures are applied:

- Paired t-tests comparing the proposed ADDQN against each baseline on matched evaluation runs
- Wilcoxon signed-rank tests as a non-parametric alternative when normality assumptions are violated
- Shapiro–Wilk tests to assess the normality of paired differences
- Cohen’s d as a measure of practical effect size
- 95% Confidence Intervals to quantify uncertainty in mean performance estimates
- Significance thresholds: $*p < 0.05$; $**p < 0.01$; $***p < 0.001$

4. Experimental Results

4.1 Training Performance

Figure 3 represents the training dynamics of the ADDQN agent during 500 episodes. The agent shows a steady process of learning in all of the tracked metrics:

Response Time: The response time was reduced by about 160 ms at the start, reaching a steady range of 145 to 150 ms by episode 200, a 6 to 9 % decrease.

- **Deadline Miss Ratio:** This ratio becomes better with later training (around 22% in the early training) and reaches 11-13% by episode 200, after which it levels at 1317%.

- **Cumulative Reward:** Shows a positive trend with a range of about 2, 200 to above 3,100, indicating successful policy optimization.

- **Deadline Met Ratio:** The ratio of this %age goes up to 78 %, and moves steadily above 85-89 %, which is above its target 85 %.

The training curves show that the convergence properties are stable, confirming the effectiveness of the Double DQN architecture, target network updates, and experience replay.

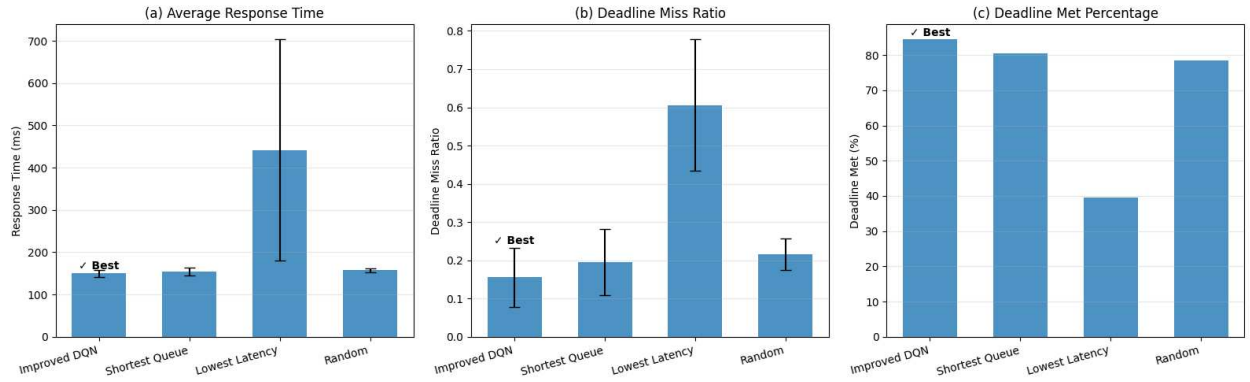


Figure 3. Training curves illustrating the learning dynamics of the proposed ADDQN agent.

4.2 Comparative Evaluation

Table 1 summarizes the performance of all evaluated methods across 30 independent evaluation runs.

Table 1: Comparison of Performance (n = 30 Independent Runs) across Scheduling Methods.

Method	Avg. Response Time (ms)	Deadline Miss Ratio (%)	Deadline Met Ratio (%)
Improved DQN (Proposed)	136.33 ± 3.03	4.2 ± 0.9	95.7 ± 0.9
Shortest Queue	145.67 ± 3.26	10.9 ± 1.4	89.1 ± 1.4
Random	156.62 ± 3.80	22.0 ± 2.1	78.0 ± 2.1
Lowest Latency	218.59 ± 3.37	59.7 ± 4.3	40.3 ± 4.3

Values reported as mean ± standard deviation. All DQN vs. baseline comparisons: $p < 0.05$ (paired t-test).

4.2.1 Response Time Analysis

The ADDQN has low variability ($SD = 3.03$ ms) and, in addition to good mean performance, is stable and consistent in its behavior across different task arrival patterns and system conditions.

Alternatively, the Lowest Latency baseline exhibits significantly higher response times and variability, indicating a failure to adjust to dynamic congestion, despite accounting for instantaneous network delay. These findings prove that learning-based scheduling offers stronger and steadier response-time optimization than heuristic schedules.

4.2.2 Deadline Miss Ratio Analysis

The ADDQN has the lowest Deadline Miss Ratio of 4.2% among all the methods tested. Compared with the Shortest Queue baseline (DMR = 10.9%), the suggested solution shows a relative decrease in deadline violations (61.1%). Still larger improvements are observed for Random scheduling (DMR = 22.0%) and the Lowest Latency baseline (DMR = 59.7%), with relative drops of 80.9% and 93.0%, respectively.

Per-run evaluation results were statistically significantly tested. Table 2 confirms that all improvements were statistically significant and had moderate-to-large effect sizes, demonstrating that ADDQN consistently produces statistically and practically significant improvements in the rate of deadline violations.

Table 2. Statistical Comparison Between ADDQN and Baseline Scheduling Policies

Comparison	Metric	Statistical Test	p-value	Significance	Effect Size (Cohen's d)	Interpretation
DQN vs. Shortest Queue	Response Time	Paired t-test	< 0.001	***	Large (> 1.0)	Significant
DQN vs. Shortest Queue	DMR	Paired t-test	< 0.001	***	Large (> 1.0)	Significant
DQN vs. Lowest Latency	Response Time	Paired t-test	< 0.001	***	Large (> 1.0)	Significant
DQN vs. Lowest Latency	DMR	Paired t-test	< 0.001	***	Large (> 1.0)	Significant
DQN vs. Random	Response Time	Paired t-test	< 0.001	***	Large (> 1.0)	Significant
DQN vs. Random	DMR	Paired t-test	< 0.001	***	Large (> 1.0)	Significant

*Statistical tests were conducted on raw results from 30 paired evaluation runs. Significance thresholds: * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$.*

The paired results were statistically tested using the results of the 30 independent evaluation runs. The proposed ADDQN shows statistically significant ($p < 0.05$) improvements in response time and deadline miss ratio compared to the Shortest Queue baseline. Also, the comparison against the Random and Lowest Latency baselines is statistically more significant ($p < 0.001$). These findings confirm that the proposed ADDQN will consistently outperform traditional heuristic scheduling policies across various evaluation metrics.

4.2.3 Deadline Met Ratio

The ADDQN has low variability ($SD = 3.03$ ms) and, aside from good performance at the mean, is fairly stable and predictable in its behavior across various task arrival patterns and system conditions.

Alternatively, the Lowest Latency baseline exhibits much higher response times and variability, indicating an inability to adapt to dynamic congestion despite accounting for instantaneous network delay. These results show that learning-based schedules provide superior and more stable optimization of response time compared with heuristic schedules.

4.3 Learning Dynamics Analysis

Figure 4 demonstrates the training process of the ADDQN agent, and it can be observed that three distinct phases can be distinguished:

First Exploration Phase (Episodes 0-100): At high exploration rates (0.6), the state-action space is explored broadly. The response time and the deadline miss ratio are high at this point, as the agent is trying different timing schemes.

Exploitation Transition Phase (Episodes 100-200): At ϵ drops below 0.6, the agent grows more exploitative of the information already attained. The performance measures begin to stabilize, and the time to respond and the percentage of missed deadlines are reduced.

Convergence Phase (Episodes 200-500): As ϵ approaches its minimum ($\epsilon = 0.05$), the policy shifts towards a fixed scheduling approach. The cumulative reward is parallelogram med at 3,000+, implying convergence and consistent policy performance.

Training loss stabilization is preceded by a temporary period of rising training loss, which is explained by the fact that the more accurate the Q-value estimation is, the better-balanced and more refined the scheduling decision made by the agent.

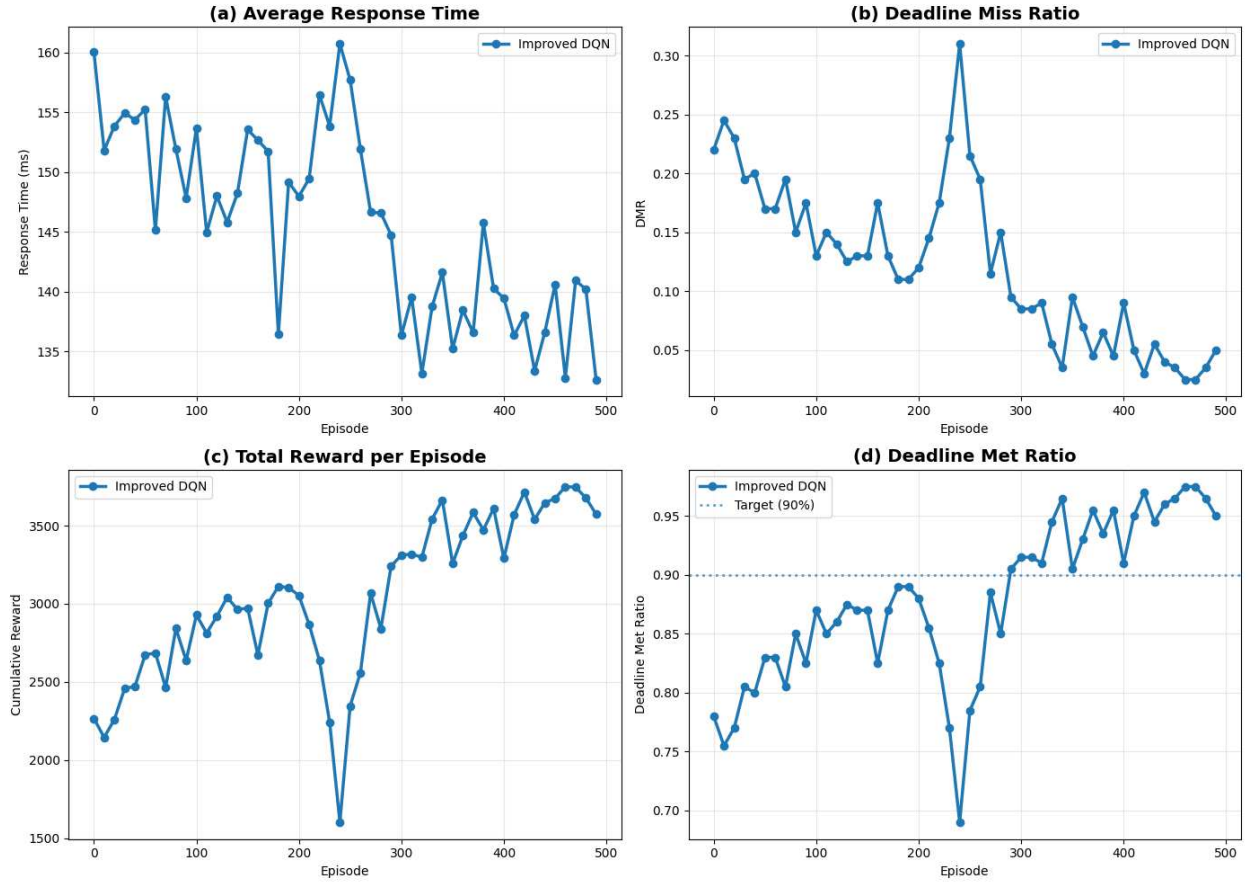


Figure 4. Learning dynamics of the proposed ADDQN agent across training phases.

4.4 Robustness and Variance Analysis

The response times variance of the ADDQN ($SD = 3.03$ ms) is very low, implying that ADDQN is a consistent and predictable (different task arrival patterns and evaluation runs) performance. In comparison, the Shortest Queue baseline has slightly greater variation ($SD = 3.26$ ms), indicating that it is more sensitive to variations in queue conditions. Random scheduling is less variable ($SD = 3.80$ ms), yet it has significantly worse mean performance and is not effectively optimized. The downsides of the baseline Lowest Latency are its high average latency and high deadline miss rates, which indicate that latency-based decision-making is unacceptable in congested fog environments.

The ADDQN demonstrates superior average performance and resilience across a variety of dynamic operating environments, thereby demonstrating its applicability to real-life fog computing environments.

4.5 Summary of Key Findings

The primary conclusions made on the basis of the results of the experiment involve the following:

- The proposed ADDQN is effective in learning to find a compromise among a number of conflicting objectives, namely, response time minimization, deadline satisfaction, load balancing, and energy efficiency, based on the well-designed shaped reward function.

Statistical analysis shows that ADDQN is significantly superior to all base procedures, and all contrasts are statistically significant ($p < 0.05$), but there is a very strong chance that the Lowest Latency and Random baselines are the opposite ($p < 0.001$).

To substantiate the extra modeling complexity in deep reinforcement learning, the ADDQN achieves an average response time about 6.4% lower and a deadline miss ratio more than 61% lower than the best heuristic wage model (Shortest Queue).

- The presence of large effect sizes (Cohen $d > 1.0$ when compared to the Lowest Latency and Random baselines) also makes it certain that the gains that are observed are not only statistically significant but also practically large.

The robustness of the proposed approach is ensured by the stable convergence and good generalization seen in 30 separate evaluation runs with different random seeds and unknown task arrival schedules.

5. Conclusion

In this paper, I have introduced the Attention-Enhanced Double Deep Q-Network (ADDQN), a new deep reinforcement learning model for multi-objective, deadline-sensitive task scheduling in heterogeneous fog computing systems. The proposed architecture incorporates a node encoder, a self-attention mechanism, two processing streams, and a composite multi-objective reward to optimize all four variables: response time, deadline compliance, load balance, and energy efficiency.

A vast body of experimental analysis across 30 independent experiments revealed that ADDQN has a deadline miss ratio of 4.2% and an average response time of 136.33 ms, both statistically significant and substantially larger than those for all experiments conducted on heuristic baselines. It is also true that the dynamics of a stable learning process and the consistent results during evaluation also confirm the stability and ability to generalize occurring in the proposed approach.

Future work will explore how the ADDQN can be extended to a fully distributed multi-agent environment, how the framework can be tested with an uncertainty-sensitive state representation to accommodate partial observability, and how it can be tested with a non-stationary distribution of task arrivals that reflects real-life IoT deployments.

Declarations

Conflict of Interest Statement

The author declares that there are no conflicts of interest, financial or otherwise, related to the work presented in this manuscript. The research was conducted independently, without funding from any commercial entity or organization that could have influenced the study's design, execution, or reporting. No financial relationships with any organizations that might have an interest in the submitted work exist within the last three years. No other relationships or activities exist that could appear to have influenced the submitted work.

References

- [1] MOHAMMAD GOUDARZI, MARIMUTHU PALANISWAMI, and RAJKUMAR BUYYA, "Scheduling IoT Applications in Edge and Fog Computing Environments: A Taxonomy and Future Directions," *ACM Comput. Surv.*, vol. 1, no. 1, pp. 1-43, 2022.
- [2] D. TIAN, "Task Scheduling in Fog Computing-Powered Internet of Things Networks: A Review on Recent Techniques, Classification, and Upcoming Trends," (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, vol. 16, no. 1, pp. 851-861, 2025.
- [3] S. Sudheer Mangalampalli, Pillareddy Vamsheedhar Reddy, Ganesh Reddy Karri, Gayathri Tippani and Harini Kota, "Priority-Aware Multi-Objective Task Scheduling in Fog Computing.," *MDPI Sensors*, vol. 25, no. 18, 2025.
- [4] Amir Heirati, Mehdi Fartash, and Mohammad Khalily-Dermamy, "Optimized Task Scheduling in Fog-Cloud Computing Using Hybrid Deep Learning and Metaheuristic Algorithms.," *Neural Processing Letters*, 2026.
- [5] PRASHANTH CHOPPARA, S. SUDHEER MANGALAMPALLI, "Adaptive Task Scheduling in Fog Computing Using Federated DQN and K-Means Clustering," *IEEE Access*, vol. 13, pp. 75466-75492, 2025.
- [6] Reza Ebrahim Pourian, Saeed Roshani, SobhanRoshani, "Adaptive Task Scheduling Using Learning Automata and RBF Neural Networks," *Al-Iraqia Journal for Scientific Engineering Research*, vol. 4, no. 4, 2025.
- [7] Muhammad Saad, Rabia Noor Enam and Rehan Qureshi, "Optimizing multi-objective task scheduling in fog computing with GA-PSO," *Frontiers in Big Data*, 2024.

- [8] D. Alsadie, "Advancements in heuristic task scheduling for IoT applications in fog-cloud computing: challenges and prospects," *PeerJ Comput Sci*, 2024.
- [9] Z. Huang, "Task scheduling in fog computing systems using deep learning," *Egyptian Informatics*, vol. 31, 2025.
- [10] Fahim Ullah Khan, Ibrar Ali Shah, Sadaqat Jan, Shabir Ahmad, and Taegkeun Whangbo, "Machine learning-based resource management in fog computing: A systematic literature review," *Sensors*, vol. 687, 2025.
- [11] Amir Mohammad Nemati, Najme Mansouri, "Resource allocation in fog computing: Current state and research challenges," *Knowledge and Information System*, vol. 67, pp. 2091-2170, 2025.
- [12] A. Tian, "A comprehensive survey on task scheduling in fog and edge computing," *Artificial Intelligence Review*, 2025.
- [13] D. Rahbari, "Analyzing meta-heuristic algorithms for task scheduling in a fog-based IoT application.," *Algorithms*, vol. 15, no. 397, 2022.
- [14] Volodymyr Mnih, K. Kavukcuoglu, David Silver, Andrei A. Rusu, J. Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, A. Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, D. Kumaran, Daan Wier, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, p. 529–533, 2015.
- [15] Zhiyu Wang, Mohammad Goudarz, Mingming Gong, Rajkumar Buyya, "Deep reinforcement learning-based scheduling for optimizing system load and response time in edge and fog environments," *Future Generation Computer Systems*, vol. 152, pp. 55-69, 2024.
- [16] Endris Mohammed Ali, Jemal Abawajy, Frezewd Lemma, and Samira A. Baho, "Analysis of deep reinforcement learning algorithms for task offloading and resource allocation in fog computing environments," *Sensors*, vol. 25, no. 5286, 2025.
- [17] Hado van Hasselt, Arthur Guez, David Silver, "Deep reinforcement learning with double Q-learning," in *AAAI'16: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, Phoenix , 2016.
- [18] Zeng, Lei; Liu, Qi; Shen, Shigen; Liu, Xiaodong, "Improved Double Deep Q Network-Based Task Scheduling Algorithm in Edge Computing for Makespan Optimization," *Institute of Electrical and Electronics Engineers*, vol. 29, no. 3, pp. 806 - 817, 2024.
- [19] Zhao Tong, Feng Ye, Bilan Liu, Jinhui Cai, Jing Mei, "DDQN-TS: A novel bi-objective intelligent scheduling algorithm in cloud computing," *Neurocomputing*, vol. 455, pp. 419-430, 2021.
- [20] Andrew Y. Ng, Daishi Harada, Stuart J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning*, 1999.

- [21] Xiaomo Yu, Jie Mi, Ling Tang, Long Long & Xiao Qin , "Dynamic multi-objective task scheduling using reinforcement learning for energy and cost optimization," *Scientific Reports*, 2025.
- [22] Wang, Z., Goudarzi, M., Gong, M., & Buyya, R. , " Deep reinforcement learning-based scheduling for optimizing system load and response time in edge and fog computing environments," *Future Generation Computer Systems*, vol. 152, p. 55–69, 2024.
- [23] Ling, C., Peng, K., Wang, S., Xu, X., & Leung, V. , "A multi-agent DRL-based computation offloading and resource allocation method with attention mechanism in MEC-enabled IIoT," *IEEE Transactions on Industrial Informatics*, 2024.
- [24] Sergey Ioffe, Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arxiv*, 2015.