August 2021
Master's Degree Thesis

A Study on Deep Reinforcement Learning for Energy Conservation in IoT Tracking Applications

Graduate School of Chosun University

Department of Information and Communication Engineering

Sultan Salman Md

A Study on Deep Reinforcement Learning for Energy Conservation in IoT Tracking Applications

IoT 트래킹 응용에서 에너지 절약을 위한
강화학습에 관한 연구

August 27, 2021

Graduate School of Chosun University

Department of Information and Communication Engineering

Sultan Salman Md

A Study on Deep Reinforcement Learning for Energy Conservation in IoT Tracking Applications

Advisor: Prof. Jae-Young Pyun

A thesis submitted in partial fulfillment of the requirements for a Master's degree

April, 2021

Graduate School of Chosun University

Department of Information and Communication Engineering

Sultan Salman Md

술탄 살만 엠디
석사학위논문을 인준함

위원장 조선대학교 권구락 교수 (인)
위　원 조선대학교 변재영 교수 (인)
위　원 조선대학교 최동유 교수 (인)

2021 년 05 월

조선대학교 대학원

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

A Study on Deep Reinforcement Learning for Energy Conservation in IoT Tracking Applications

Salman Md Sultan
Advisor: Prof. Jae-Young Pyun
Dept. Information and Communication Engineering
Graduate School of Chosun University

The Internet of Things (IoT) based target tracking system is required for applications such as smart farm, smart factory and smart city where many sensor devices are jointly connected to collect the moving target positions. Each sensor device continuously runs on battery-operated power, consuming energy while perceiving target information in a particular environment. To reduce sensor device energy consumption in real-time IoT tracking applications, many traditional methods such as clustering, information-driven, and other approaches have previously been utilized to select the best sensor. However, applying machine learning methods, particularly deep reinforcement learning (Deep RL), to address the problem of sensor selection in tracking applications is quite demanding because of the limited sensor node battery lifetime. In this study, the proposed system utilized a long short-term memory deep Q-network (DQN) based Deep RL target tracking model to overcome the problem of energy consumption in IoT target applications. The simulation results show favorable features in terms of the best sensor selection and energy consumption.

한 글 요 약

IoT 트래킹 응용에서 에너지 절약을 위한 강화학습에 관한 연구

술탄 살만 엠디
지도 교수: 변재영
정보통신공학과
조선대학교 대학원

사물 인터넷 (IoT) 기반 표적 추적 시스템은 움직이는 표적 위치를 수집하기 위해 많은 센서 디바이스가 공동으로 연결된 스마트 팜, 스마트 팩토리, 스마트 시티와 같은 애플리케이션에서 필요하다. 각 센서 디바이스는 배터리로 작동하되 지속적으로 실행되며, 디바이스가 위치한 환경에서 목표 정보를 인식하는 과정으로 에너지를 소비한다. 기존의 실시간 IoT 추적 애플리케이션에서는 센서 디바이스의 에너지 소비를 줄이기 위해 클러스터링, 정보 기반 및 기타 접근 방식과 같은 많은 방법을 사용하여 최상의 센서 디바이스를 선택했다. 그러나 추적 애플래케이션에서 센서 선택 문제를 해결하기 위해 기계 학습 방법, 특히 심층 강화 학습 (Deep RL)을 적용하는 것은 센서 노드의 배터리 수명 제한으로 매우 까다롭다. 본 연구에서 제안된 시스템은 IoT 타켓 애플리케이션에서는 에너지 소비 문제를 극복하기 위해 Deep Q-Network 기반 심층 강화 학습 모델을 활용하였다. 시뮬레이션 결과는 최상의 센서 선택 및 에너지 소비 측면에서 유리한 기능을 보여주고 있다.

I. INTRODUCTION

In a 5G sensor network, a massive amount of data is handled via sensor devices in a large area. International Data Corporation (IDC) research states that 70% of companies will drive to use 1.2 billion devices for the connectivity management solution by 5G services worldwide [1]. The Internet of Things (IoT) is the future of massive connectivity under 5G sensor networks. Currently, the IoT is performing a vital role in collecting a large amount of data via numerous sensors in real-time applications [2]. Kevin Ashton initially coined the IoT concept in 1999 [1], [3]. Sensor-based IoT devices can provide various types of services, such as health, traffic congestion control, robotics, and data analysis, which play a significant role in daily life assistance [4], [5]. Target tracking is another critical area where the sensors can be utilized to collect the target real-time position and report it to a server with its relevant information. The practice of tracking one or multiple targets has vast applications in different research areas, such as detecting gas leakage, border monitoring to prevent illegal crossing, or battlefield surveillance.

A. Motivation

In target-tracking scenarios, tracking single or multiple targets can be realized using one or more sensors. However, it is impractical to utilize a single sensor for collecting the target position information owing to an extended area and will take increased time with high energy consumption. Therefore, it is pertinent to use multiple sensors, particularly in tracking applications. Energy consumption in sensor applications is a key task because of the sensor battery lifetime [6], [7].

With these issues, it is essential to efficiently reduce energy consumption because energy conservation leads to an increased battery lifespan. There are various energy consumption reduction methods used in recent years (e.g.,

information-driven clustering and support vector machine approaches) [8], [9]. However, these methods require substantial time, computational complexity, and resources for large-scale practical application.

Reinforcement learning (RL) is a machine-learning subfield that solves a problem without any predefined model. The use of RL comprises two main elements: action and reward. In any dynamic environment, a precisely selected action will provide the best reward. The RL system is exceptionally effective with minimal resources by offering a better solution in real-time decision-based applications. Thus, providing the best outcome, based on current observations after acquiring a good reward in a real-time environment [10]. Deep reinforcement learning (Deep RL) is an extended version of the conventional RL algorithm (i.e., tabular Q-learning). Deep RL is embedded with a deep neural network as a Q-approximator, which can reduce the overall system computational complexity and energy consumption [11]. Moreover, the dense and long short-term memory (LSTM)-based Q-approximators are frequently utilized to increase energy efficiency in time-series environments [12], [13]. Note that the LSTM Q-approximator is more suitable than the dense Q-approximator because of long-term dependencies in a real-time environment [14]

B. Contributions

In this study, a novel Deep RL method based on LSTM deep Q-network (LSTM-DQN) was introduced to solve the energy constraints of IoT tracking applications.

1. The study utilizes LSTM as a Q-approximator to determine the best sensor as an action from a discrete action space while tracking the target. The best sensor is defined by the minimum distance function, which leads to lower energy consumption.

2. The different action selection strategies (i.e., epsilon-greedy and softmax) also study, which are used in target tracking applications [15].

3. However, the epsilon-greedy method has faster improvement and convergence ability than the softmax method in the proposed action space.

4. Therefore, the study proposes an LSTM-DQN-epsilon-greedy method and compare it with LSTM-DQN-softmax, Dense-DQN-epsilon-greedy, and Dense-DQN-softmax approaches in terms of average cumulative rewards, loss convergence, average sensor selection accuracy, and average cumulative energy consumption.

C. Thesis Layout

The remainder of this paper is organized as follows. A description of the background related works is provided in Section II and III, respectively. Sections IV and V show the proposed LSTM-DQN-epsilon-greedy algorithm and numerical results, respectively, for a detailed comparison. Finally, the conclusion summarizes future research work.

II. Background

A. Reinforcement Learning (RL)

The RL agent is used as a decision-maker to take the best action (a_t) from the set of possible actions over the current state (s_t). The RL agent does not learn with the labeled training dataset, but learns from its experience with environmental interaction. During environmental interaction at a particular time, the agent receives an immediate reward (r_t) and jumps to the next state (s_{t+1}). The entire process continues until the agent reaches the final state and begins a new episode after resetting the environment. Figure 1 represents the basic architecture of reinforcement learning.

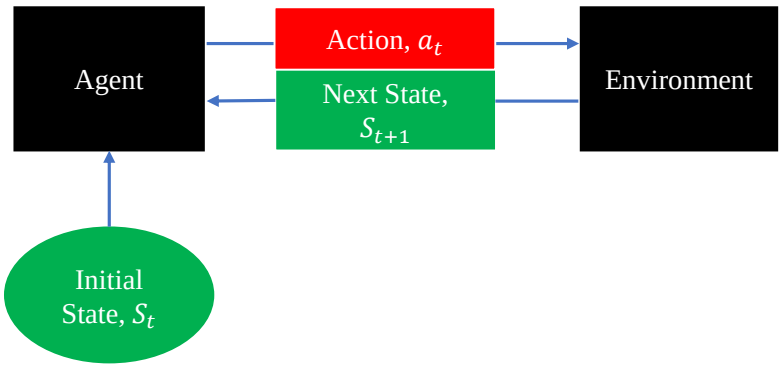

Figure 1: Basic reinforcement learning architecture.

B. Tabular Q-Learning

Tabular Q-learning (TQL) is a common model-free RL approach that is considered an off-policy algorithm because the Q-function learns from the interactive environment by taking random actions during exploration time [16].

Taking action with the help of exploration is essential because initially, the agent has no idea about the new state in an environment; therefore, the agent

needs to explore the environment. After acquiring environmental experience by exploration, the agent can easily exploit the environment by utilizing the greedy strategy. The exploration and exploitation technique is also called the epsilon-greedy technique [10]. Because the TQL is a value-based method, the agent learning policy is utilized through the value function (Q-value) of state-action pairs. In TQL, the Q-value $Q(s_t,\ a_t)$ of an individual action of a particular state is stored in a matrix called the Q-table, which is updated in each time step in (1),

$$Q(s_t,\ a_t) = Q(s_{t-1},\ a_{t-1}) + \partial\,(r_t + \gamma \max(Q(s_{t+1}, a_{t+1})) - Q(s_{t-1}, a_{t-1})), \tag{1}$$

where ∂ and $\gamma \in [0, 1]$ represent the learning rate and discount factor, respectively.

However, TQL has difficulty in extending the Q-table to a large environment, as it is only appropriate for a small environment. To extend the method to a large environment it is necessary for an agent to learn the value function with a Q-approximator instead of saving all values into a Q-table.

C. Deep-Q-Network

The DQN was introduced by Mnih et al. in [17] based on the Deep RL method with the help of a deep neural network, which is known as a Q-approximator. The Q-values of different actions are predicted by utilizing the Q-approximator in a particular state. Figure 2 illustrates the basic DQN architecture where the Q-approximator predicts Q-values of different actions after taking state input.

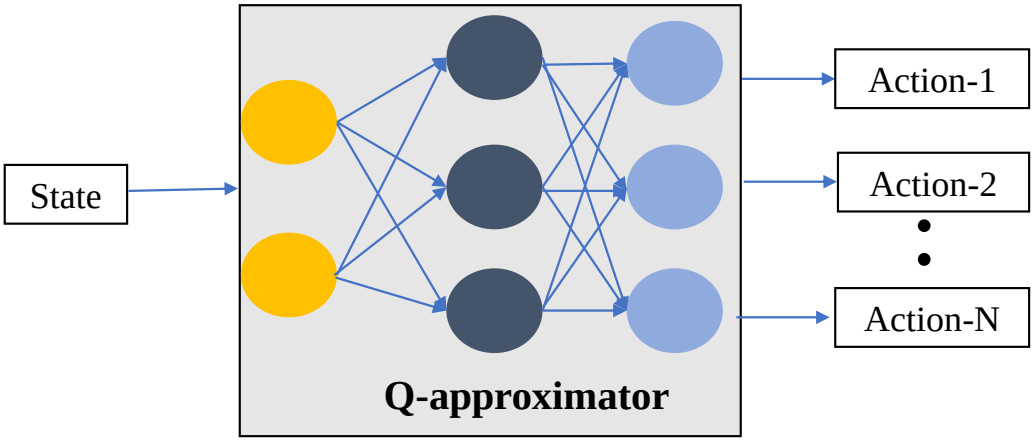

Figure 2: Basic DQN architecture.

In DQN, there is a possibility of a significant correlation between the data, forming the Q-approximator instability during the training period. Because of the extreme correlation between state and actions, it is quite challenging to converge the Q-approximate appropriately. To overcome these difficulties, two techniques: 1) Experience Replay Memory with Mini-Batch and 2) Separate Target Q-approximator are used frequently.

1. Experience Replay Memory

Experience replay memory and mini-batch techniques are utilized to obtain a stable Q-approximator. Experience replay memory (E) stores the experience (s_t, a_t, r_t, s_{t+1}) in each time step to re-utilize previous experiences multiple times. After storing each experience, the DQN uses the mini-batch technique to randomly sample data from the experience replay memory to reduce the correlation between the samples. Figure 3 shows prototype of experience replay memory with mini-batch technique.

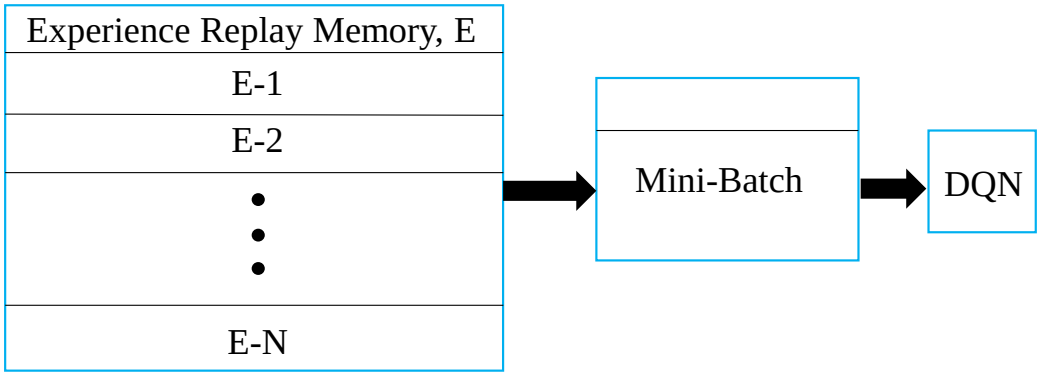

Figure 3: Experience replay memory with mini-batch prototype.

2. Seperate Target Q-approximator

To estimate the predicted and target Q-values with two different Q-approximators θ and θ', to overcome the divergence of the Q-approximator. If the Q-approximator convergences more, the learning performance of the agent will be more improved. The Q-approximator loss $L(\theta)$ is described as,

$$L(\theta) = (r_t + \gamma \max(Q(s_{t+1}, a_{t+1}; \theta')) - Q(s_t, a_t; \theta))^2. \quad (2)$$

D. Long Short-Term Memory (LSTM)

In the proposed system, the LSTM utilizes as a Q-approximator to select the best sensor. The LSTM is a specific type of recurrent neural network (RNN) with the ability to learn long-term dependencies that can memorize and connect related patterns over a time-series input [14]. The LSTM consists of four gates: forget (F_{st}), input (X_{st}), cell (C_{st}), and output (O_{st}) states. These four gates store the combined information of the previously hidden (h_{t-1}) and the current input layer (x_t) and apply the "sigmoid" operation to all gates except the cell state that is finally activated by "tanh" operation, as shown in Figure 4.

In the LSTM mechanism, when the forget state output is near 1, it keeps the data and transfers it to multiply with the previous cell state value (C_{t-1}). The input and cell state gates receive the same information as the forget state gate. After separately applying "sigmoid" and "tanh" operations to input and cell state gate, the outputs are multiplied with each other and added to the forget state output multiplying of the previous cell state value for acquiring a new cell state (C_t). Finally, the output of the new cell state and output state gate after the sigmoid operation multiply with each other to obtain the new hidden state (h_t). The reason behind deploying LSTM for the designed system is that it works flawlessly in a dynamic environment because it depends on the gate operation. The gates regulate the information flow and can also decide which information should be stored or removed.

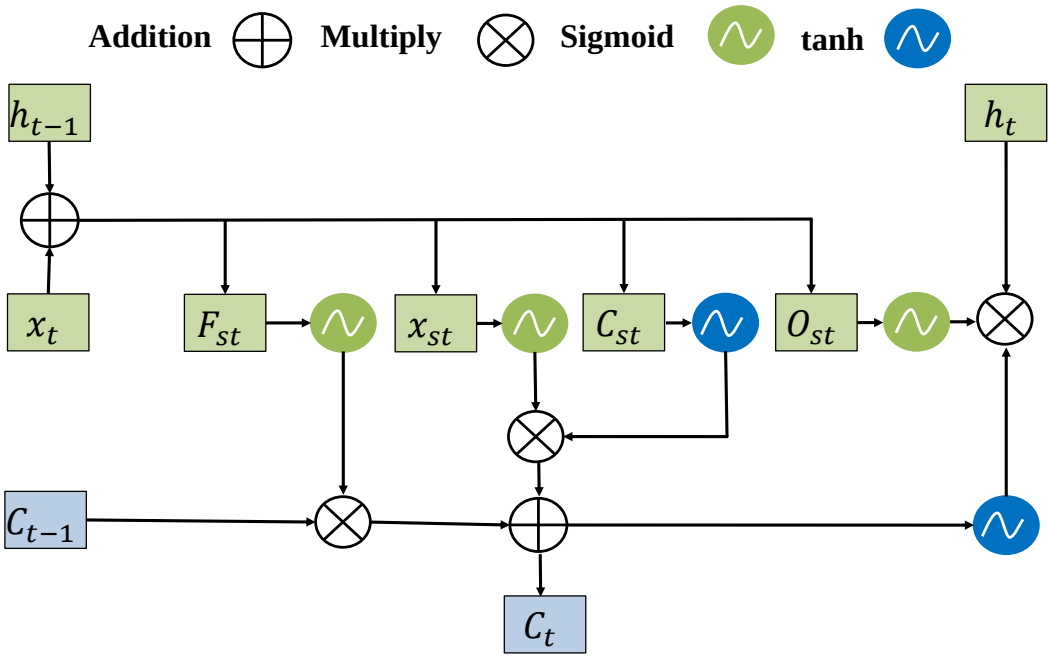

Figure 4: LSTM architecture.

E. Activation Function

The activation function is an essential task while designing the Q-approximator to improve the agent's outcome. The activation function transforms different output signal value ranges into a finite range [18]. There are different types of activation functions available. In this section, the activation functions are described below.

1. Rectifier Linear Unit (ReLU)

The ReLU is used frequently in the deep neural network, particularly in the hidden layer, because of its efficient computation ability. The ReLU is used to obtain the unbounded positive outcome by neglecting any hidden layer's negative weighted sum values during the training.

2. Sigmoid

Generally, the sigmoid activation function works at the output layer of any Q-approximator. The main advatage of this kind of activation function is that it is bounded the value between 0 and 1. Figure 5 shows the output of ReLU and sigmoid activation of function according to different input values.

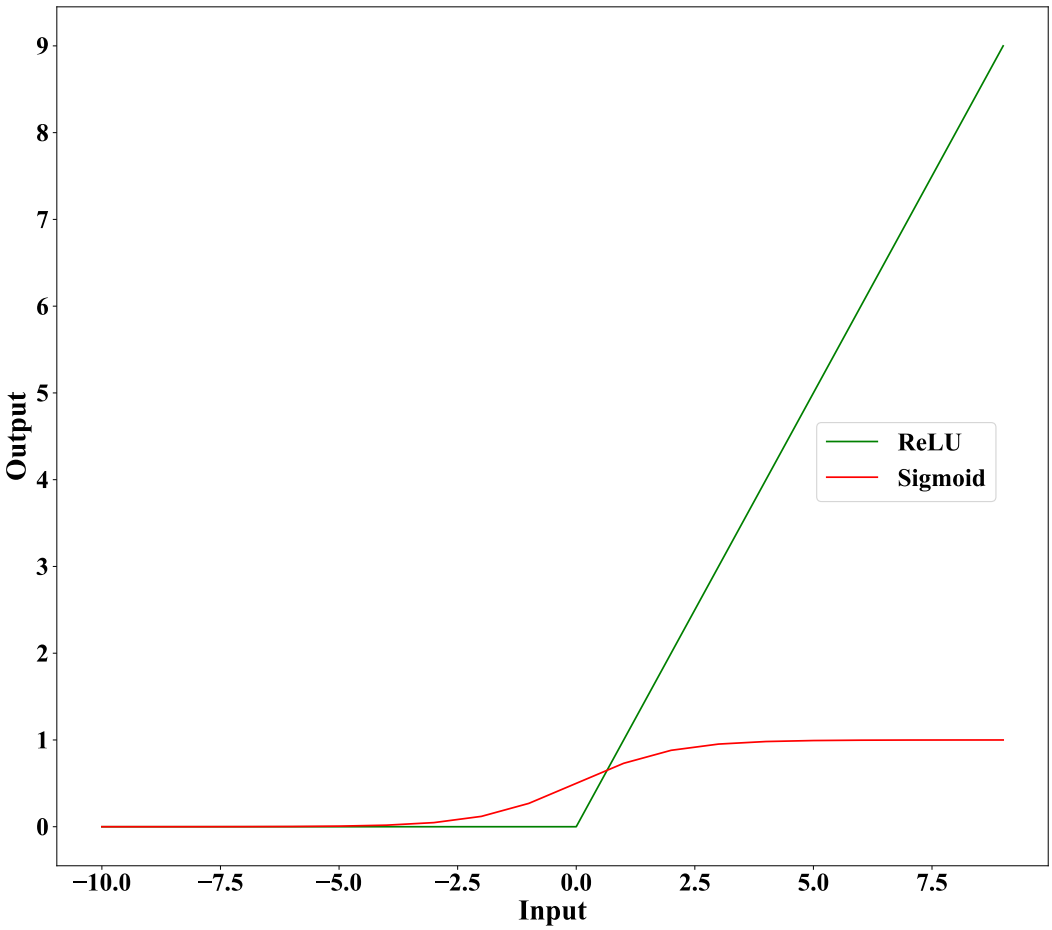

Figure 5: ReLU vs Sigmoid.

F. Kalman Filter

In the proposed system, it is imperative to estimate the distance between the moving target and the sensors to get the best sensor. A sensor with the minimum distance to the target location was selected. For the stated goal, the Kalman filter [19] was used to localize the target position.

The Kalman filter estimates the current system state from a series of noisy measurements, which is useful in tracking applications [19]. The Kalman filter

is a recursive estimator that can compute the target position along with the uncertainty. The system has two significant steps: prediction and updating. Various essential Kalman filter parameters are listed in Table 1.

The initial state matrix α_0 indicates the early stage target observation and consists of four key information pieces such as the x- (x) and y-axis (y) positions, velocity along the x- (vx) and y-axis (vy). In general, the covariance process measures the variation in random variables. The covariance for the four random variables is defined as follows:

$$\sigma(x, y, vx, vy) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})(x_i' - v\bar{x})(y_i' - v\bar{y}), \tag{3}$$

where n is the number of samples, and the covariance matrix is defined as $\sigma(x, y, vx, vy)^T$.

Table 1: Kalman filter parameters.

Symbols	Description
α_0	Initial state matrix
P_0	Initial process covariance matrix
α_{k-1}	Previous state matrix
M_k	Measurement input
G	Kalman gain
Acc_k	Control variable matrix
P_{k-1}	Previous process covariance matrix
$N_{k\alpha}$	Predicted noise matrix
N_{kp}	Process noise matrix
X, Y, Z	Transition matrix
M_e	Measurement error covariance matrix
H, I	Identity matrix

The initial state α_0 and process covariance matrices P_0 are expressed as,

$$\alpha_0 = \begin{bmatrix} x \\ y \\ vx \\ vy \end{bmatrix} \tag{4}$$

$$P_0 = \begin{bmatrix} \sigma^2 x & \sigma_x\sigma_y & \sigma_x\sigma_{vx} & \sigma_x\sigma_{vy} \\ \sigma_y\sigma_x & \sigma^2 y & \sigma_y\sigma_{vx} & \sigma_y\sigma_{vy} \\ \sigma_{vx}\sigma_x & \sigma_{vx}\sigma_y & \sigma^2 vx & \sigma_{vx}\sigma_{vy} \\ \sigma_{vy}\sigma_x & \sigma_{vy}\sigma_y & \sigma_{vy}\sigma_{vx} & \sigma^2 vy \end{bmatrix}. \tag{5}$$

1. Prediction Step

In the Kalman filter, the prediction step estimates the current predicted state α_k and the process error covariance matrix P_k, which are expressed as,

$$\alpha_k = X\alpha_{k-1} + YAcc_k + N_{k\alpha}, \tag{6}$$

$$P_k = X(P_{k-1}X^T) + YAcc_k + N_{kp}\ , \tag{7}$$

where α_{k-1} and P_{k-1} denote the previous state and process error covariance matrices, respectively. The variable X represents the state transition matrix for the previous state α_{k-1}, and Y is the input transition matrix for the control vector. The Acc_k in (8) shows the acceleration of the moving target, given as,

$$YAcc_k = \begin{bmatrix} \frac{1}{2}\Delta T^2 ax & \frac{1}{2}\Delta T^2 ay & \Delta T ax & \Delta T ay \end{bmatrix}^T, \tag{8}$$

where ΔT represents the time for one cycle, while ax and ay are the acceleration control variables.

2. Updated Step

In the updated step, it is time to estimate a new measurement M_k for state prediction at time step k. The Kalman gain, G, shows the prediction accuracy at time step k compared to the input value measurement. The new measurement M_k and gain G are described as follows:

$$M_k = Z - H\alpha_k, \tag{9}$$

$$G = \frac{(P_k H^T)}{H.(P_k H^T) + M_e}, \tag{10}$$

where Z, H, and M_e represent the transition, identity matrix, and measurement error covariance matrix, respectively. After estimating the Kalman gain G, the predicted state α_k and error covariance matrix P_k are updated in (11) and (12), respectively:

$$\alpha_k = X\alpha_k + GM_k, \tag{11}$$

$$P_k = [(I - (GH)) + P_k]. \tag{12}$$

III. Related Works

In recent years, researchers have been working and investing much of their time to solve the problem of excessive energy consumption in tracking-based applications. Below, applications based on the respective techniques from background studies are presented.

A. Tracking Application based on Information-driven Approaches

Information-driven is a collaborative sensing technique for various target tracking applications, where each deployed sensor is responsible for collaborating with other deployed sensors to collect moving target information [20]. Information-driven methods were first proposed in terms of collaborative sensor selection via the information utility function [21]. In this information-driven sensor selection method, the authors considered different Bayesian estimation problems (e.g., entropy and Mahalanobis distance-based utility measurements) to determine which sensor would track the moving target. Wei et al. [22] proposed a dual-sensor control technique based on the information utility function in a multi-target tracking application. In this work, the authors used the posterior distance between sensor and targets (PDST) function to minimize the distance between sensors and targets, which helped the sensors directly drive the targets. Ping et el. in [23] used a partially observed Markov decision process (POMDP) to select sub-optimal sensors for tracking multiple targets. The POMDP sensor selection approach is implemented by maximizing the information gain via a probability hypothesis density (PHD)-based Bayesian framework. Although the techniques proposed in [21]–[23] illustrated good tracking results, there is a limitation in choosing an energy-efficient sensor to make their model work in an intelligent manner to reduce the computational complexity.

B. Machine Learning based Techniques for Tracking Application

Machine learning is an excellent technique to overcome the computational complexity issue in any complicated engineering problem because it is a self-learner, and it does not need to be reprogrammed [24]–[26]. Based on background studies, there are three types of machine learning approaches (i.e., supervised, unsupervised, and reinforcement learning), which have been intelligently utilized for energy optimization. The study of supervised learning techniques is beyond the scope of this research.

1. Unsupervised Learning based Clustering Approaches

To address the energy consumption problem, Hosseini and Mirvaziri in [27] introduced a dynamic K-means clustering-based approach to minimize the target tracking error and energy consumption in wireless sensor networks (WSNs). The proposed technique used a tube-shaped layering method for the sensor nodes to reduce energy dissipation during target tracking. In addition, Tengyue et al. [28] employed a clustering algorithm to control the sensor energy, which detected the target in a real-time mobile sensor network. They used the k-means++ algorithm to separate the sensor nodes into sub-groups. The k-means++ separated the sensor nodes, which carried a higher weighted probability for target detection, and the remaining unnecessary sensors remained in sleep mode to save energy consumption. Juan and Hongwei in [29] proposed another clustering approach to balance energy in terms of multisensory distributed scheduling. Their work used the energy-balance technique to control the activation and deactivation modes of communication modules. They employed a multi-hop coordination strategy to decrease energy consumption. However, these types of unsupervised techniques are time-consuming to address because of the lack of available prior data labeling

[24].

2. Reinforcement Learning Approaches

Sensor scheduling is a promising approach for reducing energy consumption in many tracking applications. Muhidul et al. in [30] proposed a cooperative RL to schedule the task of each node based on the current tracking environment observation. The proposed method helped the deployed sensor nodes cooperate by sharing the adjacent node information during tracking. They applied a weighted reward function that combined both energy consumption and tracking quality matrices to improve the sensor node task scheduling at a particular time. Moreover, transmission scheduling is another necessary task in which Deep RL can be applied. Jiang et al. in [31] proposed an approximation technique for transmitting packets in a scheduling manner for cognitive IoT networks. Their DQN model utilized two parameters (i.e., the power for packet sending via multiple channels and packet dropping) to enhance the system capacity in throughput terms. They used a stacked auto-encoder as a Q-function approximator that mapped the policy to maximize system performance via a utility-based reward technique. However, they exploited the action using a comprehensive index evaluation method in a single relay to sync transmission.

To reduce IoT device energy consumption, Mehdi et al. [32] employed a Deep RL technique to learn an optimal policy for indoor localization problems in IoT-based smart city services. They deployed a semi-supervised technique to classify unlabeled data and integrated classified data with label data. They used iBeacons to provide a received signal strength indicator (RSSI) as an input for a semi-supervised Deep RL model, which consists of a variational autoencoder neural network Q-learning technique to enhance indoor localization performance. In [15], the authors used two Deep RL methods (e.g., DQN and DDPG)

to adjust the activation area radius so the system can minimize the average energy consumption in terms of vehicle-to-infrastructure (V2I) technology-based tracking applications. They also used two action selection strategies (e.g., epsilon-greedy and softmax) to determine the activation area radius.

The Deep RL method has not been widely applied for energy saving in IoT target tracking applications, particularly in energy-efficient sensor selection approaches. Intelligently selecting the appropriate sensor to track the target is challenging because the target position varies over time, creating tracking environment uncertainty. In this case, the DQN-based Deep RL is a sophisticated method because it has the best learning capability when interacting with an uncertain dynamic environment. In DQN, selecting a Q-approximator for the tracking environment is vital for obtaining improved learning performance. Therefore, the LSTM Q-approximator utilized to predict the sub-optimal decisions (i.e., sensor selection) based on sequential information (i.e., target position) with the assistance of different gate operations.

The study is based on a discrete action space, which means that the proposed LSTM Q-approximator selects the most energy-efficient sensor among a finite set of sensors. [15] showed epsilon-greedy and softmax-based action selection methods for the discrete action space. The epsilon-greedy-based sensor-selection technique presented improved efficiency compared to the softmax technique in the simulation results. Thus, the LSTM-DQN method proposed with epsilon-greedy action selection (described as LSTM-DQN-epsilon-greedy in this paper) in a target tracking environment to select the best sensor for maximum energy conservation.

IV. System Overview and Best Sensor Selection

A. System Overview

Figure 6 illustrates the tracking environment where multiple sensor devices represented as $S=\{ S_1, S_2, ..., S_D \}$ are deployed at different positions to observe the moving targets, $T=\{ T_1, T_2, ..., T_L \}$, where L is the number of targets moving in the test area. The area consists of subareas $X=\{X_1, X_2, ..., X_N\}$, where N is the number of subareas.

In this study, the proposed LSTM-DQN-epsilon-greedy scheme allows one sensor to track a single target at time t in a particular area, which eventually leads to tracking T targets in N subareas. For instance, the selected sensors shown in green detect the targets, as shown in Figure 6. The remaining sensors remained unselected to minimize energy consumption.

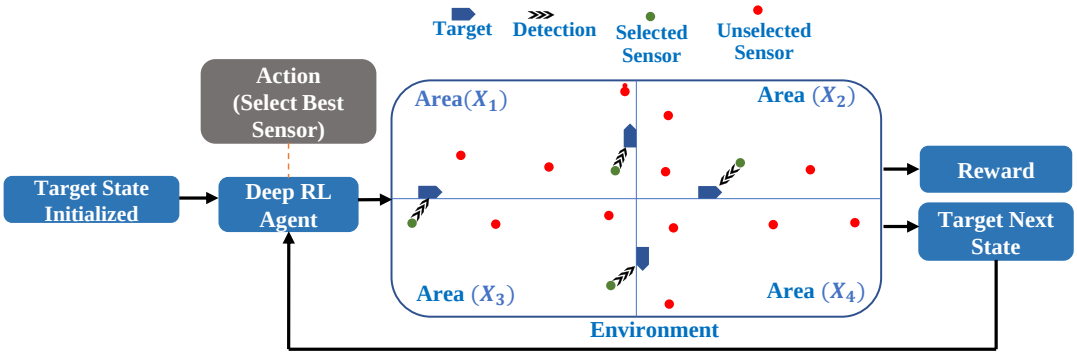

Figure 6: Deployed sensors for tracking target based environment.

For optimal sensor selection, the proposed LSTM-DQN-epsilon-greedy-based IoT tracking system tracks more than one target simultaneously in four subareas X_1, X_2, X_3, and X_4, as shown in Figure 6, thus allowing the system to track all T targets in the first attempt. If a single DQN algorithm applies for all N subareas, there is a possibility of not achieving the required goal because

when the system interacts with a large area, the sensor selection space is more complicated to utilize the algorithm for effective simultaneous tracking more than one target.

B. Best Sensor Selection

The designed LSTM-DQN-epsilon-greedy system uses multiple sensors to track the target position. The system operates in such a manner that it does not allow all sensors to simultaneously track the target. If all the sensors work simultaneously, it leads to high energy consumption. Therefore, the system intelligently adjudicates to select the best sensor at a particular time. The sensor with low energy consumption is considered to be the best sensor and is apportioned to acquire target position information. In the example shown in Figure 6, if the energy consumption of the four sensors (i.e., S_1, S_2, S_3, and S_4) are $6J$, $5J$, $7J$, and $8J$, respectively, then sensor S_2 is selected to track the target.

V. The Proposed LSTM-DQN-epsilon-greedy Method

A. State Space

The proposed LSTM-DQN-epsilon-greedy model acts as an agent that takes the current state as the input. Estimated minimum distance leads to low energy consumption at a specific time. The sensor with the minimum distance and energy consumption is considered to be the best sensor for an individual area. Therefore, the organized state includes individual distances (i.e., $d_{S_1}, d_{S_2}, ..., d_{S_D}$) between the target and sensors. The distance is measured at each time step by using the Euclidean distance formula:

$$d_{S_D}(t) = \sqrt{(P_{target\_xcord} - P_{xcord_{S_D}})^2(t) + (P_{target\_ycord} - P_{ycord_{S_D}})^2(t)}, \quad (13)$$

where $P_{xcord_{S_D}}$, $p_{ycord_{S_D}}$, P_{target_xcord}, and p_{target_ycord} are the positions of all the deployed sensors and the moving target in the two dimensional x-y plane. Furthermore, the position of any target is computed using the Kalman filter.

B. Preprocess State

The state has different distance value ranges, which can create instability for the Q-approximator. Therefore, it is necessary to preprocess the state value by normalization before sending it to the LSTM Q-approximator [33]. The mini-max normalization method is used, which is represented as $state_{normalized}(t) = \frac{(s_t - \min(s_t))}{\max(s_t) - \min(s_t)}$ to scale the state between 0 and 1 to enhance the state quality before sending it to the proposed LSTM Q-approximator.

C. Action Space

The discrete action space ($A = \{A_{S_1}, A_{S_2}, ..., A_{S_D}\}$) represents all the allocated sensors (i.e., S_1, S_2,..., S_D), respectively, in a defined area. The epsilon-greedy is used as an action-selection strategy in the designed system because it is suitable for the discrete action space. In the epsilon-greedy approach, initially, the agent takes a random action to explore the environment through the epsilon method. There are three key parameters: maximum-epsilon (ε_{max}), minimum-epsilon (ε_{min}), and epsilon-decay (ε_{decay}) that are considered to fix the epsilon period. First, it begins with the maximum-epsilon value and then decays with an absolute epsilon-decay value at each time step. The epsilon period is completed when the value of epsilon reaches the minimum-epsilon. Subsequently, the agent greedily exploits the environment to take sub-optimal action with the proposed LSTM Q-approximator, as shown in Figure 7. The rectified linear unit (ReLU) is used in the first three layers, whereas the sigmoid activation function works at the output layer. Moreover, the LSTM Q-approximator predicts the Q-values for all possible actions, which are defined in the action space. Finally, the agent selects the suboptimal action with the highest action-Q value that is obtained by $\arg\max(state_{normalized_t, a_t; \theta})$.

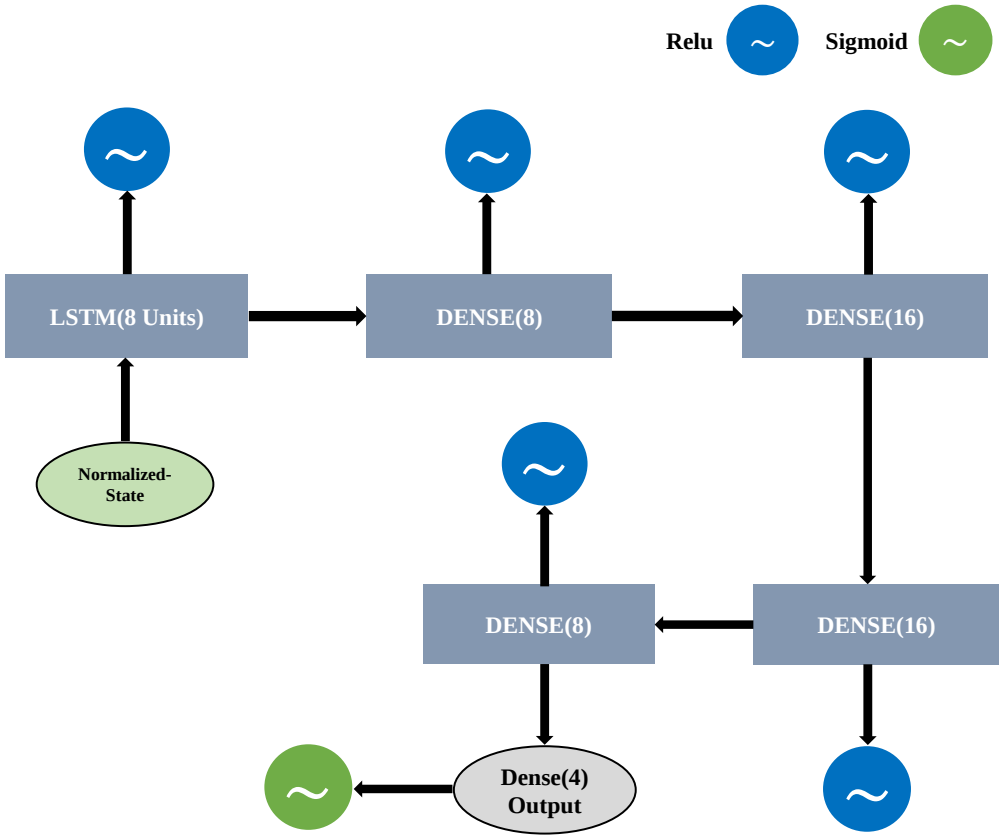

Figure 7: Proposed LSTM Q-approximator.

D. Energy Consumption Model

The proposed agent selects the best sensor as an action that consumes minimum energy during target tracking. The energy consumption ($E_{con_{S_D}}$) of each sensor at time step (t) is estimated using (14), where d_{S_D}, pow_{Sensor}, and t_{track} indicate the distance value between a particular sensor (S_D) and the target, the working mode sensor power and time to track the target in a single area, respectively. Similarly, the energy consumption measured for the other N areas. Note that the energy consumption of all sensors is stored in an array as ($E_{con_{all}}$) in (15).

Furthermore, the selected sensor energy consumption ($E_{con_{act}}$) and minimum energy consumption ($E_{con_{min}}$) are obtained from (16) and (17). Finally, the total energy consumption ($E_{con_{total}}$) and energy savings estimated in a particular observation using (18) and (19), respectively:

$$E_{con_{S_D}}(t) = d_{S_D}(t) \times pow_{sensor}(t) \times t_{track}(t) \tag{14}$$

$$E_{con_{all}}(t) = E_{cons_{D:1\sim D}}(t), \tag{15}$$

$$E_{con_{act}}(t) = E_{con_{all}}[A_{S_D}](t) \tag{16}$$

$$E_{con_{min}}(t) = \min(E_{con_{all}}(t)) \tag{17}$$

$$E_{con_{total}}(t) = \sum_{S_{D=1}}^{D} E_{cons_{S_D}}(t), \tag{18}$$

$$E_{save}(t) = E_{con_{total}}(t) - E_{con_{action}}(t). \tag{19}$$

E. Binary Based Reward Space

The primary goal of the proposed system is to maximize the cumulative rewards after a certain number of steps; therefore, it needs to generate a suitable reward mechanism to improve the agent action. The binary reward function is used in the proposed system design as follows:

$$r_t = \begin{cases} 1 & \text{if } E_{con_{action}} = E_{con_{min}} \\ 0 & \text{if } E_{con_{action}} \neq E_{con_{min}}, \end{cases} \tag{20}$$

where r_t is the reward at time t; further, if the energy $E_{con_{action}}$ is equal to $E_{con_{min}}$, it returns 1; otherwise, the output will be 0. The proposed LSTM-DQN-epsilon-greedy system architecture and algorithm are shown in Figure 8 and Algorithm 1, respectively.

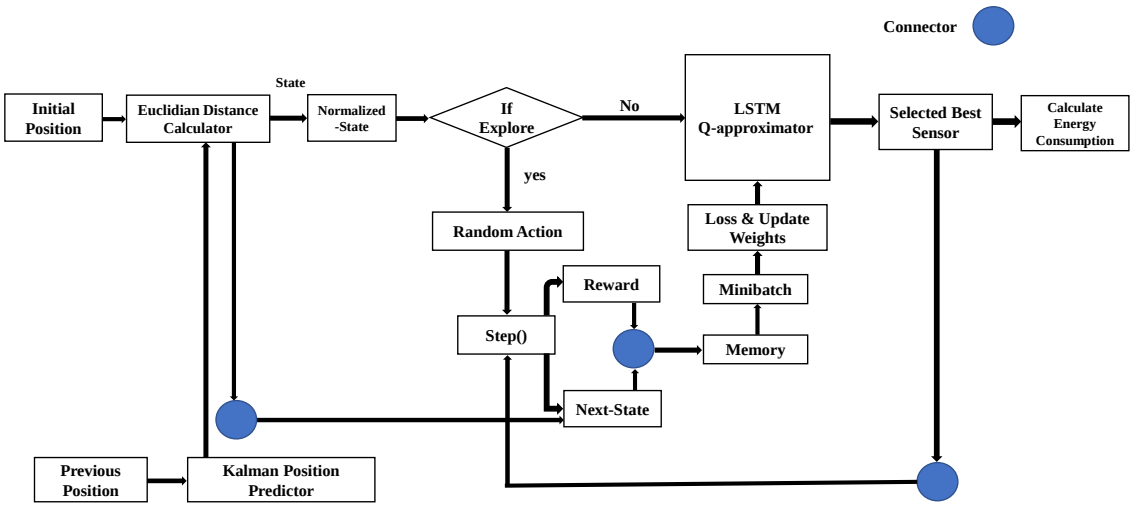

Figure 8: Proposed LSTM-DQN-epsilon-greedy system architecture.

Algorithm 1: The proposed LSTM-DQN-epsilon-greedy algorithm.

Input : Distance between sensor and target position
Output: Best sensor selection accuracy and energy consumption
initialization() ▷ Total number of episodes ep_{total}, Total number of steps $step_{total}$, Environment and Training Hyperparameters
for *(Episode 1 to* ep_{total}*)* **do**
  s_t = reset_environment() ▷ Get the initial state using (13)
  Cumulative rewards, $c_r = 0$
  for *(time-step,* $t = 1$ *to* $step_{total}$*)* **do**
    Preprocess s_t as $state_{normalized_t}$ ▷ Mini-Max normalization
    rand = random.uniform(0,1)
    $\varepsilon = \max(\varepsilon_{min}, \varepsilon)$
    if *(*$rand < \varepsilon$*)* **then**
      take action randomly ▷ Exploration
      $\varepsilon = \varepsilon \times \varepsilon_{decay}$
    else
      action = $\arg\max(state_{normalized_t}, a_t; \theta))$ ▷ Exploitation
    end
    Calculate $E_{con_{S_{D:1\sim D}}}$ and $E_{con_{action}}$ ▷ From (14), (16)
    Calculate $E_{con_{min}}$ ▷ From (17)
    Calculate $E_{con_{total}}$ and E_{save} ▷ From (18) and (19)
    Predict next target kalman state using Kalman Filter
    Calculate s_{t+1} ▷ From (13)
    Normalize s_{t+1} as $state_{normalized_{t+1}}$
    Calculate r_t
    $c_r = c_r + r_t$ ▷ Calculate cumulative rewards
    Store all expereinces into experience replay memory E
    Perform random mini-batch sampling from E

$$target = \begin{cases} r_t & \text{if } r_t = 0 \\ r_t + \gamma \max(Q(state_{normalized_{t+1}}, a_{t+1}; \theta')) & \text{if } r_t = 1 \end{cases}$$

    Perform gradient descent of $(target - Q(s_t, a_t; \theta))^2$ to update θ
    $s_t = s_{t+1}$
  end
end

VI. Simulation and Results

A. Environment Setup and Hyper Parameters

To evaluate our proposed system, a simulation platform with moving target observation of 16 sensor devices is considered with four subareas, where each subarea consists of $200m \times 200m$. Four sensors are allocated in each subarea, and each sensor can cover an area of up to $50m \times 50m$. Thus, 16 sensors cover a total area of $800m \times 800m$. Furthermore, the distance between each sensor was the same in each subarea. one target assumes in a particular subarea and extend it to four targets in four different subareas at a specific time. The environmental details are listed in Table 2.

Table 2: Details of proposed environment.

Parameters	Value
Total number of subareas (N)	4
Size of a subarea (X_N)	$200m \times 200m$
Number of sensors in a subarea (X_N)	4
Total number of sensors in 4 subareas	16
Each sensor tracking range	$50m \times 50m$
Power of sensor in working mode (pow_{sensor})	5 watts
Tracking time of sensor per meter (t_{track})	2 s
Number of target (each subarea)	1
Total number of targets in 4 subareas	4
Targets initial positions	[0, 0]-[200, 200]-[400, 400]-[600, 600]
Target initial velocity	[0.1 m/s, 0.2 m/s]
Target initial acceleration	[5 m/s^2, 5 m/s^2]

During our simulation, the total number of episodes was 500, where each episode consisted of 100-time steps. In each time step, the target positions are updated using the Kalman filter method. Thus, 100 different states utilize for the proposed LSTM-DQN-epsilon-greedy system in one episode. Figure 9 shows a

sample of different state values in one area after applying the normalization (i.e., mini-max normalization, which was described in section V.B) at the time of the experiment. Here, $d1$, $d2$, $d3$, and $d4$ represent the normalized distance values between the four sensors and the target. The normalized state was near zero when the moving target passed near a particular sensor. Conversely, the particular distance values were greater than 0 and gradually increased to 1 when the target moved far behind the sensor. The figure clearly shows that the initial value of $d1$ (i.e., the distance between the first sensor and the target) is zero as the target moves very close to the first sensor. The same is true for the other sensor distance values during the simulation period.

Note that the system restarts each episode when the number of steps reaches 100, and targets again start moving from the initial position. Moreover, some useful hyperparameters were set during the training session, as presented in Table 3. These parameters are used to tune the proposed LSTM-DQN-epsilon-greedy scheme to achieve a more stable output. These hyperparameter values were chosen by a trial and error process. The simulations performed using Python 3.7.7 [34]. TensorFlow 2.0.0 and Keras 2.3.1 were used to implement the LSTM Q-approximator [35], [36]. The source code of our proposed system is given in the Appendix section.

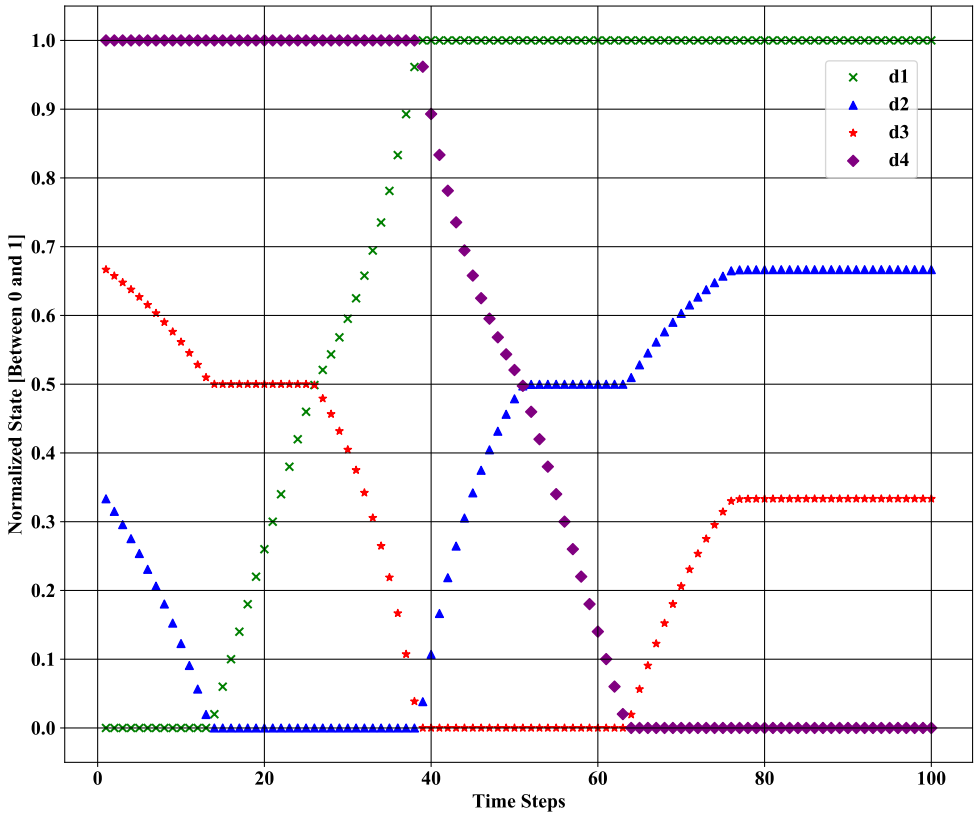

Figure 9: Normalized State value for each time step during the experiment.

Table 3: Hyperparameters for LSTM-DQN-epsilon-greedy during training.

Hyperparameter	**Value**
Optimizer	adam
Loss	categorical crossentropy
Batch Size	16
Size of experience replay memory (E)	50
Learning rate (∂)	0.001
Discount factor (γ)	0.9
Maximum epsilon (ε_{max})	1
Minimum epsilon (ε_{min})	0.01
Epsilon decay (ε_{decay})	0.995

B. Results

1. Cumulative Rewards

In our proposed LSTM-DQN-epsilon-greedy method, the cumulative rewards (c_r) measure in (21) for each episode.

$$c_r = \sum_{t=1}^{101} r_t. \tag{21}$$

The estimation of the cumulative reward is important because it indicates the agent's learning performance during interaction with the target tracking environment. The proposed agent receives a reward of 1 when the agent successfully selects the best sensor, as discussed briefly in Sections V.C and V.E. In Figure 10, the cumulative reward is shown per episode for each subarea. It shows that the cumulative reward is less than 35 for each subarea and does not reach the highest value in the first two episodes (200 steps), as it initially explores the environment. In general, the exploration duration depends on the epsilon parameter values (i.e., ε_{max}, ε_{min}, and ε_{decay}) given in Table 2.

Following the exploration stage, the proposed agent starts exploiting the environment through a greedy approach for selecting the best sensor to track the target. In this case, the agent selects the suboptimal action based on the maximum predicted action-Q value. During the greedy process, the cumulative reward gradually increased after the second episode for all subareas. With the proposed method, the highest cumulative reward of up to 100 was achieved before reaching 100 episodes for all subareas showing outstanding performance while selecting the best sensor.

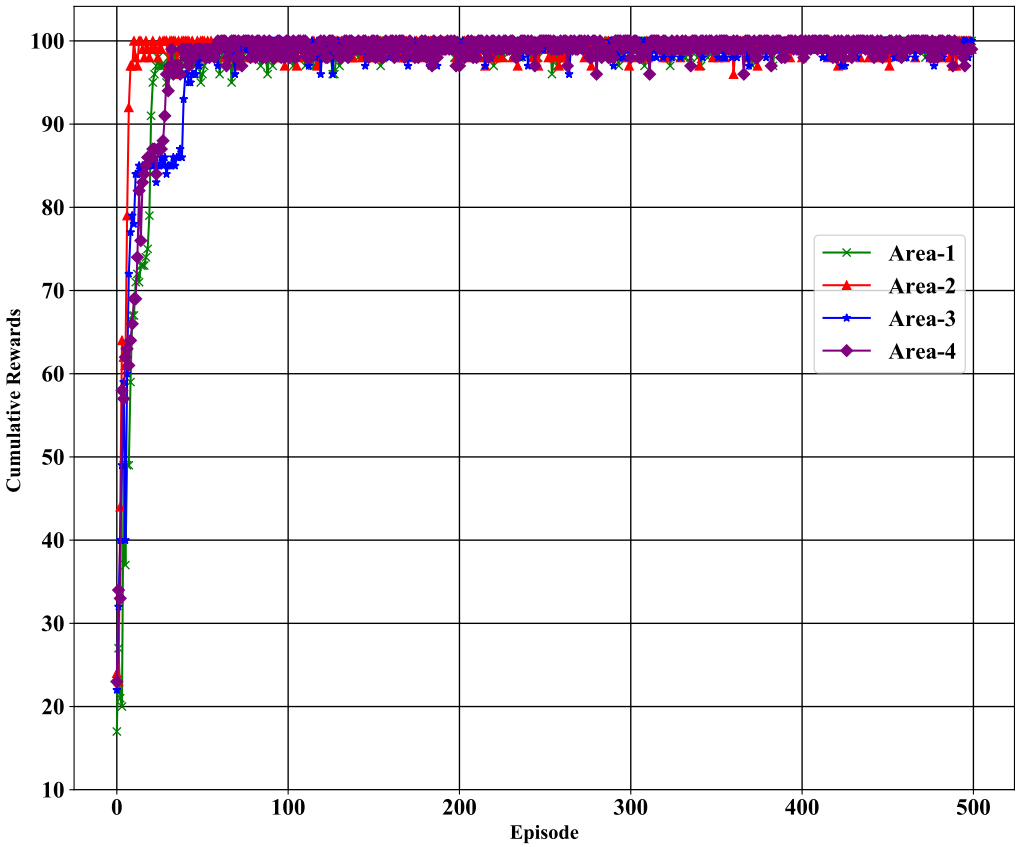

Figure 10: Cumulative rewards for each area.

2. Best Sensor Selection Accuracy

Because sensors have a limited battery lifetime, it is essential to reduce energy consumption as much as possible. In the proposed scheme, the system selects the four best sensors at a particular time within an area of $800m \times 800m$ divided into Areas 1, 2, 3, and 4, as shown in Figure 6. The accuracy of selecting the best sensor affects energy consumption during the tracking target because the best sensor selection is based on the minimum energy consumption described in Section V.D. Figure 11 shows the best sensor selection accuracy for the 16

sensors. This demonstrates that the proposed LSTM-DQN-epsilon-greedy system has a significant accuracy of approximately 99% for Sensors 1, 8, 12, 14, and 16. Similarly, the system achieved an accuracy of 98% for Sensors 4, 5, 6, and 10. Moreover, the proposed system provides more than 90% accuracy in the case of all other sensors, leading to promising results.

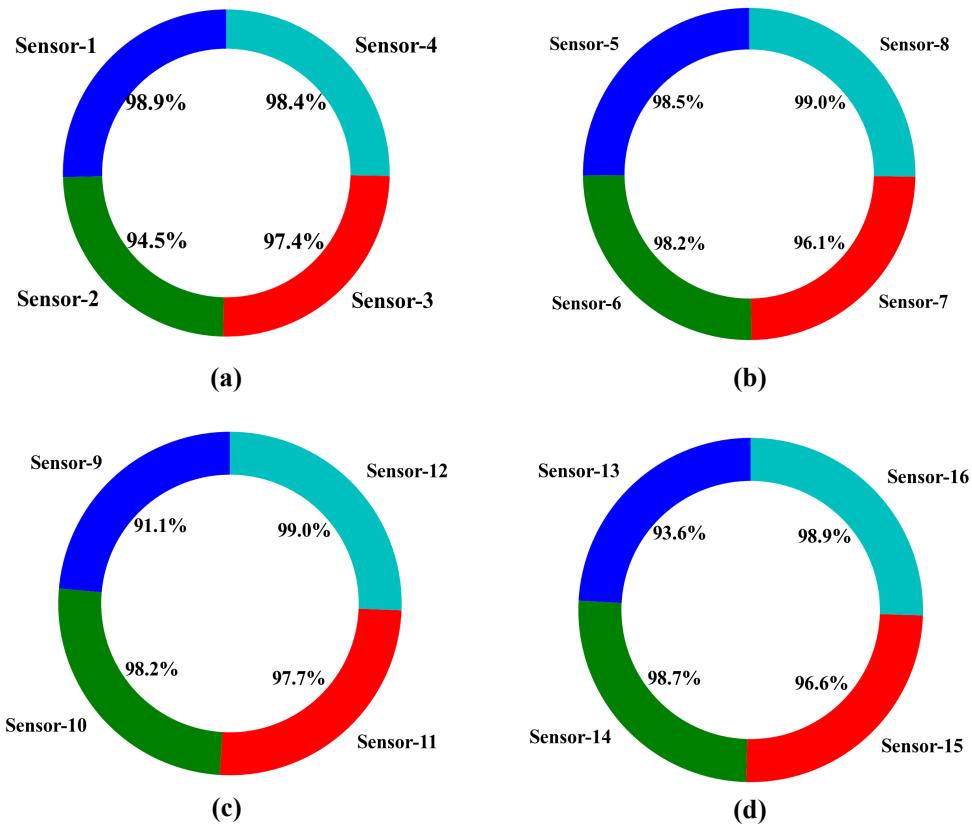

Figure 11: Best sensor selection accuracy: **(a)** Sensor selection accuracy for Area 1; **(b)** Sensor selection accuracy for Area 2; **(c)** Sensor selection accuracy for Area 3; **(d)** Sensor selection accuracy for Area 4.

C. Comparative Analysis

The proposed LSTM-DQN-epsilon-greedy system is also compared with three benchmark schemes: LSTM-DQN-softmax, Dense-DQN-epsilon-greedy, and Dense-DQN-softmax in terms of average cumulative reward, loss convergence, average best sensor selection accuracy, and cumulative energy consumption. In DQN, the LSTM and density-based Q-approximator are used frequently for the dynamic environment. However, LSTM exhibits better performance in handling such an environment because of memory features. The different action-selection strategies (e.g., epsilon-greedy and softmax) also utilized and compared with each other.

Table 4: Overall performance analysis

Methods	Loss convergence	Overall rewards	Overall energy consumption
LSTM-DQN-epsilon-greedy (proposed)	fast and highly stable	97.71	23457.90 J
LSTM-DQN-softmax	fast but highly stable	97.13	23614.98 J
Dense-DQN-epsilon-greedy	slow and highly unstable	93.44	27235.31 J
Dense-DQN-softmax	slow and less stable	88.15	28210.15 J

1. Average Cumulative Reward

The key designed method deployment objective is to increase the average cumulative reward to measure the agent's performance. The average cumulative reward (avg_{c_r}) is obtained in (22), where ep denotes the episode and X_1, X_2, X_3,

and X_4 are four system subareas.

$$avg_{c_r} = \sum_{ep=1}^{501} \frac{c_{r_{X_1}}(ep) + c_{r_{X_2}}(ep) + c_{r_{X_3}}(ep) + c_{r_{X_4}}(ep)}{4}. \tag{22}$$

Figure 12 shows the average cumulative reward per episode for the four DQN-based schemes. The figure shows that our proposed model and the LSTM-DQN-softmax model both achieved the highest average cumulative reward, which was up to 100 during the simulation period. However, LSTM-DQN-epsilon-greedy reached achievedthe highest value faster in 63 episodes compared to the LSTM-DQN-softmax, which reached that level in 115 episodes.

The efficiency of our proposed system is that the epsilon-greedy action selection strategy directly learns from the action-Q-value function, which is suitable for discrete action space. Furthermore, the comparison has been extended to the other two Dense-DQN-based schemes: Dense-DQN-epsilon-greedy and Dense-DQN-softmax. The performance of both LSTM-DQN-based approaches is better than that of Dense-DQN methods because of the long-term memory dependencies described in Section II.D. Therefore, both the Dense-DQN-epsilon-greedy and Dense-DQN-softmax schemes are unable to reach the highest average cumulative reward over the entire 500 episodes, and the average cumulative reward increase of both methods is much slower than the proposed LSTM-DQN-epsilon-greedy scheme.

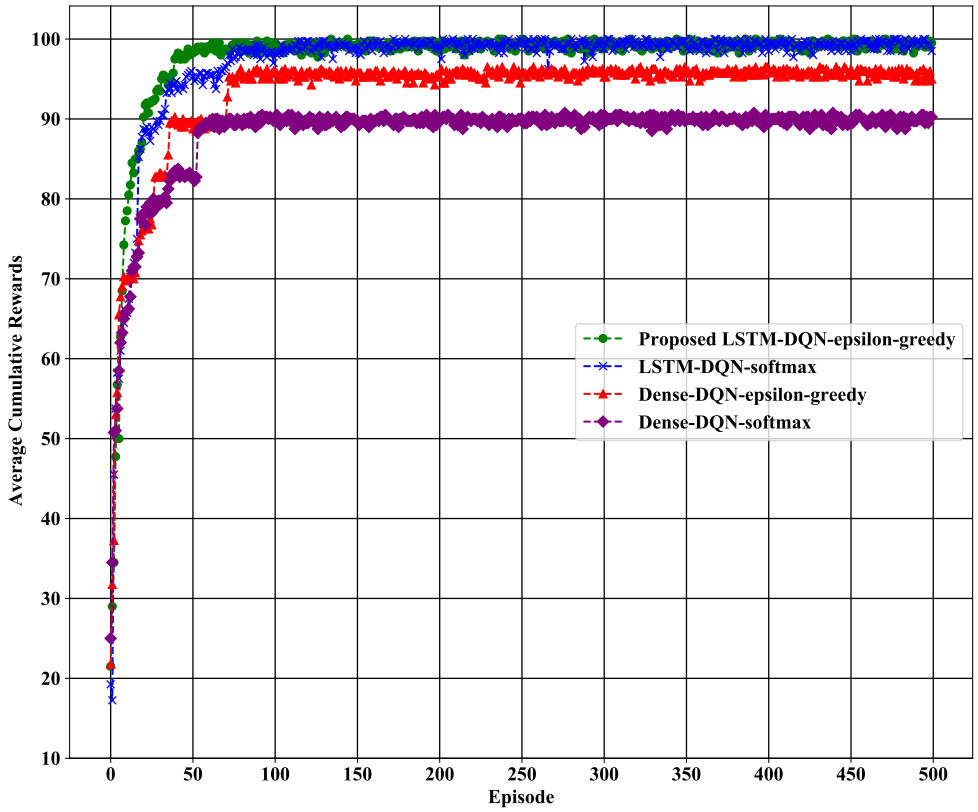

Figure 12: Average cumulative rewards per episode.

2. Loss Convergence

The loss convergence depreciation to the minimum level is also vital, along with the system stability. The proposed LSTM-DQN-epsilon-greedy system signifies good convergence behavior around 200,000 epochs and is more stable, as illustrated in Figure 13. Moreover, the LSTM-DQN-softmax convergence also appeared around 200,000 epochs, but was less stable than our proposed scheme. Furthermore, Dense-DQN-epsilon-greedy and Dense-DQN-softmax methods show unstable behavior and converge at 500,000 epochs, which is time-consuming. Therefore, the proposed LSTM-DQN-epsilon-greedy algorithm is

efficient and stable, leading to promising results.

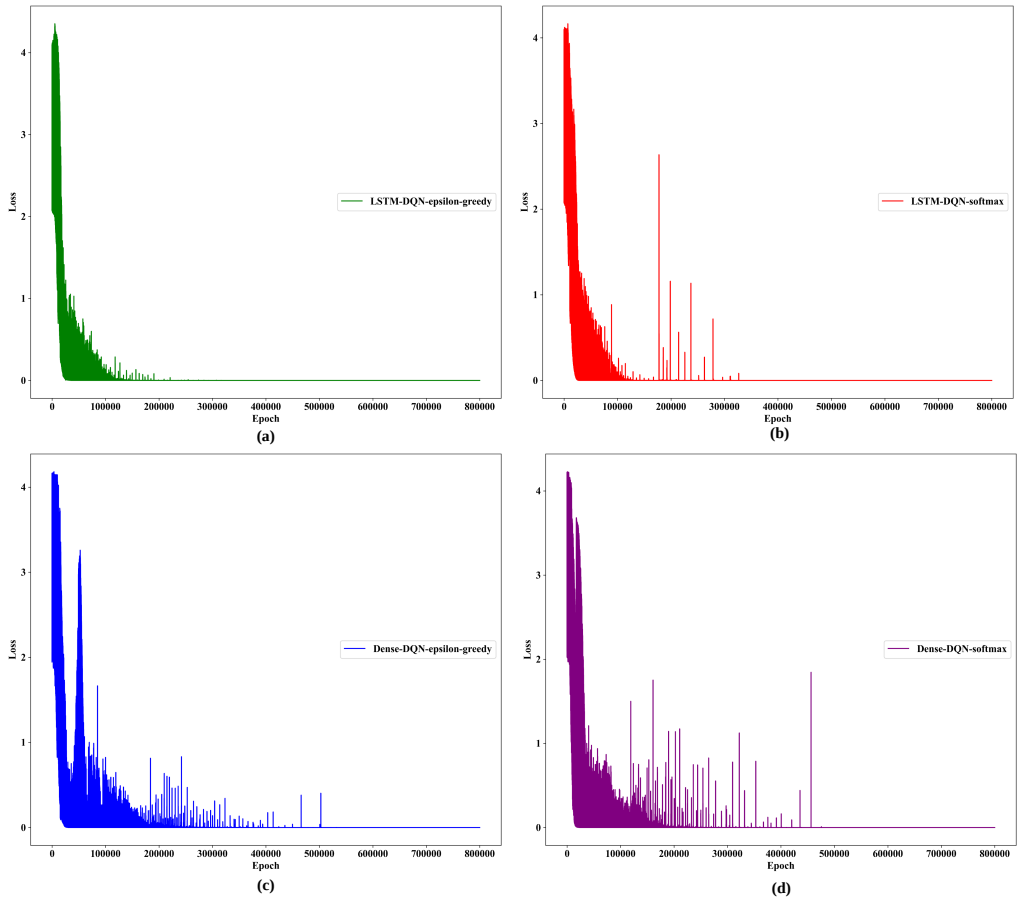

Figure 13: Loss convergence per epoch during training: **(a)** Loss convergence for proposed epsilon-greedy-LSTM-DQN; **(b)** Loss convergence for softmax-LSTM-DQN; **(c)** Loss convergence for epsilon-greedy-Dense-DQN; **(d)** Loss convergence for softmax-Dense-DQN.

3. Average Best Sensor Selection Accuracy

In this section, the comparison of the average best sensor selection accuracy of the proposed system with that of the other three DQN methods, as presented in Figure 14. In our study, the agent selects the best sensor that has minimum

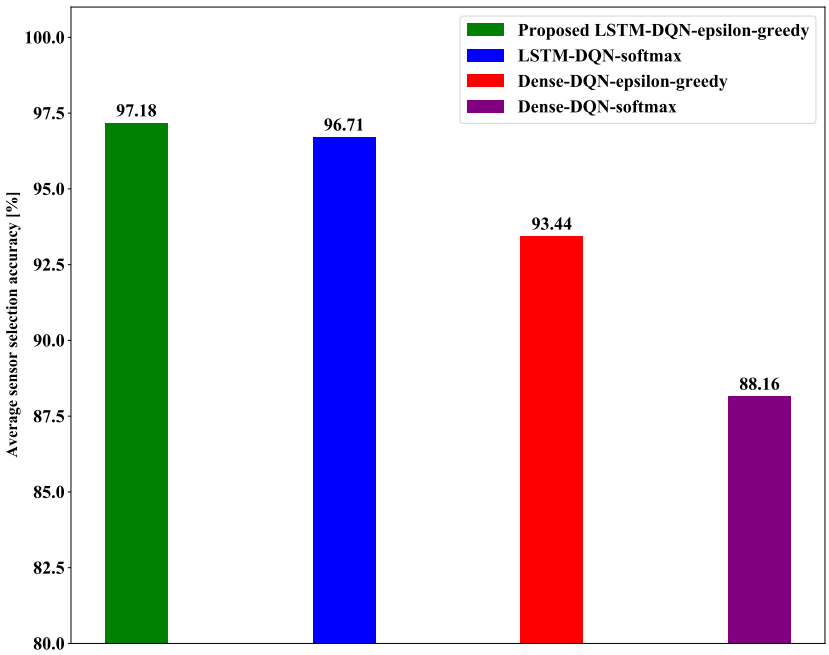

Figure 14: Average best sensor selection accuracy.

energy consumption when the target moves in any particular area. The critical task is to significantly enhance the best sensor selection accuracy to reduce the average energy consumption. As shown in Figure 14, the proposed system agent selects the best sensor with a slightly higher average accuracy than LSTM-DQN-softmax. Furthermore, the proposed LSTM-DQN-epsilon-greedy scheme achieved significantly higher best sensor selection accuracy than the Dense-DQN-epsilon-greedy and Dense-DQN-softmax methods.

4. Average Cumulative Energy Consumption

Our designed system was also utilized to reduce the average cumulative energy consumption while tracking the target. The study has already mentioned in Sections VI.C.1 and VI.C.3, that a higher average cumulative reward effectively

enhances the best sensor selection accuracy and reduces the average cumulative energy consumption. The average cumulative energy consumption ($avg_{E_{con}}$) is obtained using (23).

$$avg_{E_{con}} = \sum_{ep=1}^{501} \frac{E_{con_{act_{X_1}}}(ep) + E_{con_{act_{X_2}}}(ep) + E_{con_{act_{X_3}}}(ep) + E_{con_{act_{X_4}}}(ep)}{4}. \quad (23)$$

Figure 15 shows the average cumulative energy consumption in 500 episodes. It can be observed from the figure that the average cumulative energy consumption for each method is higher, particularly in the first 100 episodes. The reason behind it is that initially, the agent has no experience with the environment. However, as the number of episodes increases, the average cumulative energy consumption decreases significantly for both LSTM-DQN and Dense-DQN based schemes.

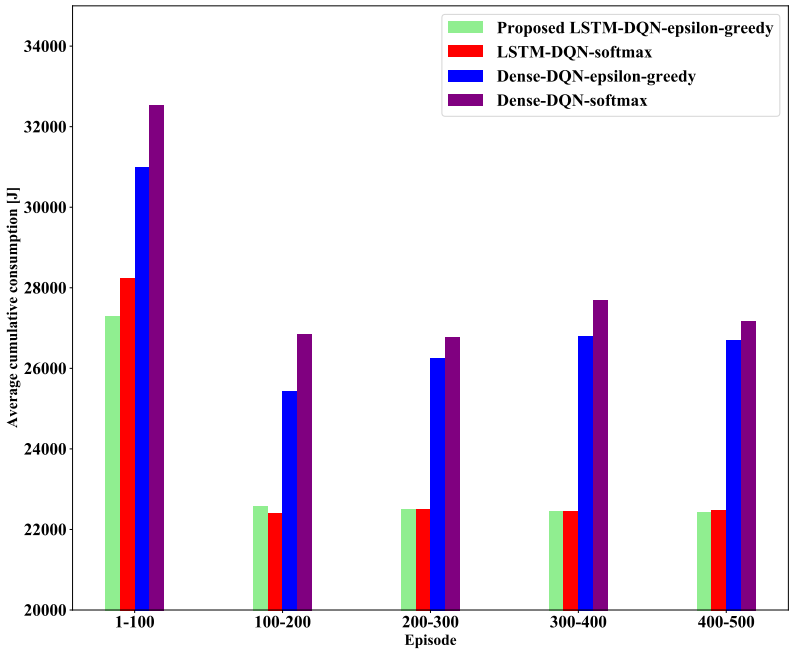

Figure 15: Average cumulative energy consumption.

In contrast, both LSTM-DQN-epsilon-greedy and LSTM-DQN-softmax methods have much lower average cumulative energy consumption compared to Dense-DQN-epsilon-greedy and Dense-DQN-softmax because the LSTM Q-approximator can regulate the information flow in memory in the long and short term. Furthermore, both the LSTM-DQN-epsilon-greedy and LSTM-DQN-softmax schemes approximately reduce the same average cumulative energy consumption in each episode except 1 to 200. However, the proposed LSTM-DQN-epsilon-greedy method shows a faster and better reduction of the average cumulative energy consumption than LSTM-DQN-softmax, particularly in the first 100 episodes. Thus, our designed LSTM-DQN-epsilon-greedy method significantly reduced the average cumulative energy consumption compared to the other three methods by selecting the best energy-efficient sensor in our designed target tracking environment. The Table 4 shows the overall performance analysis for each schemes, where our proposed system has been achieved significant outcome with low energy consumption compared to other three schemes.

VII. Conclusion and Future Directions

Sensors are widely used in IoT applications (e.g., tracking and attaining target location information). In such scenarios, energy consumption optimization is a critical challenge because of the sensor battery lifespan. For this reason, an adequate learning method with Deep RL has been proposed to overcome the problem of energy consumption. The proposed idea is based on selecting the best sensor with minimum energy using the proposed Deep RL agent at a particular time to collect the target location information. The Kalman filter and LSTM-DQN-epsilon-greedy algorithms have been utilized to predict the target position and best sensor selection, respectively. Furthermore, The proposed LSTM-DQN-epsilon-greedy system compared with the other three benchmark schemes: LSTM-DQN-softmax, Dense-DQN-epsilon-greedy, and Dense-DQN-softmax. A comparative analysis was performed in terms of average cumulative reward, loss convergence, average best sensor selection accuracy, and cumulative energy consumption. The proposed LSTM-DQN-epsilon-greedy method addresses the problem of best sensor selection and converges the energy consumption issue efficiently, which is significantly improved in the proposed tracking environment than the other three methods.

References

[1] S. Li, L. Da Xu, and S. Zhao, "5g internet of things: A survey," *Journal of Industrial Information Integration*, vol. 10, Elsevier, 2018, pp. 1–9, DOI: 10.1016/j.jii.2018.01.005.

[2] A. Farhad, D.-H. Kim, S. Subedi, and J.-Y. Pyun, "Enhanced lorawan adaptive data rate for mobile internet of things devices," *Sensors*, vol. 6466, p. 21, Nov. 2020. DOI: 10.3390/s20226466.

[3] A. Mihovska and M. Sarkar, "Smart connectivity for internet of things (iot) applications," in *New Advances in the Internet of Things*, vol. 715, Springer, 2018, pp. 105–118. DOI: 10.1007/978-3-319-58190-3_7.

[4] A. Farhad, D. H. Kim, B. H. Kim, A. F. Y. Mohammed, and J. Y. Pyun, "Mobility-aware resource assignment to iot applications in long-range wide area networks," *IEEE Access*, vol. 8, pp. 186 111–186 124, 2020. DOI: 10.1109/ACCESS.2020.3029575.

[5] A. Farhad, D.-H. Kim, and J.-Y. Pyun, "Resource allocation to massive internet of things in lorawans," *Sensors*, vol. 20, p. 20, May 2020. DOI: 10.3390/s20092645.

[6] A. Ez-Zaidi and S. Rakrak, "A comparative study of target tracking approaches in wireless sensor networks," *Journal of Sensors*, vol. 2016, pp. 1–11, 2016. DOI: 10.1155/2016/3270659.

[7] Y. Zhang, "Technology framework of the internet of things and its application," in *2011 IEEE International Conference on Electrical and Control Engineering 16–18 Sep., Yichang, China*, pp. 4109–4112. DOI: 10.1109/ICECENG.2011.6057290.

[8] O. Demigha, W.-K. Hidouci, and T. Ahmed, "On energy efficiency in collaborative target tracking in wireless sensor network: A review," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1210–1222, 2012. DOI: 10.1109/SURV.2012.042512.00030.

[9] B. Sebastian and P. Ben-Tzvi, "Support vector machine based real-time terrain estimation for tracked robots," *Mechatronics*, vol. 62, p. 102 260, 2019. DOI: 10.1016/j.mechatronics.2019.102260.

[10] P. R. Montague, "Reinforcement learning: An introduction, by sutton, rs and barto, ag," *Trends in cognitive sciences*, vol. 3, no. 9, p. 360, 1999. DOI: https://doi.org/10.1016/S1364-6613(99)01331-5.

[11] M. Ali Imran, A. Flávia dos Reis, G. Brante, P. Valente Klaine, and R. Demo Souza, "Machine learning in energy efficiency optimization," *Machine Learning for Future Wireless Communications*, pp. 105–117, 2020. DOI: 10.1002/9781119562306.ch6.

[12] N. Liu, Z. Li, J. Xu, Z. Xu, S. Lin, Q. Qiu, J. Tang, and Y. Wang, "A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), 5-8 June, Atlanta, GA, USA*, pp. 372–382. DOI: 10.1109/ICDCS.2017.123.

[13] Z. Xu, Y. Wang, J. Tang, J. Wang, and M. C. Gursoy, "A deep reinforcement learning based framework for power-efficient resource allocation in cloud rans," in *2017 IEEE International Conference on Communications (ICC) 21–25 May, Paris, France*, pp. 1–6. DOI: 10.1109/ICC.2017.7997286.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[15] J. Li, Z. Xing, W. Zhang, Y. Lin, and F. Shu, "Vehicle tracking in wireless sensor networks via deep reinforcement learning," *IEEE Sensors Letters*, vol. 4, no. 3, pp. 1–4, 2020. DOI: 10.1109/LSENS.2020.2976133.

[16] H. Nguyen and H. La, "Review of deep reinforcement learning for robot manipulation," in *2019 Third IEEE International Conference on Robotic Computing (IRC), 25-27 Feb., Naples, Italy*, pp. 590–595. DOI: 10.1109/IRC.2019.00120.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015. DOI: 10.1038/nature14236.

[18] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[19] Y. Kim and H. Bang, "Introduction to kalman filter and its applications," in *Introduction and Implementations of the Kalman Filter*, IntechOpen, November 5th 2018. DOI: 10.5772/intechopen.80600.

[20] H. Ma and B. Ng, "Collaborative signal processing framework and algorithms for targets tracking in wireless sensor networks," in *Microelectronics: Design, Technology, and Packaging II*, International Society for Optics and Photonics, vol. 6035, 2006, 60351K. DOI: 10.1117/12.637948.

[21] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration," *IEEE Signal processing magazine*, vol. 19, no. 2, pp. 61–72, 2002. DOI: 10.1109/79.985685.

[22] W. Li and C. Han, "Dual sensor control scheme for multi-target tracking," *Sensors*, vol. 18, no. 5, p. 1653, 2018. DOI: 10.3390/s18051653.

[23] P. Wang, L. Ma, and K. Xue, "Multitarget tracking in sensor networks via efficient information-theoretic sensor selection," *International Journal of Advanced Robotic Systems*, vol. 14, no. 5, p. 1 729 881 417 728 466, 2017.

[24] S. Lata and S. Mehfuz, "Machine learning based energy efficient wireless sensor network," in *2019 IEEE International Conference on Power Electronics, Control and Automation (ICPECA), 16-17 Nov., New Delhi, India*, pp. 1–5. DOI: 10.1109/ICPECA47973.2019.8975526.

[25] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1996–2018, 2014. DOI: 10.1109/COMST.2014.2320099.

[26] M. Waleed, T.-W. Um, T. Kamal, A. Khan, and A. Iqbal, "Determining the precise work area of agriculture machinery using internet of things and artificial intelligence," *Applied Sciences*, vol. 10, no. 10, p. 3365, 2020. DOI: 10.3390/app10103365.

[27] R. Hosseini and H. Mirvaziri, "A new clustering-based approach for target tracking to optimize energy consumption in wireless sensor networks," *Wireless Personal Communications*, vol. 114, pp. 3337–3349, 2020. DOI: 10.1007/s11277-020-07534-5.

[28] T. Zou, Z. Li, S. Li, and S. Lin, "Adaptive energy-efficient target detection based on mobile wireless sensor networks," *Sensors*, vol. 17, p. 1028, May 2017. DOI: 10.3390/s17051028.

[29] J. Feng and H. Zhao, "Energy-balanced multisensory scheduling for target tracking in wireless sensor networks," *Sensors*, vol. 18, p. 3585, Oct. 2018. DOI: 10.3390/s18103585.

[30] M. I. Khan and B. Rinner, "Energy-aware task scheduling in wireless sensor networks based on cooperative reinforcement learning," in *2014 IEEE International Conference on Communications Workshops (ICC), 10-14 June, Sydney, NSW, Australia*, pp. 871–877. DOI: 10.1109/ICCW.2014.6881310.

[31] J. Zhu, Y. Song, D. Jiang, and H. Song, "A new deep-q-learning-based transmission scheduling mechanism for the cognitive internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2375–2385, 2017. DOI: 10.1109/JIOT.2017.2759728.

[32] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J.-S. Oh, "Semisupervised deep reinforcement learning in support of iot and smart city services," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 624–635, 2017. DOI: 10.1109/JIOT.2017.2712560.

[33] S. Nayak, B. B. Misra, and H. S. Behera, "Impact of data normalization on stock index forecasting," *International Journal of Computer Information Systems and Industrial Management Applications*, vol. 6, no. 2014, pp. 257–269, 2014.

[34] G. V. Rossum, *Python*, https://www.python.org/, (Accessed on March 12, 2020), 1991.

[35] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *Tensorflow*, `https://www.tensorflow.org/`, (Accessed on April 15, 2020), 2015.

[36] F. Chollet, *Keras*, `https://keras.io/`, (Accessed on April 15, 2020), 2015.

[37] S. M. Sultan, M. Waleed, J.-Y. Pyun, and T.-W. Um, "Energy conservation for internet of things tracking applications using deep reinforcement learning," *Sensors*, vol. 21, no. 9, p. 3261, 2021. DOI: 10.3390/s21093261.

Publication

Salman, Md, Sultan; Waleed, Muhammad; Pyun, Jae-Young; Um, Tai-Won: *Energy Conservation for Internet of Things Tracking Applications Using Deep Reinforcement Learning.* Sensors 08/2021; 21, 3261; DOI:10.3390/s21093261

Acknowledgements

In the first place, I am grateful to the creator Almighty ALLAH. Then, I have no words to elucidate my appreciation towards my honourable advisor, my advisor, Professor Jae-Young Pyun. I am enormously thankful that I have enjoyed advanced hardware resources, absolute creative freedom, limitless, ongoing support, and great individualism under his supervision. Moreover, I would like to mention his heartfelt meticulousness, which drives me to be more precise in my research career.

Secondly, I would like to mention Professor Tai-Won UM with great respect because of his tremendous support to finish my Master's degree. I would always be indebted to him for the valuable lessons of professionalism, time management, and impeccable mentorship.

Thirdly, I am obligated for the chance to be a part of such a unique group of students, faculty, and staff in the Department of Information and Communication Engineering, Chosun University. Also, I would like to show my sincere acknowledgement to Wireless and Mobile Communication System Lab for providing me with such an excellent opportunity and an atmosphere to improve academically and otherwise. I am thankful to my lab-mates for their moral as well as academic support. Furthermore, I would like to thank all my seniors and friends from Bangladesh at Chosun University for their affection and cooperation, making my life easier and cheerful in South Korea.

Finally, I would like to thank my parents, wife, family members, and friends for their constant support in difficult times. It would have been impossible for me to achieve anything without their encouragement and assistance. I would like to dedicate my work to them.