Adaptive Energy-Efficient Control of IoT Wake-Up Radio Systems using Deep Reinforcement Learning and DQN Optimization

Naman Chauhan,[1, a)] Bharathi Mohan G,[1, b)] V. Sulochana,[2, c)] and R. Prasanna Kumar[1, d)]

[1]*Department of Computer Science and Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham, Chennai, India*

[2]*Course Director, Anna Administrative Staff College, Chennai, India*

[a)]namanchauhan200293@gmail.com

[b)]g_bharathimohan@ch.amrita.edu

[c)]sulo62002@yahoo.com

[d)]Corresponding author: kumarprasanna.r@gmail.com

Abstract. Managing an intelligent scheduling system for wake- up radio in Internet of Things (IoT) networks is essential for optimizing energy consumption and extending sensor battery life. Wake-up radio technology offers unique opportunities to create efficient scheduling requests by integrating knowledge of the system state, statistical data, and sensor battery capabilities. This energy efficiency not only prolongs the lifespan of IoT devices but also enhances overall system performance and reliability by reducing energy demand. Advances in reinforcement learning, specifically Deep Q-Network (DQN) algorithms, provide effective tools for implementing intelligent scheduling. By training an agent with these techniques, we can optimize the scheduling process and further conserve energy. The proposed framework is rigorously evaluated through simulations, demonstrating its effectiveness and highlighting areas for improvement. This ap- proach significantly contributes to developing sustainable IoT ecosystems, with wide-ranging applications in smart homes, healthcare, transportation, and more.

INTRODUCTION

The Internet of Things is a transformative technological paradigm that has reshaped the landscape of digital communication and computing. In the midst of this revolution, the challenge of energy management looms large, particularly in wake-up radio systems that govern the energy states of IoT devices. These systems decide when a device should be active or in a low-energy state, making them critical to the overall energy efficiency of an IoT network. The burgeoning field of intelligent scheduling provides an intriguing solution to this pressing issue. This research paper explores the application of Reinforcement Learning and Machine Learning, with a focus on the Deep Q-Network model, to optimize intelligent scheduling for wake-up radio systems in IoT networks.

The main objective of this research is to examine the feasibility and effectiveness of using Reinforcement Learning and the Deep Q-Network model to achieve intelligent scheduling. This novel approach will be rigorously compared with traditional baseline models to evaluate its capability in optimizing energy consumption, thereby establishing its viability as a superior alternative. Wake-up radio is a technology that allows Internet of Things (IoT) nodes to respond to requests, which can be ID- or content-based (in the former case, the sensor will send its latest reading if its ID is in the request, while in the latter, it will transmit if the data matches the conditions specified in the request message). This type of system allows for interesting scheduling opportunities, particularly if the sensor measurements are correlated: by carefully crafting scheduling requests, the IoT gateway can save energy and improve its estimate of the state of the system.

The structure of this research study is as follows: Section 2 presents the study on the past works, Section 3 delves deeper into the proposed approach, Section 4 reports on the work's results and the outcomes of the proposed research, and Section 5 wraps up by talking about future endeavours.

LITERATURE REVIEW

The study by shie-yuan,et al.[1] developed a drone-based system using deep reinforcement learning (DRL) to optimize data collection from smart meters in a neighborhood, accounting for real-life wireless transmission measurements. Experimental results showed that the DRL-trained drone significantly reduced flying distance compared to other methods. The study by Ramadhani Sinde,et. al[2] introduces the Energy-Efficient Scheduling using Deep Reinforcement Learning (E2S-DRL) algorithm to enhance Wireless Sensor Network (WSN) performance. By combining clustering, duty-cycling, and routing phases, E2S-DRL significantly reduces energy consumption, decreases delays by up to 40%, and improves network lifetime and throughput by up to 35% compared to existing methods. The work by Cheng li,et al.[3] addresses the challenge of energy efficiency in Underwater Acoustic Sensor Networks (UWSNs) by proposing a deep reinforcement learning-based adaptive asynchronous wake-up scheme. The scheme optimizes idle listening policies using a Markov decision process and enhances performance with LSTM networks for traffic estimation. The study by Razehi, et al. [4] proposes a Reinforcement Learning-based Sleep Scheduling (RLS2) method for Energy-Harvesting Wireless Body Area Networks (EH-WBAN) to enhance network reliability and connectivity while ensuring self-sustainability. RLS2 optimizes the sleep/wake schedules of heterogeneous body nodes using Q-learning, resulting in improved network connectivity by 50%, energy efficiency by 31%, and reduced network delay by 27%. This review work by Hillal bello, et. al[5] examines passive wake-up radio technology for the Internet of Things (IoT), focusing on cross-layer approaches involving physical (PHY) and media access control (MAC) layers. It surveys and classifies existing RF energy harvesting techniques and hardware architectures, evaluates related MAC protocols. Work by yi chu, et. al [6] introduces ALOHA-Q, a Q-Learning-based medium access control protocol for single-hop wireless sensor networks, enhancing energy-efficiency, delay, and throughput compared to Slotted ALOHA. It offers comparable performance to S-MAC and Z-MAC with lower complexity and overheads, supported by a Markov model for convergence estimation. The findings by Robert Fromm,et. al [7] presents advancements in wake-up receivers for power-aware wireless sensor networks, introducing an improved passive envelope detector that operates at 868 MHz with a power consumption of 5.71 mW, latency of 9.02 ms, and signal detection sensitivity of -61.6 dBm. The research by subba amin, et. al[8] proposes a deep reinforcement learning-based system for IRS-assisted NOMA beamforming to enhance capacity gains and energy efficiency in next-generation wireless networks. The system optimizes phase shifts and power allocation despite unknown instantaneous CSI, energy efficiency.

PRELIMINARIES

Wireless Sensor Network Environment Design

In the development of the WSN environment[9], a range of technologies and tools were employed to ensure effective implementation, simulation, and analysis. The following subsections highlight the primary technologies leveraged.

1. **Gymnasium**:The OpenAI Gym library is a popular toolkit in the field of reinforcement learning, providing standardized environments for developing and testing algorithms. It offers a diverse suite of predefined environments, ranging from simple control tasks to game playing, and allows users to create custom environments with a standardized[11] interface.

2. **Stable Baselines 3 (SB3)**: It is a set of high-quality implementations of reinforcement learning algorithms in Python. It builds upon the foundations of Stable Baselines and OpenAI's Baselines, providing a clean and simple interface that's consistent with OpenAI Gym. The primary goal of SB3[12] is to enable efficient experimentation and development by providing well-tested and well-documented code.

Simulation Environment Design

The system model for the wireless sensor network deployed in a hostile region is detailed below. The sensor nodes are strategically placed to cover the entire region while maintaining network connectivity as illustrated in Figure 1. Each sensor node has a circular coverage area with a specified radius. The deployment of sensor nodes in the hostile region follows a Poisson point process distribution, ensuring a random and even spatial

distribution (both random and evenly spaced throughout the hostile region. The use of a Poisson point process for deployment ensures a random spatial distribution of the sensors, while maintaining a relatively uniform spacing between them. During the pulling process, the gateway sequentially pulls updates from the sensor nodes depending on the type of scheduling or the algorithm (Round-Robin, Random, RLagent,..). To determine the generated events, a threshold probability, denoted as pth, is introduced[10]. When an event is generated within the hostile region, its probability of being occurred is evaluated. If the event's probability is lower than the threshold probability (p < pth), it is considered a non-occurred event and assigned a None. Conversely, if the event's probability exceeds the threshold (p>=pth), it is deemed an occurred event and assigned a value of 1. By incorporating the threshold probability, the system allows for selective event occurrence based on the event's probability. This provides the flexibility to adjust the threshold probability and optimize the system's performance in terms of event occurrence, detection sensitivity and false positives.

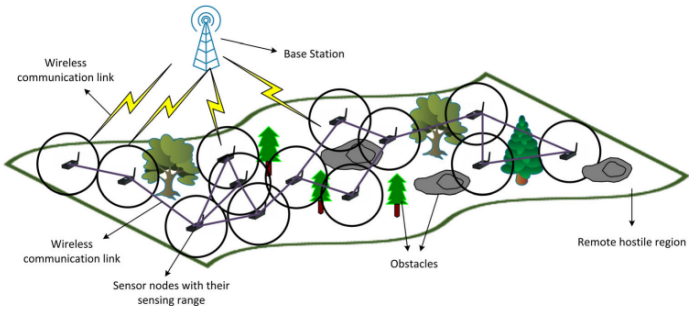

FIGURE 1. Simulation Environment Design

- **Sensor Placement and Coverage:** Sensors are deployed according to a Poisson point process distribution in a 2D space. Each sensor has 2 coordinates indicating its position, with a circular coverage radius, customizable based on the object. The hostile region has a total surface area of 1, ensuring comprehensive coverage.
- **Event Generation and Detection:** Events are generated based on a random variable exceeding a threshold probability. If an event occurs within a sensor's coverage area, it is placed in the sensor's buffer.
- **Observation and Action Spaces:** The observation space is an Nx3 matrix, with N representing the number of sensors. The first two columns indicate sensor positions, while the third column tracks sensor selection count during the episode. The action space consists of the sensor indices, ranging from 0 to N-1.
- **Reward Structure:** The reward function considers event type and selected action. For a None event, the reward is 0. If the event is None and the selected sensor's buffer is empty, a penalty of -0.4 is applied. Otherwise, the reward is calculated as seen in equation 1.

$$\text{reward} = \sum_{x \in \text{selected\_buffer}} \frac{1}{\text{AoI}} \tag{1}$$

  where, AoI (Age of Information) represents the time since the event was generated.
- **Simulation Parameters:** The default parameters for the simulation are set as: *sensor_numbers=13, sensor_coverage=0.4, max_steps=1000, threshold_prob=0.3*

METHODOLOGY

Round Robin Scheduler

The round-robin scheduling algorithm is implemented as a simple loop within the simulation environment. It was executed for a specified number of episodes and the scores for each episode were collected. The implementation of

the algorithm involves running it over a defined number of episodes, which in this case is set to 20. At the start of each episode, the environment is reset to its initial state, and variables are initialized to track the score and the current index. The action selection is performed using a Round Robin scheduler, which cycles through the available actions sequentially. The index representing the action is incremented with each step and wraps around to the start once it reaches a specified limit, RR, equal to the number of points in the environment. During each step, the selected action is executed using the 'env.step' method, and the reward obtained from this action is added to the episode's total score. The process continues, checking for termination or truncation conditions after each step, until one of these conditions is met, signaling the end of the episode. Once an episode concludes, its final score is recorded. After all episodes are completed, the scores for each episode are printed along with the average score across all episodes, providing an overview of the algorithm's performance over the entire run.The complete process flow can be understood using the figure 2.

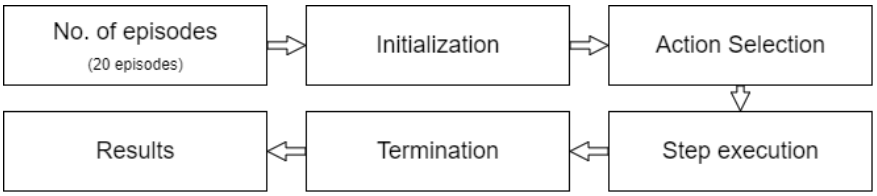

FIGURE 2. Proposed algorithm

Random scheduler

The Random scheduling algorithm is implemented as a loop within the simulation environment. it was executed for a specified number of episodes and the scores for each episode were collected as the previous scheduler. It also follows the same algorithm as used in the figure 2. The Random scheduler provides a basic and unbiased approach to action selection in the simulation. While it lacks sophistication and may lead to less efficient outcomes, it serves as a useful benchmark and introduces an element of randomness that can be valuable in testing and exploration.

Q-Learning Model

The agent was designed to implement the Q-learning algorithm in the context of a Wireless Sensor Network (WSN) environment. The following hyperparameters were used:

- Learning Rate (a): The learning rate was set to 0.01, controlling the extent to which new Q-values overwrite the old values.
- Discount Factor (Y): The discount factor was set to 0.95, determining the agent's consideration for future rewards.
- Epsilon (E): A start value of 1.0 was used for epsilon, which controls the exploration vs exploitation trade-off. This value decays linearly over time and has a final epsilon value of 0.1.

The agent used an epsilon-greedy policy, which acts randomly with a probability of E, ensuring exploration of the environment, and greedily (choosing the action with the highest estimated reward) with a probability of 1 - E, to exploit the gathered knowledge.

The Q-value update rule implemented in the agent is given by equation 2.

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot \left(r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a) \right) \tag{2}$$

where s is the current state, a is the current action, s' is the next state, r is the immediate reward, and maxa' Q(s', a') is the maximum Q-value for the next state. The agent was trained over 10, 000 episodes. Each episode begins

TABLE 1. Hyperparameters of DQN model

HYPERPARAMETER	Value
Discount Rate Y	0.99
Batch Size	32
Buffer Size	$1X10^6$
Minimum replay size	1000
Epsilon Start	1.0
Epsilon end	0.1
Epsilon Decay	$3X10^6$
Target update frequency	1000
Learning rate	$5X10^5$

with the environment being reset, followed by the agent taking actions until termination or truncation. The actions are chosen using the epsilon-greedy policy described above, and the Q-values are updated according to the chosen action, observed reward, and the next state.

Deep Q-Network (DQN) Model

The DQN model consists of a neural network used to approximate the Q-function. The model is implemented using PyTorch and includes the following components: The neural network architecture is defined as a PyTorch class named Network, inheriting from nn.Module. The architecture consists of the following layers:

- Input Layer: Flattens the observation space of the environment and connects to the first hidden layer.
- Hidden Layer: A dense layer with 64 units, followed by a hyperbolic tangent activation function.
- Output Layer: A dense layer that outputs a value for each possible action in the environment's action space.

The hyperparameters marked in the table 1 are used in the DQN model.

Stable Baseline 3 (SB3) Model

Before implementing and training the model using the Stable Baselines 3 library, it was essential to ensure that our custom environment adhered to the necessary interface requirements. This guarantees that the environment can correctly interact with the chosen algorithm (in this case, Deep Q-Network) provided by the library. The following steps detail the procedure carried out to validate the environment's compatibility, declare the model, train it, and finally test it to evaluate performance. The model was then trained for a total of 200,000 timesteps, with logging information printed at every fourth interval, using the command model.learn(total_timesteps=200_000, log_interval=4).

RESULTS AND DISCUSSION

This section provides a detailed analysis of the models implemented and evaluated in this study. Various algorithms, including Random Scheduler, Round Robin, Q-learning, Deep Q-Network (DQN), and Stable Baseline 3 Model, are assessed based on their performance, time latency, and other key metrics.

1. **Random Scheduler**: The Random Scheduler served as a baseline scheduler for comparison with more advanced algorithms. The results were obtained by executing the scheduler for 20 episodes, and the average reward score result was 285.047.
   The time latency in timesteps was plotted, and the average, maximum, and minimum latencies for an episode were computed and analyzed which can be seen in figure 3, providing insights into the performance of the Random Scheduler in terms of time efficiency.

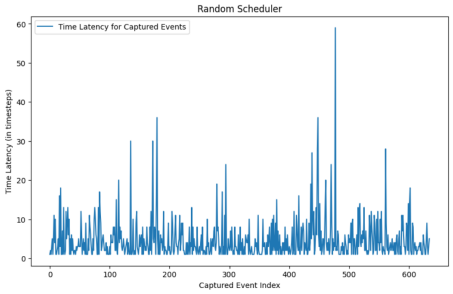

FIGURE 3. Time latency for the Random Scheduler

2. **Round Robin Scheduler**: The Round Robin scheduler was next examined to observe its efficacy in comparison to the random scheduler. In this approach, the scheduling is done in a systematic and periodic manner, looping through the sensors to ensure a fair allocation of events. The experimental results for the Round Robin scheduler over 20 episodes are presented below. A notable improvement was observed in the reward and timelatency compared to the random scheduler. Figure 4, illustrates the time latency plot for the Round Robin scheduler.

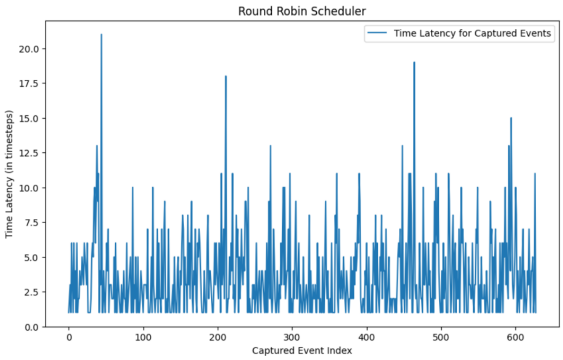

FIGURE 4. Time latency for the Round Robin Scheduler

3. **Q-learning Model**: The Q-learning model[13] was tested for performance, and the results were indicative of certain limitations in handling the complexity of the Wireless Sensor Network environment. The average score reward for 20 episodes was measured at 214.78, a value that, although promising, reveals certain challenges in optimizing the scheduling patterns. Figure 5, illustrates the time latency plot for the Q-learning scheduler.

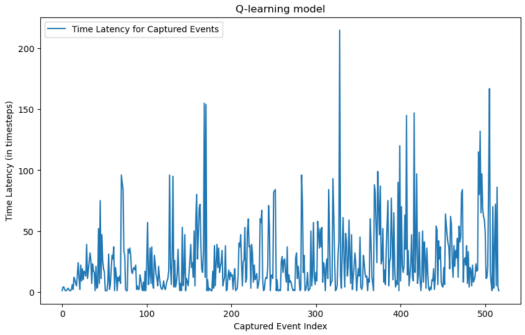

FIGURE 5. Time latency for the Q-Learning Model

4. **Deep Q-Network (DQN) Model**: The DQN model[14] represents a substantial part of this study, and its results are presented in this section. Training for 170 episodes with 1000 timesteps each, the model's performance

was carefully analyzed. The evolution of the reward function was monitored throughout the training. Figure 6, depicts this progression, showing fluctuations before converging to an approximate value of 290. These fluctuations signify the exploration phase, which eventually stabilizes after nearly 120 episodes.

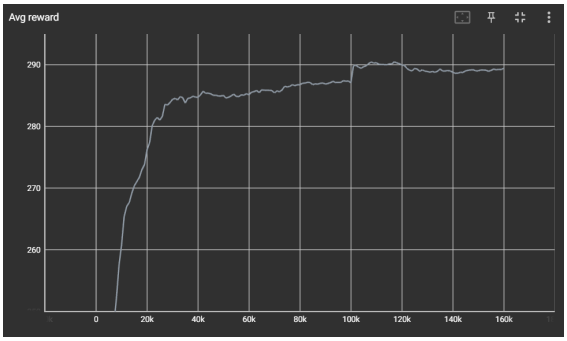

FIGURE 6. Progress of Reward Function in DQN Model

From these results, it can observe that the DQN model performed better than the Random Scheduler in terms of the maximum latency. However, the Round Robin Scheduler still maintains a lead with a lower average and maximum time latency. This comparison illustrates the trade-offs and relative benefits of different scheduling techniques. Figure 7, provides a graphical representation of the time latency for the DQN model.

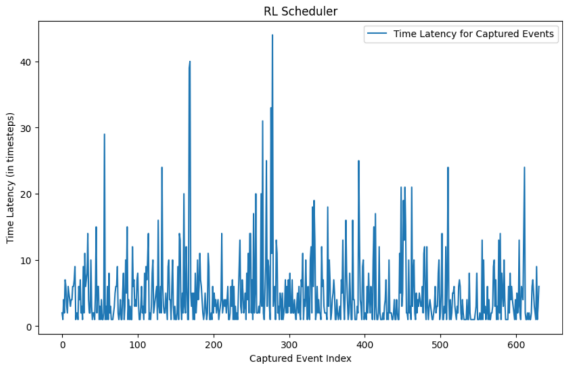

FIGURE 7. Time Latency for the trained agent

5. **Stable Baseline 3 Model**: The SB3 model was employed to explore and understand the complex task of sensor selection and event capturing in the given context. As a more advanced and refined model, SB3 was anticipated to demonstrate an intelligent balance in selecting sensors and optimizing rewards. The following subsections detail the training process, testing outcomes, and specific behavioural patterns observed during the simulation, providing comprehensive insights into the model's performance and potential areas for improvement.
   Most events have time latency between 1 and 20, with two distinct clusters around 90 and between 100 to 130. Comparatively, the Stable Baselines 3 model exhibited a higher average time latency than previous models, such as the Random Scheduler, Round Robin, and the DQN model. The issue of long latency and high number of non-captured events indicates potential areas for further refinement.

6. **Comparison between different schedulers**: The following table resumes the different values and results for the different algorithms tested can be seen in table 2. Figure 8 presents a plot comparing the time latency of three selected algorithms—Random Scheduler, Round Robin, and DQN. This plot captures only the first 100 values for each algorithm to offer better visibility and focus on the initial behavior.

   The DQN model emerges as a robust and versatile solution for WSN event scheduling. It was built upon the foundational principles of Q-learning to offer a scalable, adaptive, and efficient approach. The exploration of SB3 hints at the future direction of research, emphasizing continuous innovation and refinement in the pursuit

TABLE 2. Performance Metrics of Different Scheduling Algorithms

Algorithm	Avg Reward	Avg Time Latency	Min Latency	Max Latency
Random Scheduler	285.047	4.67	1.00	59.00
Round Robin	308.34	3.62	1.00	21.00
Q-learning	214.78	25.54	1.00	215.00
DQN	295.28	4.85	1.00	44.00
SB3	305.27	7.49	1.00	133.00

of optimal scheduling solutions. In summary the analysis presented here sets the stage for future work, opening new avenues for research and development in the realm of scheduling algorithms.

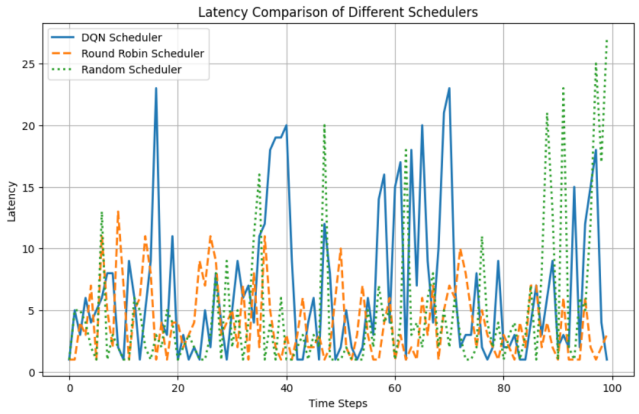

FIGURE 8. Comparison of Time Latency.

7. **State-of-art comparison**: Current work by Xun Wang,et. al[15] illustrates MDP used for determining state, action and reward and Q-learning is used to make the optimized policy for better decision-making and better rewards. Uses Distributed methods that often use stochastic decisions, which can be inefficient when nodes have varying energy levels, causing some nodes to run out of energy too soon. Traditional methods like Q-Learning in managing wake-up radio systems are effective to some extent, they cannot often adapt to dynamic network conditions.
   Improving upon this traditional Q-Learning approach, DQN model is used to make the model more adaptive to dynamic conditions that take into account not just the overall state of the system, but also factors such as battery states, operational demands, and the energy levels at each node. Also, added noise into the signals to reduce False wakeups, which occur when the secondary radio misinterprets noise or irrelevant signals as a wake-up command.

CONCLUSION AND FUTURE SCOPE

The research delved into optimizing energy consumption in IoT systems through a comprehensive framework integrating Wake-Up Radio Systems, Intelligent Scheduling, and Reinforcement Learning, particularly Deep Q-Networks (DQN). It utilized a custom simulation environment developed with the Gym API and validated with Stable Baselines3, including comparisons with a predefined DQN model. The results demonstrated significant improvements in energy efficiency, offering practical benefits for industrial and home automation. Additionally, the study advanced methodological approaches by showcasing how Q-Learning can tackle complex real-world problems, contributing to the literature on applying Machine Learning for better energy management and adaptable scheduling in IoT scenarios.

Future research can explore diverse algorithmic strategies, real-world implementations, and adaptations to various environmental conditions and IoT systems. Building on the current study's robustness and scalability, these efforts promise significant theoretical and practical advancements. The developed framework establishes a strong foundation for future exploration, offering a promising path for both academic and practical innovations.

REFERENCES

1. Using deep reinforcement learning to train and periodically re-train a data-collecting drone based on real-life measurements,https://doi.org/10.1016/j.jnca.2023.103789
2. Sinde, R.; Begum, F.; Njau, K.; Kaijage, S. Refining Network Lifetime of Wireless Sensor Network Using Energy-Efficient Clustering and DRL-Based Sleep Scheduling. Sensors 2020, 20, 1540. https://doi.org/10.3390/s20051540
3. An Adaptive Asynchronous Wake-Up Scheme for Underwater Acoustic Sensor Networks Using Deep Reinforcement Learning,10.1109/TVT.2021.3055065
4. RLS2: An energy efficient reinforcement learning- based sleep scheduling for energy harvesting WBANs,https://doi.org/10.1016/j.comnet.2023.109781
5. Bello, H.; Xiaoping, Z.; Nordin, R.; Xin, J. Advances and Opportunities in Passive Wake-Up Radios with Wireless Energy Harvesting for the Internet of Things Applications. Sensors 2019, 19, 3078. https://doi.org/10.3390/s19143078
6. Yi Chu, Selahattin Kosunalp, Paul D. Mitchell, David Grace, and Tim Clarke. 2015. Application of reinforcement learning to medium access control for wireless sensor networks. Eng. Appl. Artif. Intell. 46, PA (November 2015), 23–32. https://doi.org/10.1016/j.engappai.2015.08.004
7. An Improved Wake-Up Receiver Based on the Optimization of Low-Frequency Pattern Matchers,doi: 10.3390/s23198188
8. A deep reinforcement learning for energy efficient resource allocation Intelligent Reflecting Surface (IRS) driven Non-Orthogonal Multiple Access Beamforming (NOMA-BF),https://doi.org/10.1016/j.phycom.2023.102148
9. Deep Reinforcement Learning for Power Control in Next-Generation WiFi Network Systems, https://doi.org/10.48550/arXiv.2211.01107
10. A Safe Deep Reinforcement Learning Approach for Energy Efficient Federated Learning in Wireless Communication Networks,https://doi.org/10.1109/TGCN.2024.3372695
11. Reinforcement Learning Meets Wireless Networks: A Layering Perspective,DOI:10.1109/JIOT.2020.3025365
12. Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-baselines3: reliable reinforcement learning implementations. J. Mach. Learn. Res. 22, 1, Article 268 (January 2021), 8 pages.
13. A study on a Q-Learning algorithm application to a manufacturing assembly problem, https://doi.org/10.1016/j.jmsy.2021.02.014
14. Deep Q Network (DQN), Double DQN, and Dueling DQN: A Step Towards General Artificial Intelligence,DOI:10.1007/978-981-13-8285-7_8
15. Wang, X., Chen, H. & Li, S. A reinforcement learning-based sleep scheduling algorithm for compressive data gathering in wireless sensor networks. J Wireless Com Network 2023, 28 (2023). https://doi.org/10.1186/s13638-023-02237-4