Using Deep Q-learning To Prolong the Lifetime of Correlated Internet of Things Devices

Jernej Hribar*, Andrei Marinescu*, George A. Ropokis†, and Luiz A. DaSilva*
*CONNECT- Research Centre at Trinity College Dublin, Dublin 2, Ireland
† CentraleSupélec/IETR, CentraleSupélec Campus de Rennes, 35510 Cesson Sevigne, France
E-mail: jhribar@tcd.ie.

***Abstract*— Battery-powered sensors deployed in the Internet of Things (IoT) require energy-efficient solutions to prolong their lifetime. When these sensors observe a physical phenomenon distributed in space and evolving in time, the collected observations are expected to be correlated. In this paper, we propose an updating mechanism leveraging Reinforcement Learning (RL) to take advantage of the exhibited correlation in the information collected. We implement the proposed updating mechanism employing deep Q-learning. Our mechanism is capable of learning the correlation in the information collected and determine the frequency with which sensors should transmit their updates, while taking into consideration a highly dynamic environment. We evaluate our solution using environmental observations, namely temperature and humidity, obtained in a real deployment. We demonstrate that our mechanism is capable of adapting the transmission frequency of sensors' updates according to the ever-changing environment. We show that our proposed mechanism is capable of significantly extending battery-powered sensors' lifetime without compromising the accuracy of the observations provided to the IoT service.**

***Index Terms*—Internet of Things (IoT), extending lifetime, reinforcement learning, deep Q-learning**

I. Introduction

Over the next few years, billions of devices are expected to be deployed in the Internet of Things (IoT) network [1], many of which will be low-cost sensors powered by non-rechargeable batteries. These battery-powered sensors will provide vital information regarding the environment to numerous services, e.g. smart farming [2], autonomous vehicles [3], air pollution monitoring [4], etc. Providing accurate and up-to-date information to services while keeping the battery-powered sensors functional as long as possible is one of the primary challenges in the IoT.

It is possible to take advantage of the correlation exhibited in measurements when multiple sensors measure the same physical phenomenon occurring in the environment. In our work, we utilise the correlation in the data collected by multiple sensors to prolong battery-powered sensors' lifetime. In particular, we have designed an energy-efficient updating mechanism capable of taking advantage of correlation exhibited in the collected information by applying a Reinforcement Learning (RL) [5] technique.

Most existing research that applies RL in the context of IoT has focused on exploiting devices' behaviour in the physical layer to improve their energy performance. For example, in [6] authors used Q-learning to enhance the spectrum utilisation of industrial IoT devices. They demonstrated that devices are capable of learning a channel selection policy to avoid collisions, thus improving their energy efficiency. Similarly, in [7], the authors applied Q-learning to improve device access to the channel, to avoid collisions. The use of deep RL was investigated in [8], where authors relied on Bluetooth signal strength to improve indoor users' location estimation. In contrast to the works mentioned above and those described in [9] focusing mostly on the physical layer, we learn from information collected. Applying RL to learn from the content of information collected to prolong the sensors' lifetime has not been proposed before, to the best of our knowledge.

In this paper, we propose an updating mechanism capable of learning the frequency of updates, i.e., how often an IoT sensor should transmit updated readings. Our approach prolongs battery-powered sensors' lifetime by leveraging correlation exhibited in observations collected, without hindering the accuracy and timeliness of the information provided to services relying on these observations. We define the decision-making problem that our proposed mechanism is capable of solving in Section II. To solve the proposed problem using a Deep Q-Network (DQN) [10], we describe the system dynamics using states, actions, and rewards from an RL perspective (Section III B). We also describe the overall mechanism in the form of a block diagram (Section III C). We evaluate the proposed mechanism using data obtained from a real deployment and show that the system is capable of prolonging the minimum expected sensor lifetime by over two and a half times (Section IV). Finally, we discuss open issues and our future work (Section V).

II. Problem formulation

In our work, we focus on inexpensive battery-powered sensors transmitting observations. These sensors are constrained in terms of available computing power, communication capabilities, and energy. Furthermore, such sensors rely on the use of sleep mode to preserve energy. When a sensor is in sleep mode, the rest of the network cannot communicate with it. Consequently, the network controller, responsible for collecting observations, has to inform each sensor, while the sensor is still in active mode, when it should wake up again and transmit the next observation. The low power IoT sensor is usually in active mode only after it has transmitted. For example, a sensor using Long Range Wide Area Network (LoRaWAN)

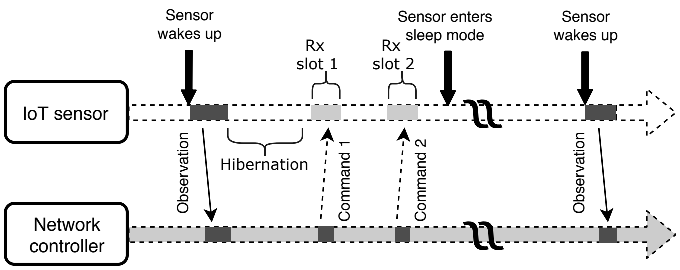

Fig. 1: Message sequence of an IoT sensor using LoRaWAN.

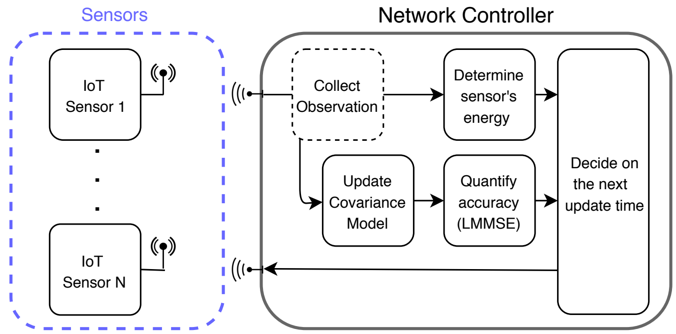

Fig. 2: A high-level overview of the decision-making process in a network controller managing N IoT sensors.

class A radio, will listen for two short time-windows after it has transmitted, as illustrated in the LoRaWAN message sequence in Fig. 1 [11].

The network controller has two objectives when setting the transmission times for sensors. The first objective is to satisfy the accuracy goals set by services for the collected observations. The second objective is to prolong the lifetime of battery powered sensors. Both objectives are achievable when sensors gather correlated information, as we demonstrated in [12]; our previous work, however, we did not propose a procedure for the controller to establish the desired frequency of updates. The network controller decides on the sensors' next update time by evaluating the accuracy of collected observation and the sensors' available energy. We summarize the decision-making process by the network controller in Fig. 2. In what follows, we present our methodology for modelling the observations' accuracy as well as the network controller decision process.

A. Quantifying observations' accuracy: LMMSE

We consider a network observing a phenomenon distributed over a geographical area and evolving in time t. The network employs N sensors deployed at positions $\boldsymbol{x}_n, n = 1, \dots, N$. We use notation $Z(\boldsymbol{x}_n, t)$ to denote the value of the observed physical process at time instance t, at location $\boldsymbol{x}_n$. We can write collected observations into a vector $\mathbf{y} = [y_1, \dots, y_N]^T$ with $y_n = Z(\boldsymbol{x}_n, t_n)$ where $t_n \in [0, t]$ is the latest time at which sensor n has reported an observation. Then, using a simple Linear Minimum Mean Square Error (LMMSE) estimator, we can approximate the measurement at position $\boldsymbol{x}$ at time instance t, as:

$$\hat{Z}(\boldsymbol{x}, t) = a_0 + \sum_{n=1}^{N} a_n y_n, \tag{1}$$

where $a_n, n = 0, \dots, N$ are LMMSE estimator weights. We obtain the LMMSE estimator weight vector $\mathbf{a} = [a_0, \dots, a_N]^T$ as follows:

$$\mathbf{a} = (\boldsymbol{C_{YY}})^{-1} \boldsymbol{C_{YZ}}. \tag{2}$$

The $\boldsymbol{C_{YY}}$, $\boldsymbol{C_{YZ}}$ are covariance matrices:

$$\boldsymbol{C_{YY}} = \begin{bmatrix} \sigma\rho_{y_1 y_1} \dots \sigma\rho_{y_1 y_N} \\ \vdots \quad\quad \vdots \\ \sigma\rho_{y_N y_1} \dots \sigma\rho_{y_N y_N} \end{bmatrix}; \boldsymbol{C_{YZ}} = \begin{bmatrix} \sigma\rho_{y_1 z} \\ \vdots \\ \sigma\rho_{y_N z} \end{bmatrix}; \tag{3}$$

in which σ represents standard deviation and ρ represents covariance. Covariance provides a measure for the relationship between two observations separated in time and space. For example, when temperature changes at one location, the covariance enables us to calculate the probability that the temperature will also have changed by a certain value at another distant location. In our model, $\boldsymbol{C_{YY}}$ captures the covariance between observations at different locations, taken at the same time, and $\boldsymbol{C_{YZ}}$ captures the covariance between observations taken at the same location, at different times. In this work, we adopt the separable covariance model defined in [13] that allows us to express the correlation between two observations with time difference $\Delta_i(t)$ at locations at a distance r_i as:

$$\rho_i(r_i, t) = exp(-\theta_2(t) r_i - \theta_1(t) \Delta_i(t)). \tag{4}$$

Note that $\theta_1(t)$ and $\theta_2(t)$ are scaling parameters of time and space, respectively. Both parameters change over time and are extracted from the obtained observations. In our work, we follow a scaling extraction method with Pearson's correlation coefficient formula for samples, as described in [14].

Using Eq. (1) and Eq. (2) we can obtain estimates for the observed phenomenon, at the point $\boldsymbol{x}_n$, even at time instances in which the n-th sensor is in sleep mode. However, the system requires a way to evaluate how accurate these estimations are. For that purpose, we use the Mean Square Error (MSE):

$$\varepsilon\big(\boldsymbol{x}, t | \theta_1(t), \theta_2(t)\big) = \sigma^2 - \boldsymbol{C_{ZY}} \mathbf{a}, \tag{5}$$

where $\boldsymbol{C_{ZY}}$ is the transpose of $\boldsymbol{C_{YZ}}$ defined above.

By assessing the quality of estimates in the absence of fresh observations, the network controller can set the update times in such a way to ensure accurate collection of observations. However, the determined update time might not result in accurate estimation of the observed phenomenon due to changes in covariance model scaling parameters. The network controller should be able to anticipate such changes and act before they happen, i.e., while the sensor is still active. Additionally, the controller should be aware of sensors' available energy when deciding on sensors' next update time.

B. Prolonging sensors' lifetime in a dynamic environment

The sensors' lifetime depends on the frequency of transmitted observations and on the continuous power consumption, i.e., the minimal power sensors always require to function. The sensor's lifetime can be simply modeled as in [15]:

$$\mathbb{E}[\mathcal{L}] = \frac{E_0}{P_c + \frac{\mathbb{E}[E_{tr}]}{T}}, \tag{6}$$

where P_c represents the continuous power consumption, T denotes the time between updates, $\mathbb{E}[E_{tr}]$ represents the expected energy required to acquire and transmit the observation, and E_0 represents the sensors' starting energy.

The network controller seeks to prolong the lifetime of battery-powered sensors by maximising the time between two consecutive updates by a sensor, while keeping the accuracy of collected observations at every location of interest within the pre-specified boundary. In a real deployment, services dictate which locations are of interest. In this paper, we define every sensor location, i.e., $\boldsymbol{x}_n$, as a location of interest. The decision is also based the on the sensors' available energy, which the network controller can determine from the sensor's reported power supply measurement. The network controller compares each sensors' energy with the energy available to other sensors and decides on the update time accordingly. Ideally, the system will set a higher update rate for a sensor with more available energy, to provide a longer lifetime to those sensors with less available energy.

In our system, the MSE continuously varies, as every received observation can change the covariance model scaling parameters. These changes are application-specific: for example, in a smart factory, environmental changes are very frequent due to many factory processes simultaneously impacting the observed environment. In contrast, the changes in a smart farming scenario tend to be much more gradual. However, regardless of the frequency of changes, the system requires a means to adapt sensors' update time to the ever-changing environment. To that end, we propose for the network controller to employ RL to decide on the next update time. Using RL, the network controller can find a long-term updating policy to collect accurate observations and prolong battery-powered sensors' lifetime.

III. Reinforcement Learning Approach

RL allows an agent to learn its optimal behaviour, i.e., a set of actions to take in every state, solely from interactions with the environment. In our case, the agent is the network controller. The goal of learning is to determine the next update time at which each sensor should transmit its observations. Our agent observes the environment through the value of the MSE in the latest observation of the physical phenomenon reported by each sensor, as well as the remaining energy available to each sensor. The MSE is a non-stationary process, changing with every new observation, and the system exhibits non-Markovian properties. However, RL has been proven to be applicable even when the system is non-Markovian [16]. In such a case, using an Artificial Neural Network (ANN) enables the agent to reconstruct the hidden part of environment in the neural network. We implement the updating mechanism, i.e., the network controller decision process of when a sensor should transmit its next observation, using deep Q-learning.

A. Q-learning

The Q in Q-learning [17] stands for the quality of an action in a given state. The learning agent should take the action with the highest Q-value, unless the algorithm decides to explore. The agent learns the best action possible by updating the Q-value every time it takes an action and observes a reward. When the agent takes enough actions in every state of the environment, it can correctly approximate the real action values, i.e., Q-values, associated with every state. With Q-values determined, the agent can choose the optimal action in every state. A Q-value is calculated as follows:

$$Q^{new}(s,a) \leftarrow Q(s,a) + \alpha\Big(R(s') + \gamma \max_{a'} Q(s',a') - Q(s,a)\Big) \tag{7}$$

where $Q(s,a)$ is the previous Q-value, α is the learning rate, γ is the discount factor, and R is the reward observed in the new state s' after taking action a in state s. The $\max_{a'} Q(s',a')$ stands for an estimate of the optimal future value an agent can acquire from the next state s'.

The role of the agent, i.e., the network controller, is to determine the sensor state and to select the action with the highest Q-value. Every time a sensor transmits an observation, the network controller will respond by instructing the sensor for how long it should enter sleep mode, i.e., set its next update time. Next, we define states, actions, and reward functions, i.e., a tuple in $\langle \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$ that enables the network controller to determine sensors' optimal update times.

B. States, Actions, and Rewards

Our *state* space, $\mathcal{S}$, captures three critical aspects of the decision-making process in the network controller: a sensor's current update time, its available energy, and the estimation error value. Whenever a sensor transmits an observation, the network controller stores the information regarding sensor's update time, i.e., T, available energy, i.e., E, and value of average MSE since the last transmission, i.e., $\bar{\varepsilon}$. The learning agent then uses those values to represent each sensor state. The agent requires information regarding sensors' update times and MSE values to reconstruct the hidden part of the environment, while the energy levels enable the agent to ascertain which sensor in the network most needs to save energy. The total number of states is:

$$|\mathcal{S}| = (TE\bar{\varepsilon})^N, \tag{8}$$

with N representing the number of sensors under the agent's control. Using ANN can efficiently solve problems associated with a sizable non-linear state space [10].

In contrast to the state space, we limit *actions*, i.e., the cardinality of set $\mathcal{A}$, to five. Actions enable an agent to learn the best update time by decreasing or increasing the update time in time-steps. Furthermore, the agent can select to make a large or small step to adapt more quickly or more gradually. We denote an action to increase the update time for one time-step U_{incr1} and for ten U_{incr10}. With U_{dec1} and U_{dec10} we denote a decrease of update time for one and ten time-steps, respectively. If the agent decides to maintain the current update time unchanged, it selects action U_{cons}. As we show in the next section, the system can, by using this action space, adapt to any changes in the environment promptly. Additionally, a limited action space prevents rapid fluctuations in the system when the learning algorithm decides to explore.

We designed the *reward function* to aid the agent to quickly adapt to the changing environment.To achieve both network controller objectives, i.e., collecting accurate observations and prolonging sensors' lifetime, we split the reward function into two parts as follows:

$$\mathcal{R}(s) = \phi_{acc}\mathcal{R}_{acc}(s) + \phi_{en}\mathcal{R}_{en}(s). \tag{9}$$

$\mathcal{R}_{acc}(s)$ rewards accurate collection of observations and $\mathcal{R}_{en}(s)$ rewards energy conservation. We weigh each part of the reward function, with ϕ_{acc} for accuracy, and ϕ_{en} for energy. Our learning agent receives the reward after taking a decision for the update time and receiving the next update from the sensor. We define this learning cycle as one episode, i.e., our agent receives the reward after the end of each episode.

The accuracy reward depends on whether the set accuracy boundary, i.e., ε_{tar}, was satisfied in the episode. We compare the average MSE in an episode, i.e. $\overline{\varepsilon}(t)$, to the target MSE. The accuracy reward is as follows:

$$\mathcal{R}_{acc}(s) = \begin{cases} \text{when} \quad \overline{\varepsilon}(t) \leq \varepsilon_{tar} \ : \\ \quad \left(\frac{\overline{\varepsilon}(t)}{\varepsilon_{tar}}\right)^2 + 10\,\overline{\varepsilon}_{\Delta}(t) \\ \text{when} \quad \overline{\varepsilon}(t) > \varepsilon_{tar} \ : \\ \quad \left(\frac{\overline{\varepsilon}(t) - \varepsilon_{tar}}{\varepsilon_{tar}}\right)^2 - 10\,\overline{\varepsilon}_{\Delta}(t) \end{cases} \tag{10}$$

where $\overline{\varepsilon}_{\Delta}(t)$ is the difference in the average estimation error in since the last transmission, i.e., $\overline{\varepsilon}_{\Delta}(t) = \overline{\varepsilon}(t) - \overline{\varepsilon}(t-T)$. Our objective is for the average MSE observed in an episode to be as close as possible to the target ε_{tar}, without exceeding it. The accuracy reward in Eq. (10) accomplishes that.

Our energy reward function depends on the change in the update times and how a sensor's available energy compares to the average sensor's energy in the network. We write the energy reward as follows:

$$\mathcal{R}_{en}(s) = \begin{cases} \frac{2(E_{avg}-E)}{E_{avg}}, & \text{if} \ \ T_{\Delta} > 0 \\ \frac{E_{avg}-E}{E_{avg}}, & \text{if} \ \ T_{\Delta} = 0 \\ \frac{2(E-E_{avg})}{E_{avg}}, & \text{if} \ \ T_{\Delta} < 0 \end{cases}, \tag{11}$$

where T_{Δ} represents the change in update time, E is the sensor's available energy, and E_{avg} is the average energy available among all sensors in the network.

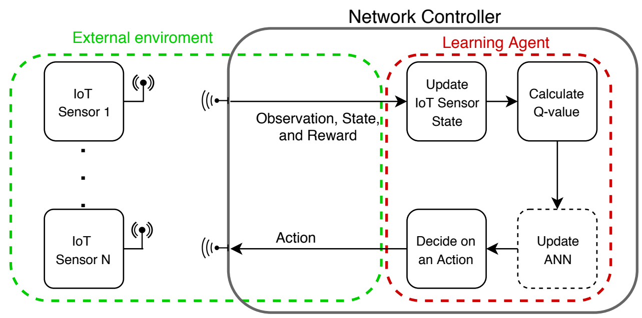

Fig. 3: Diagram of the proposed updating mechanism.

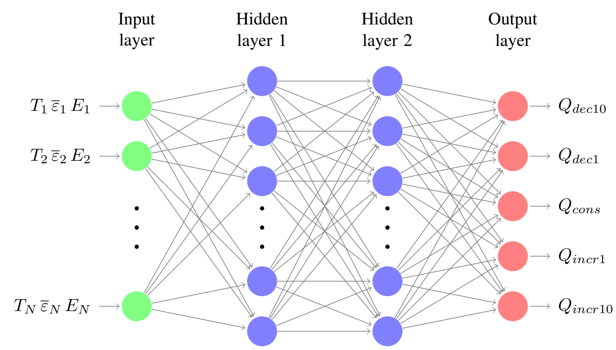

Fig. 4: We employ ANN with two hidden layers, each with 24 neurons. We use MSE loss function and Adam optimization algorithm to train ANN weights. We apply ReLU activation function for the hidden layers and linear activation function for the output layer.

C. *Proposed learning scheme*

Figure 3 shows a high-level model of our proposed mechanism. Sensors collecting information, along with the part of the network controller responsible for processing information, represent the external environment to our learning agent. An observation sent by an IoT sensor starts the learning cycle. The network controller then passes the necessary information (state and reward) to the learning agent. The learning agent then updates the state space and passes these updates to the ANN. The output of the ANN indicates which action the network controller should take (i.e., the action with the highest Q-value). The updating mechanism uses an ϵ-greedy approach: in our case, $\epsilon = 0.15$. Due to constant changes in the environment, the learning agent has to sometimes explore random actions to find the optimal action. In the last step, the network controller then transmits the action, i.e., the new update time, to the IoT sensor.

We implemented the ANN, as presented in Fig. 4, using Keras [18], a deep learning Python library. To train the ANN, the learning agent requires the values of state spaces of N sensors and corresponding Q-values. The first inserted state space values ($T_1\,\overline{\varepsilon}_1\,E_1$) are from the sensor that transmitted last, followed by the state space values of the second last sensor ($T_2\,\overline{\varepsilon}_2\,E_2$), and so on. We train the ANN periodically, using batches of recently observed states, rewards, and actions, to shorten the response time. Our learning agent is capable of responding within 2 to 3 ms, thus satisfying the timing constraints set by sensors' communications technology. For example, a device using LoRaWAN radio typically enters hibernation mode for more than a second. Additionally, response messages impose no additional energy cost to a sensor because the communication standard requires it to always listen to the channel before entering sleep mode.

We evaluate our mechanism using data obtained from a real deployment, presented in the next section.

IV. Evaluation

In this section, we evaluate our proposed solution using data provided by the Intel Berkeley Research laboratory [19]. In our simulated network, sensor locations and transmitted observations (temperature and humidity) are based on the provided Intel data. We use nine days of measurements collected from 50 sensors. We list the static simulation parameters in Table I. The selected energy parameters mimic power consumption of an IoT sensor using LoRaWAN radio. We obtained the power parameters following the analysis presented in [20].

In our simulations, we set the time-step (ts) to 10s. Each sensor starts with the same update time. We selected the starting update time for temperature, i.e., T_{0t}, and humidity, i.e., T_{0h}, by analyzing the dataset. We determined that if sensors transmit temperature updates every $1003s$ the average difference between two consecutive observations will be ± 0.30 °C, and if they transmit a humidity observation every $968s$ the average difference between consecutive observations will be 0.5%. For a fair comparison, throughout our evaluation, the updating mechanism keeps the average accuracy of temperature estimations within 0.30 °C of the real temperature and within 0.5% of the real air humidity percentage. To reduce the amount of required computations for MSE and estimation of real value we limit the number of used observations. We take eight observations from the sensors closest to the estimation location.

TABLE I: Simulation Parameters

Parameter	Value	Parameter	Value
ϕ_{acc}	6	ϕ_{en}	4
α	1	γ	0.2
T_{0t}	$100\ ts$	T_{0h}	$97\ ts$
time-step (ts)	$10s$	P_c	$15uW$
E_0	$6696J$	$\mathbb{E}[E_{tr}]$	$78.7mJ$
ϵ	0.15		

In Fig. 5 we show the change of update time and $\overline{\varepsilon}$ over a number of episodes for two sensors in a system of eight sensors. We iterate over the dataset four times. Each number in Fig. 5 represents the end of a dataset iteration. In Fig. 5(a) we plot the update time over a number of episodes for a sensor with above average available energy (95%), while in 5(b) we plot update times of a sensors with below average energy (50%). As we show, our updating mechanism sets the update time of a sensor with less energy significantly higher in comparison to the update time of a sensor with more energy available. Our updating mechanism is trying to balance the energy levels among sensors by setting a uneven update time.

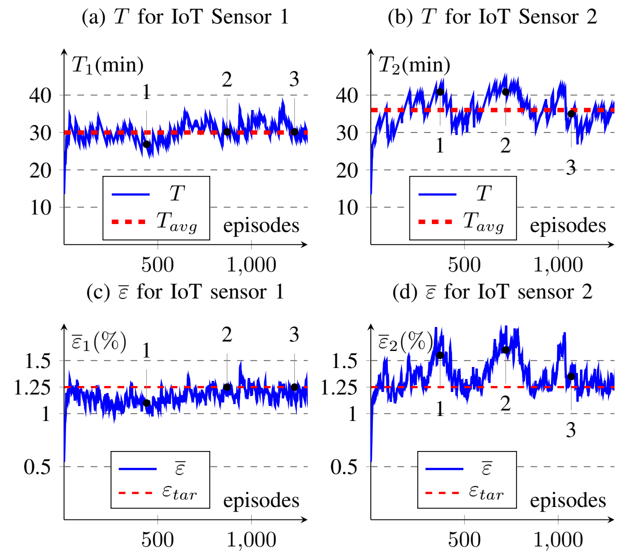

Fig. 5: Two IoT sensors learning over a number of episodes. The numbers in the graphs mark the end of an iteration over the dataset.

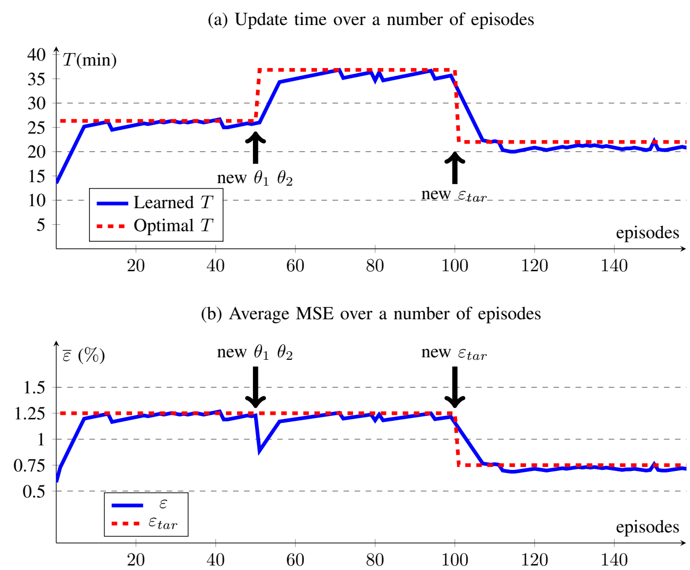

Fig. 6: Updating mechanism searching for the optimal update time, arrows indicate change in the covariance model scaling parameters or ε_{tar}.

Simultaneously, as we show in Fig. 5(c) and (d) the agent is keeping the MSE close to the set target, i.e., ε_{tar}.

To show that the updating mechanism is capable of finding the optimal solution, i.e., capable of determining the maximal update time possible, we then test the mechanism's performance in a system with only two sensors. We set only one sensor in the learning mode while the other keeps the update time constant. Furthermore, covariance model scaling parameters changed only in selected episodes. Such a system enables us to also obtain the optimal solution for comparison purposes. We changed the scaling parameters at episode 50. At episode 100 we changed ε_{tar}. The mechanism is always capable of adapting and finding the optimal update time, as

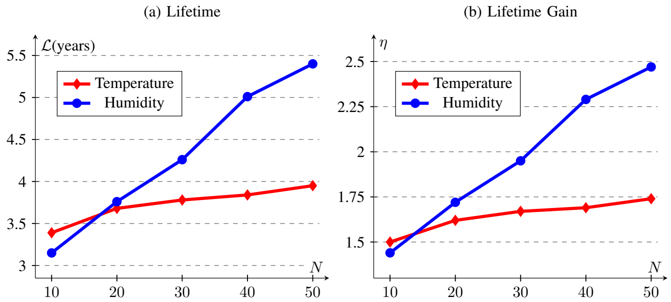

Fig. 7: Sensors' lifetime and lifetime gain achieved by our updating mechanism as the number of sensors under its control increases.

shown in Fig. 6(a). In Fig. 6(b), we show the change of $\overline{\varepsilon}$ over a number of episodes: the mechanism always converges toward the selected ε_{tar}.

Next, we test the updating mechanism performance as the number of sensors, N, under its management increases. The expected lifetime of sensors increases with more sensors in the network, as we show in Fig. 7(a). The gain for using correlated information is higher when observing humidity, due to higher correlation exhibited in the observations collected. We calculate the expected lifetime using Eq. (6). In our calculation, we use the average update time our updating mechanism achieves on the 9^{th} day. We assume that each sensor is equipped with a $620mAh$ Lithium battery. To show the improvement in comparison to the baseline case, we calculate the lifetime gain, i.e., η_i, as the ratio between the lifetime achieved using our mechanism and that achieved in the baseline case:

$$\eta = \frac{\mathbb{E}[\mathcal{L}]}{\mathbb{E}[\mathcal{L}_0]}. \tag{12}$$

We calculated the baseline $\mathbb{E}[\mathcal{L}_0]$ using T_{0t} and T_{0h}, resulting in the expected lifetime of 2.26 years when observing temperature, and 2.05 years when observing humidity. Our updating mechanism can significantly prolong the expected lifetime of battery-powered devices. When measuring humidity the expected sensor lifetime can be extended to over four years, while for sensors measuring temperature the expected lifetime extends to over three and a half years.

V. Final Remarks and Future Work

In this paper, we applied deep Q-learning to prolong the lifetime of battery-powered sensors. The proposed updating mechanism is capable of adjusting updates according to a sensor's available energy and the sensed accuracy of the observed physical phenomenon. We demonstrated that it is capable of performing in a dynamic IoT network environment. We evaluated our proposed mechanism using data obtained from a real deployment. Our results show that it is possible to increase the lifetime of battery-powered sensors by a factor of three by taking advantage of correlated information.

In our future work, we will consider a network of sensors using different primary power sources, e.g., mains powered or event-based energy harvesting. To provide the network controller with a capability to manage such devices effectively, we will expand the list of available actions. Additionally, we will design a new reward function to reflect the different energy sources across the sensors. In such a network, the primary goal of learning will be achieving an energy-aware balanced scheduling of sensors' updates.

Acknowledgments

This work was funded in part by the European Regional Development Fund through the SFI Research Centres Programme under Grant No. 13/RC/2077 SFI CONNECT and by the SFI-NSFC Partnership Programme Grant Number 17/NSFC/5224.

References

[1] P. Cerwall et al., "Ericsson mobility report," June 2018.
[2] S. Wolfert, L. Ge, C. Verdouw, and M.-J. Bogaardt, "Big data in smart farming-a review," *Agricultural Systems*, vol. 153, pp. 69–80, 2017.
[3] M. Gerla, E.-K. Lee, G. Pau, and U. Lee, "Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds," in *WF-IoT*. IEEE, 2014, pp. 241–246.
[4] A. Al-Ali, I. Zualkernan, and F. Aloul, "A mobile GPRS-sensors array for air pollution monitoring," *IEEE Sensors Journal*, vol. 10, no. 10, pp. 1666–1671, 2010.
[5] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press, 1998.
[6] F. Li, K.-Y. Lam, Z. Sheng, X. Zhang, K. Zhao, and L. Wang, "Q-learning-based dynamic spectrum access in cognitive industrial internet of things," *Mobile Networks and Applications*, pp. 1–9, 2018.
[7] Y. Chu, P. D. Mitchell, and D. Grace, "ALOHA and Q-learning based medium access control for wireless sensor networks," in *Wireless Communication Systems (ISWCS), 2012 International Symposium on*. IEEE, 2012, pp. 511–515.
[8] 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, 2018.
[9] 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.
[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.
[11] *SX1272/73 - 860 Mhz to 1020 MHz Low Power Long Range Transceiver Datasheet*, SAMTECH Corporation, March 2015, revision 3. [Online]. Available: https://www.semtech.com/uploads/documents/sx1272.pdf
[12] J. Hribar, M. Costa, N. Kaminski, and L. A. Da Silva, "Using correlated information to extend device lifetime," *IEEE Internet of Things Journal*, 2018.
[13] N. Cressie and H.-C. Huang, "Classes of nonseparable, spatio-temporal stationary covariance functions," *Journal of the American Statistical Association*, vol. 94, no. 448, pp. 1330–1339, 1999.
[14] T. Gneiting, "Nonseparable, stationary covariance functions for space-time data," *Journal of the American Statistical Association*, vol. 97, no. 458, pp. 590–600, 2002.
[15] Y. Chen and Q. Zhao, "On the lifetime of wireless sensor networks," *Communications letters*, vol. 9, no. 11, pp. 976–978, 2005.
[16] T. Jaakkola, S. P. Singh, and M. I. Jordan, "Reinforcement learning algorithm for partially observable markov decision problems," in *Advances in neural information processing systems*, 1995, pp. 345–352.
[17] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
[18] F. Chollet *et al.*, "Keras," https://keras.io, 2015.
[19] P. Bodik, W. Hong, C. Guestrin, S. Madden, M. Paskin, and R. Thibaux, "Intel lab data," *Online dataset*, 2004.
[20] M. Costa, T. Farrell, and L. Doyle, "On energy efficiency and lifetime in low power wide area network for the Internet of Things," in *IEEE, CSCN 2017*.