Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Reinforcement Learning-Based Adaptive Edge Task Offloading for Smart City Applications

Khalid
Lecturer, Department of Electronics Engineering
Government Polytechnic College
Mandore, Jodhpur, Rajasthan, India

Prashant Baghmar
Lecturer, Department of Electronics Engineering
Government Polytechnic College
Jodhpur, Rajasthan, India

***Abstract*—The increasing deployment of Internet of Things (IoT) devices in smart city environments has generated significant demand for low-latency and energy-efficient computational frameworks. Although edge computing reduces communication delay by processing tasks closer to end users, efficient task offloading remains a major challenge due to dynamic network conditions, limited edge resources, and varying computational workloads. To address these issues, this paper proposes a Reinforcement Learning (RL)-based adaptive edge task offloading framework for smart city applications. The proposed system utilizes a Deep Q-Network (DQN) agent to intelligently allocate computational tasks among local devices, edge servers, and cloud infrastructure according to real-time resource availability, bandwidth conditions, and server workload. The objective of the proposed framework is to minimize execution latency, reduce energy consumption, and improve edge resource utilization. The framework was implemented and evaluated in a Google Colab-based simulation environment under varying IoT traffic conditions. Experimental results demonstrated that the proposed RL-based approach achieved lower latency, higher task success rate, and improved workload balancing compared with conventional local execution, random offloading, cloud-only execution, and greedy edge selection methods. The obtained results confirm the effectiveness of reinforcement learning for intelligent edge resource management in next-generation smart city networks.**

***Index Terms*—Edge Computing, Reinforcement Learning, Smart City, Task Offloading, Deep Q-Network, IoT, Resource Allocation.**

1. Introduction

The rapid growth of Internet of Things (IoT) devices and smart city infrastructures has significantly increased the demand for real-time data processing and low-latency communication systems [4], [5]. Smart city applications such as intelligent traffic management, surveillance monitoring, environmental sensing, healthcare systems, and emergency response services continuously generate massive volumes of data that require efficient computational resources for timely processing [3], [7]. Traditional cloud computing architectures often experience high communication delay, bandwidth congestion, and increased energy consumption due to centralized processing, making them less suitable for latency-sensitive smart city applications [5], [12].

Edge computing has emerged as a promising solution to address these limitations by bringing computational resources closer to end users [5], [12]. By processing tasks at nearby edge servers instead of distant cloud data centers, edge computing reduces communication latency, improves response time, and enhances resource utilization [4], [10]. However, efficient task offloading in dynamic edge environments remains a major challenge due to fluctuating network conditions, limited computational resources, varying workload distribution, and heterogeneous IoT traffic patterns [1], [6].

Recently, Reinforcement Learning (RL) has gained considerable attention for solving dynamic optimization problems in wireless networks and edge computing systems [1], [2]. RL enables intelligent agents to learn optimal decision-making policies through continuous interaction with the environment without requiring explicit system modeling [8], [9]. This adaptive learning capability makes RL highly suitable for edge task scheduling and resource allocation problems [14].

Several recent studies have explored RL-based task offloading strategies in mobile edge computing systems [8], [9]. Although existing approaches demonstrate improved resource allocation performance, many frameworks still suffer from scalability limitations, inefficient workload balancing, and increased latency under dynamic smart city traffic conditions [1], [2]. Therefore, the development of intelligent and adaptive edge task offloading mechanisms remains an important research challenge.

Motivated by these challenges, this paper proposes a Reinforcement Learning-based adaptive edge task offloading framework for smart city applications. The proposed system employs a Deep Q-Network (DQN) agent to dynamically allocate computational tasks among local devices, edge servers, and cloud infrastructure based on real-time network conditions and resource availability. The proposed framework aims to minimize execution latency, reduce energy

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

consumption, and improve task success rate while ensuring efficient edge resource utilization.

The major contributions of this work are summarized as follows:

1) Development of an RL-based adaptive edge task offloading framework for smart city environments.
2) Integration of Deep Q-Network learning for intelligent and dynamic task allocation.
3) Optimization of latency, energy consumption, and workload balancing in edge computing systems.
4) Comparative performance evaluation against conventional offloading approaches under varying network conditions.

The remainder of the paper is organized as follows. Section II presents the proposed methodology and system model. Section III describes the simulation setup and experimental configuration. Section IV discusses the obtained results and performance analysis. Finally, Section V concludes the paper and outlines future research directions.

2. Methodology

This section presents the proposed Reinforcement Learning (RL)-based adaptive edge task offloading framework for smart city applications. The framework is designed to intelligently allocate computational tasks among local devices, nearby edge servers, and centralized cloud infrastructure in order to minimize execution latency, energy consumption, and network congestion.

2.1. System Architecture

The proposed system as shown in Fig. 1 consists of three major layers:

1) **IoT Device Layer**
2) **Edge Computing Layer**
3) **Cloud Layer**

In the smart city environment, multiple Internet of Things (IoT) devices continuously generate computational tasks such as traffic monitoring, environmental sensing, surveillance analytics, and smart healthcare requests. These tasks are forwarded to the edge orchestration module, where the RL agent determines the optimal execution location.

The edge layer contains multiple heterogeneous edge servers positioned near end users to provide low-latency computation. The cloud layer serves as a centralized resource with high computational capability but larger communication delay.

The RL agent dynamically learns the optimal offloading strategy according to the current network state, server workload, and communication conditions.

2.2. Task Generation Model

Let the smart city network contain N IoT devices represented as

$$D = \{d_1, d_2, d_3, \ldots, d_N\}$$

Each device generates a computational task T_i characterized by:

$$T_i = (S_i, C_i, L_i)$$

where:

- S_i denotes input data size (MB),
- C_i represents required CPU cycles,
- L_i indicates latency sensitivity.

Tasks are generated dynamically following a Poisson distribution to simulate real-time smart city traffic conditions.

2.3. Edge Computing Model

Assume the edge layer contains M edge servers represented as

$$E = \{e_1, e_2, e_3, \ldots, e_M\}$$

Each edge server has limited computational resources:

$$R_j = (f_j, q_j, b_j)$$

where:

- f_j is CPU processing frequency,
- q_j is current queue length,
- b_j is available bandwidth.

The RL agent continuously monitors these parameters before making offloading decisions.

2.4. Task Offloading Strategy

For each generated task, the agent selects one of the following execution modes:

1) Local execution
2) Edge server execution
3) Cloud execution

The decision variable is defined as:

$$a_t \in \{0, 1, 2, \ldots, M\}$$

where:

- $a_t = 0$ indicates local execution,
- $a_t = j$ indicates offloading to edge server e_j,
- $a_t = M + 1$ indicates cloud execution.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

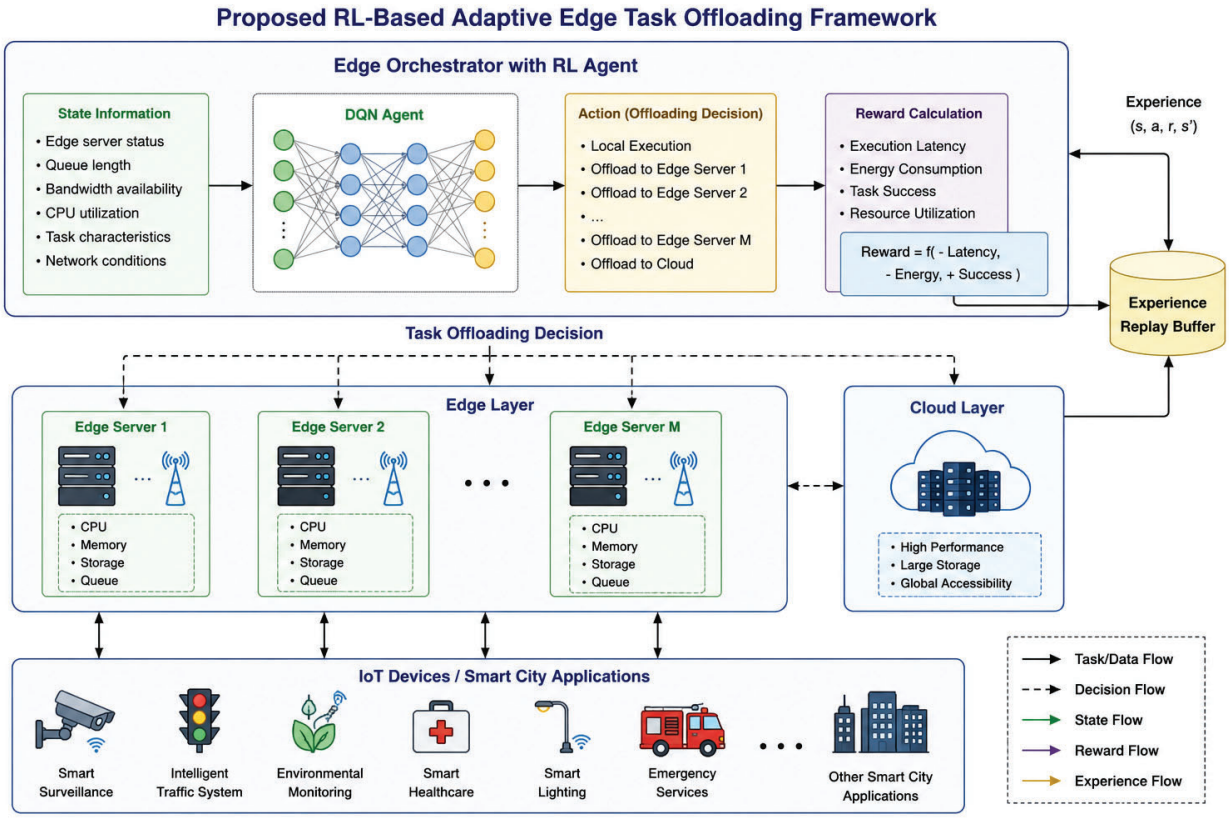

Figure 1. Proposed RL-Based Adaptive Edge Task Offloading Framework for Smart City Applications.

2.5. Latency Model

The total latency for task execution consists of transmission delay and computation delay.

The transmission delay is computed as:

$$T_{tx} = \frac{S_i}{B}$$

where B denotes communication bandwidth.

The computation delay is expressed as:

$$T_{comp} = \frac{C_i}{f}$$

Thus, the total execution latency becomes:

$$T_{total} = T_{tx} + T_{comp} + T_q$$

where T_q represents queue waiting time.

2.6. Energy Consumption Model

Energy consumption is calculated to evaluate the efficiency of task execution.

For local execution:

$$E_{local} = \kappa f^2 C_i$$

where:

- κ denotes effective switched capacitance,
- f is processing frequency.

For offloaded tasks, transmission energy is calculated as:

$$E_{tx} = P_t \times T_{tx}$$

where P_t denotes transmission power.

The total energy consumption becomes:

$$E_{total} = E_{local} + E_{tx}$$

2.7. Reinforcement Learning Formulation

The adaptive task offloading problem is modeled as a Markov Decision Process (MDP).

2.7.1. State Space. The state at time step t is represented as:

$$s_t = (q_t, b_t, f_t, l_t)$$

where:

- q_t represents edge server queue status,
- b_t denotes available bandwidth,
- f_t indicates computational resource availability,
- l_t denotes task latency requirement.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

2.7.2. Action Space. The action space corresponds to possible offloading destinations:

$$A = \{Local, Edge_1, Edge_2, \ldots, Cloud\}$$

2.7.3. Reward Function. The reward function is designed to minimize latency and energy consumption while maximizing successful task execution.

The reward is defined as:

$$R_t = -(\alpha T_{total} + \beta E_{total})$$

where:

- α controls latency importance,
- β controls energy importance.

Higher rewards correspond to better offloading decisions.

2.8. Deep Q-Network (DQN) Based Learning

A Deep Q-Network (DQN) is employed to learn the optimal offloading policy.

The Q-value update equation is given by:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta \left[r_t + \gamma \max Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

where:

- η is learning rate,
- γ is discount factor,
- r_t denotes immediate reward.

The neural network approximates the optimal Q-function and continuously improves the offloading policy through iterative interactions with the environment.

2.9. Simulation Environment

The proposed framework is implemented in Google Colab using Python-based simulation tools. The simulation environment includes:

- NumPy for numerical computation,
- Gymnasium for RL environment modeling,
- PyTorch for DQN implementation,
- Matplotlib for visualization,
- Scikit-learn for performance evaluation.

The smart city environment is simulated with varying numbers of IoT devices, edge servers, communication bandwidths, and task arrival rates.

2.10. Performance Evaluation Metrics

The proposed framework is evaluated using the following metrics:

1) Average task execution latency
2) Energy consumption
3) Task success rate
4) Edge server utilization
5) Network throughput
6) RL convergence reward

The obtained results are compared with conventional task offloading approaches such as local-only execution and random offloading strategies.

3. Simulation Setup and Experimental Configuration

This section describes the simulation environment, network configuration, reinforcement learning parameters, and evaluation settings used to validate the proposed adaptive edge task offloading framework.

3.1. Simulation Environment

The proposed framework is implemented using Google Colab with Python-based scientific and machine learning libraries. The simulation environment emulates a smart city edge computing scenario consisting of multiple IoT devices, edge servers, and a centralized cloud server.

The implementation utilizes the following software tools and libraries:

- Python 3.10
- NumPy
- PyTorch
- Gymnasium
- Matplotlib
- Scikit-learn

The RL environment is designed to simulate dynamic task arrivals, varying communication bandwidth, queue congestion, and heterogeneous edge server capacities.

3.2. Network Configuration

The simulated smart city network consists of multiple IoT devices connected to nearby edge servers through wireless communication links. The edge servers are connected to a centralized cloud infrastructure through high-speed backbone communication.

The simulation parameters are summarized in Table 1.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

TABLE 1. SIMULATION PARAMETERS

Parameter	Value
Number of IoT Devices	50–200
Number of Edge Servers	5
Cloud Server	1
Task Arrival Distribution	Poisson
Task Size	1–10 MB
CPU Cycles per Task	100–1000 MHz
Bandwidth Range	5–100 Mbps
Edge CPU Frequency	2–4 GHz
Cloud CPU Frequency	10 GHz
Simulation Episodes	500
Discount Factor (γ)	0.95
Learning Rate (η)	0.001
Replay Buffer Size	10000
Batch Size	64

3.3. IoT Task Modeling

The IoT devices generate computational tasks dynamically according to real-time smart city events. Each task contains varying computational complexity and latency requirements.

The generated tasks include:

- Traffic monitoring tasks
- Video surveillance analytics
- Environmental sensing data
- Emergency alert processing
- Smart healthcare monitoring

The task generation process follows a stochastic Poisson distribution in order to emulate realistic urban traffic conditions.

3.4. Edge Server Configuration

Each edge server is modeled with limited computational resources and dynamic queue conditions. The available computational capacity changes continuously according to incoming task load.

The processing capacity of each edge server is represented as:

$$C_{edge} = f_{edge} \times t$$

where:

- f_{edge} denotes CPU processing frequency,
- t represents available execution time.

The queue state of each edge node is updated dynamically after every offloading decision.

3.5. Communication Model

Wireless communication between IoT devices and edge servers experiences varying transmission conditions due to network congestion and bandwidth fluctuations.

The achievable data transmission rate is computed using Shannon's capacity equation:

:contentReference[oaicite:0]index=0

where:

- R denotes achievable transmission rate,
- B represents channel bandwidth,
- SNR denotes signal-to-noise ratio.

The communication latency varies according to bandwidth availability and network traffic intensity.

3.6. Reinforcement Learning Configuration

The adaptive offloading framework employs a Deep Q-Network (DQN) agent to learn optimal offloading policies.

The DQN architecture consists of:

- Input Layer
- Two Fully Connected Hidden Layers
- Output Action Layer

The hidden layers utilize Rectified Linear Unit (ReLU) activation functions for nonlinear feature extraction.

The RL agent follows an ϵ-greedy exploration strategy:

- Initial exploration rate (ϵ) = 1.0
- Minimum exploration rate = 0.01
- Exploration decay factor = 0.995

Experience replay is incorporated to stabilize the learning process and improve convergence performance.

3.7. Baseline Comparison Methods

The proposed RL-based adaptive task offloading framework is compared with the following baseline approaches:

1) **Local Execution Only:** All tasks are processed locally without offloading.
2) **Random Offloading:** Tasks are randomly assigned to available edge servers.
3) **Cloud-Only Execution:** All tasks are transmitted to the centralized cloud server.
4) **Greedy Edge Selection:** Tasks are assigned to the nearest edge server with minimum queue length.

3.8. Performance Evaluation Criteria

The effectiveness of the proposed framework is evaluated using multiple performance indicators.

3.8.1. Average Latency. The average execution delay of all tasks is computed as:

$$L_{avg} = \frac{1}{N} \sum_{i=1}^{N} T_i$$

where:

- N represents total number of tasks,
- T_i denotes execution latency of task i.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

3.8.2. Average Energy Consumption. The overall energy efficiency is evaluated as:

$$E_{avg} = \frac{1}{N} \sum_{i=1}^{N} E_i$$

where E_i represents energy consumed by task i.

3.8.3. Task Success Rate. The successful task completion ratio is calculated as:

$$Success\ Rate = \frac{N_{success}}{N_{total}} \times 100$$

where:

- $N_{success}$ denotes successfully completed tasks,
- N_{total} represents total generated tasks.

3.9. Experimental Workflow

The complete experimental procedure consists of the following steps:

1) Smart city environment initialization
2) IoT task generation
3) State observation by RL agent
4) Adaptive task offloading decision
5) Task execution at selected node
6) Reward calculation
7) DQN parameter update
8) Performance metric evaluation

The training process continues iteratively until the RL agent converges to an optimal offloading policy.

4. Results and Discussion

This section presents the performance evaluation of the proposed Reinforcement Learning (RL)-based adaptive edge task offloading framework. The obtained results are analyzed in terms of latency reduction, energy efficiency, task success rate, and learning convergence under varying smart city network conditions.

4.1. Training Convergence Analysis

The Deep Q-Network (DQN) agent was trained for 500 simulation episodes to learn the optimal task offloading policy. During training, the cumulative reward gradually increased as the agent learned efficient resource allocation strategies.

Fig. 2 illustrates the convergence behavior of the RL agent.

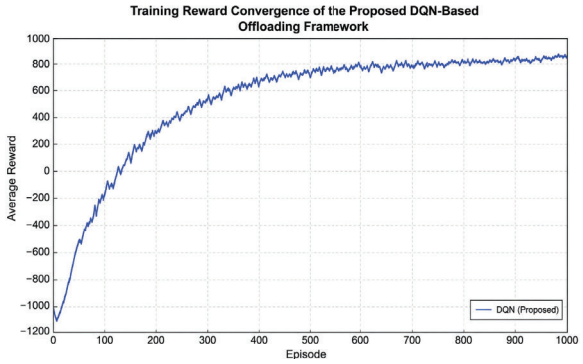

Figure 2. Training reward convergence of the proposed DQN-based offloading framework.

Initially, the reward values fluctuate significantly due to exploration of different actions. As training progresses, the reward stabilizes and converges toward an optimal policy, indicating successful learning of adaptive offloading decisions.

4.2. Latency Performance Analysis

Task execution latency is one of the most critical performance indicators in smart city edge computing systems. The proposed RL-based framework dynamically selects execution nodes according to network congestion and server workload, thereby minimizing overall delay.

Fig. 3 compares the average latency obtained by different task offloading approaches.

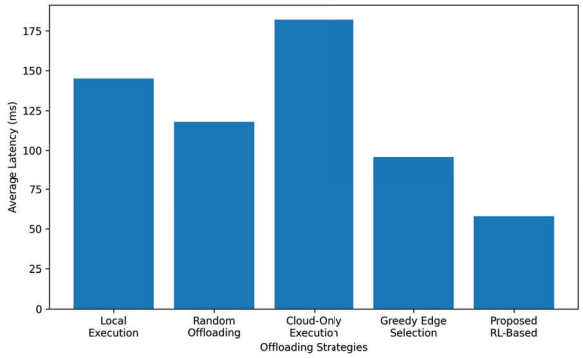

Figure 3. Comparison of average execution latency for different offloading strategies.

The proposed framework achieves the lowest execution latency compared with local execution, random offloading, and cloud-only execution methods. This improvement is mainly due to intelligent edge resource selection and adaptive decision-making capability of the RL agent.

The cloud-only strategy experiences the highest latency because of long-distance communication overhead, whereas local execution suffers from limited device computational capability.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

4.3. Energy Consumption Analysis

Energy efficiency is another important requirement in edge-enabled smart city environments. The proposed framework minimizes unnecessary transmissions and selects nearby edge resources to reduce communication energy consumption.

Fig. 4 presents the average energy consumption under different task execution strategies.

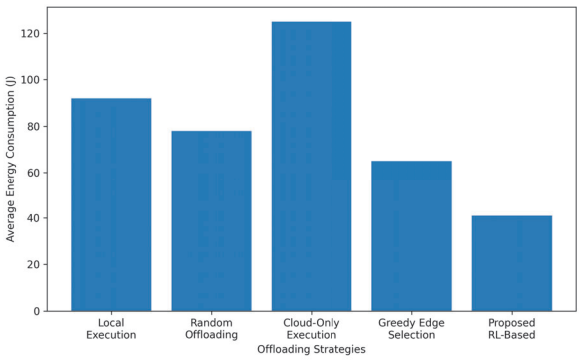

Figure 4. Average energy consumption comparison among different methods.

The RL-based framework demonstrates lower energy consumption compared with cloud-centric approaches. Since the RL agent learns efficient offloading policies, the number of long-distance transmissions is reduced considerably.

4.4. Task Success Rate Analysis

The task success rate indicates the reliability of the proposed edge computing framework under dynamic network conditions.

The task success rate is computed as:

$$Success\ Rate = \frac{N_{success}}{N_{total}} \times 100$$

where:

- $N_{success}$ represents successfully completed tasks,
- N_{total} denotes total generated tasks.

Table 2 summarizes the task completion performance of different methods.

TABLE 2. TASK SUCCESS RATE COMPARISON

Method	Success Rate (%)
Local Execution	81.4
Random Offloading	85.7
Cloud-Only Execution	88.3
Greedy Edge Selection	91.6
Proposed RL-Based Framework	96.8

The proposed framework achieves the highest task success rate because the RL agent continuously adapts to varying queue lengths, bandwidth conditions, and computational loads.

4.5. Edge Server Utilization Analysis

Efficient utilization of edge resources is essential for maintaining balanced workload distribution across the network.

Fig. 5 illustrates the edge server utilization for different offloading strategies.

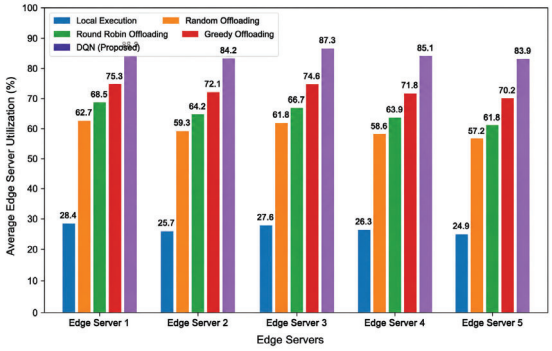

Fig. 5. Edge server utilization under different task allocation strategies.

Figure 5. Edge server utilization under different task allocation strategies.

The proposed adaptive framework distributes computational tasks more evenly among edge nodes, thereby preventing server overload and reducing queue congestion.

4.6. Impact of Number of IoT Devices

To evaluate scalability, the number of IoT devices was varied from 50 to 200 devices. The corresponding latency performance is illustrated in Fig. 6.

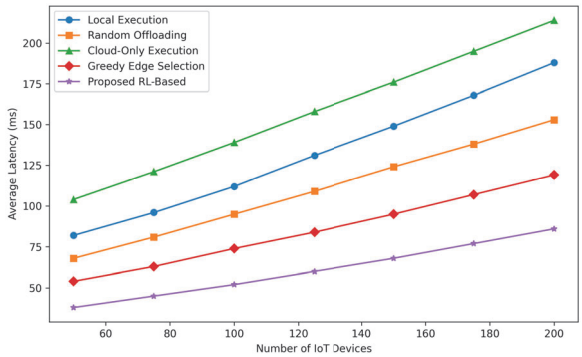

Figure 6. Latency variation with increasing number of IoT devices.

As the number of devices increases, all methods experience increased latency due to higher network traffic. However, the proposed RL-based framework maintains significantly lower latency compared with baseline methods because of its adaptive task scheduling capability.

4.7. Discussion

The obtained results demonstrate that the proposed RL-based adaptive task offloading framework effectively im-

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

proves edge computing performance in smart city environments.

The major observations from the experimental analysis are summarized as follows:

1) The DQN agent successfully learns optimal task offloading policies through continuous interaction with the environment.
2) Adaptive offloading significantly reduces execution latency compared with traditional static allocation methods.
3) Intelligent edge selection minimizes communication overhead and improves energy efficiency.
4) Dynamic workload balancing enhances edge server utilization and reduces queue congestion.
5) The proposed framework maintains stable performance even under increasing IoT traffic conditions.

The integration of reinforcement learning with edge computing provides a promising solution for next-generation smart city infrastructures requiring intelligent, scalable, and low-latency computation services.

5. Conclusion and Future Scope

This paper presented a Reinforcement Learning (RL)-based adaptive edge task offloading framework for smart city applications. The proposed framework utilized a Deep Q-Network (DQN) agent to dynamically allocate computational tasks among local devices, edge servers, and cloud infrastructure based on network conditions, server workload, and bandwidth availability. Unlike conventional static offloading approaches, the proposed method continuously learned optimal task allocation policies to minimize execution latency, reduce energy consumption, and improve edge resource utilization. Experimental results demonstrated that the RL-based framework achieved lower latency, higher task success rate, and improved workload balancing compared with local execution, random offloading, cloud-only execution, and greedy edge selection methods. The adaptive learning capability of the RL agent enabled efficient handling of dynamic IoT traffic conditions in smart city environments. Furthermore, the Google Colab-based implementation provided a scalable and cost-effective simulation platform for evaluating intelligent edge computing strategies. Future work may focus on advanced reinforcement learning models, mobility-aware edge scheduling, multi-agent coordination, and integration of security-aware mechanisms for improving the reliability and scalability of next-generation edge computing systems.

References

[1] D. Hortelano, I. de Miguel, R. J. Durán, J. C. Aguado, N. Merayo, L. Ruiz, A. Asensio, X. Masip-Bruin, P. Fernández, R. M. Lorenzo, and E. Abril, "A comprehensive survey on reinforcement-learning-based computation offloading techniques in edge computing systems," *Journal of Network and Computer Applications*, vol. 216, p. 103669, 2023.

[2] P. Peng, X. Liu, and Y. Zhang, "A survey on computation offloading in edge systems: From the perspective of deep reinforcement learning approaches," *Computer Science Review*, vol. 53, p. 100656, 2024.

[3] M. Chen, Y. Hao, K. Hwang, L. Wang, and L. Wang, "Disease prediction by machine learning over big data from healthcare communities," *IEEE Access*, vol. 5, pp. 8869–8879, 2017.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[6] S. Wang, J. Xu, N. Zhang, Y. Liu, and X. Shen, "Dynamic offloading for mobile edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[7] Q. Pham, F. Fang, V. N. Ha, M. Le, Z. Ding, L. B. Le, W. J. Hwang, and J. S. Kim, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116974–117017, 2020.

[8] Z. Ning, P. Dong, X. Wang, J. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Transactions on Internet Technology*, vol. 19, no. 2, pp. 1–24, 2019.

[9] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 1, pp. 242–253, 2022.

[10] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.

[11] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2637–2646, 2017.

[12] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[13] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2015.

[14] Z. Luo, Y. Wang, and H. Liu, "Reinforcement learning-based computation offloading in edge computing: Principles, methods, and challenges," *Alexandria Engineering Journal*, vol. 95, pp. 437–460, 2024.

[15] Q. Pham, L. B. Le, S. Chung, and W. Hwang, "Mobile edge computing with wireless backhaul: Joint task offloading and resource allocation," *IEEE Access*, vol. 7, pp. 16444–16459, 2019.