On Predicting Sensor Readings with Sequence Modeling and Reinforcement Learning for Energy-Efficient IoT Applications

Roufaida Laidi, Djamel Djenouri, Ilangko Balasingham.

***Abstract*—Prediction of sensor readings in event-based IoT (Internet of Things) applications is considered. A new approach is proposed, which allows turning off sensors in periods when their readings can be predicted, and thus preserving energy that would be consumed for sensing and communications. The proposed approach uses a long short term memory (LSTM) model that learns spatio-temporal patterns in sequences of sensorial data for future predictions. The LSTM model and the sensors collaboratively monitor the environment. They are controlled by a reinforcement learning (RL) agent that dynamically decides about using the LSTM prediction vs. physical sensing in a way that maximizes energy-saving while maintaining accuracy in prediction. Two approaches are used for the RL, 1) Markov decision process (MDP) model-based for low scale applications, and 2) Deep Q-Network based for larger scales. Compared to the current literature, the proposed solution is unique in predicting all sensor readings for real-time event detection and providing a model capable of learning long term spatio-temporal correlations, which enables balancing power conservation and detection accuracy. We compare the proposed solutions to the most relevant state-of-the-art approaches using a large real-dataset collected in a dynamic space, by measuring the accuracy, the amount of consumed energy, the network's lifetime, the latency, and the missed events' ratio. To investigate the scalability of the solutions, these parameters are calculated for different network sizes. The results show that the system achieves 50% accuracy with 32% of activation time and 75% accuracy with 60% activation time.**

***Index Terms*—Energy-efficient IoT applications, sensor readings prediction, sequence modeling, deep learning, reinforcement learning, dynamic programming, deep Q-learning.**

I. Introduction

A. Motivation

THE lifetime of batteries used in current micro-embedded systems is one of the persistent limitations in most IoT applications. Solutions based on duty cycling the sensor nodes help to reduce energy wasted by the radio in idle listening, i.e., the periods when the sensor is waiting for a packet in vain [1], but they remain insufficient. For example, this approach does not apply to the sensing hardware in event-based applications, which should be kept on all the time to detect events. Another example is the applications with high data exchange rates in which the radio is on most of the time (to receive or transmit data). Exploring the characteristics of the sensors and the targeted IoT application may achieve more savings. First, sensors work collaboratively to monitor their space, i.e., many sensors may detect the same event either due to overlapping in the sensing ranges or to a moving subject. Second, events do not occur all the time and happen only in specific periods. Some of these events are repetitive over short or long periods. Some others start in one region (a part of the network) then get spread to other regions. These two properties yield a “temporal” and “spatial” correlation between events and sensors' coverage.

A practical example is the use of motion detection sensors in smart buildings. Occupants' presence inside buildings follows repeated patterns throughout time, e.g., reaching the working spaces (e.g., offices) in the morning, having a mid-day pause, then leaving at the end of the day. Consequently, sensors at the entrance and corridors will detect more events during arrival and departure periods, whereas events are more likely to happen inside working spaces during working hours. Because the position of the sensor and the time dimension impact its readings, detecting the temporal and spatial relationship between sensors and events (by focusing on historical readings) allows predicting the future readings. This prediction allows reducing the sensing energy and the number of packets to be exchanged, thus the communication's energy. Reducing packets exchange also helps in avoiding traffic congestion and improving the quality of service in communications.

B. Paper Contribution

The current literature related to the use of historical data to reduce data sensing and exchange can be divided into two main categories: (1) approaches using statistical inference to decide which sensors to sample, (2) those approximating the data to well-know distributions and automatically deciding about the need for sampling new measurements. The first category includes solutions that use PCA (Principal Component Analysis) to select nodes responsible for sensing and transmitting (e.g., Malik at al. [2]), and those that divide the network into working and sleeping sensors and use linear transformations to estimate the sleeping sensor readings (e.g., Emekci et al. [3]). Similarly, Silvestri et al. [4] proposed heuristics to select active nodes and uses a Gaussian distribution to estimate the values of sleeping sensors. Our solution differs from solutions of the first category as it predicts readings of all sensors (instead of just a subset of them) and thus extends the saving

R. Laidi is with CERIST and Ecole Nationale Supérieure d'Informatique, Algiers, Algeria. Email: ar_laidi@esi.dz

D.Djenouri is with Computer Science Research Centre, Department of Computer Science & Creative Technologies, University of the West of England, Bristol, UK. Email: Djamel.Djenouri@uwe.ac.uk

I. Balasingham is with Norwegian University of Science and Technology (NTNU), Trondheim, Norway. Email: ilangko.balasingham@ntnu.no

potentials. Solutions in the second category targeted reducing the sampling needed to answer SQL queries. The queries include the confidence level (for estimation errors) tolerated by the user in addition to the required data. Approaches proposed by Deshpande et al. [5] and Al-Hoqani et al. [6] are examples of solutions in this category. The former uses linear regression to estimate the values of voltage and temperature sensors, while the latter uses a normal distribution to estimate temperature and light readings. These models succeed in approaching analog sensor values. However, real-time event detection is more challenging and represents the subject of this work. Moreover, we consider generating a sequence of future readings, and we are not limited to single values.

The work by Papataxiarhis and Hadjiefthymiades [7] has some similarity with the current and share the goal of predicting the following events. However, the motivation in this solution was completely different, and the authors did not consider prolonging sleep periods or reducing transmissions. They targeted event prediction in a data stream and anticipated an event using a large stream of data. This is to foresee in near future events that will affect the system's state and support system's experts in their decision-making. Moreover, the solution is also incapable of capturing long-term dependencies, i.e., correlations between events that are not sequential and thus cannot be used to provide long sleeping periods for sensors. Packets transmissions are also not reduced as the sensors keep sending their stream of data continuously. Tab.I compares the current work with the most relevant from the literature.

To deal with these problems, we explore Recurrent Neural Networks (RNN), and particularly Long Short Term Memory (LSTM), that captures spatiotemporal patterns in sequential data and use them to predict events for next time steps. We combine the LSTM architecture with a Reinforcement Learning (RL) agent. The RL agent learns the accuracy of the RNN for each situation and decides accordingly to turn sensors "on" or "off" and then start a sequence of predictions. During the decision process, the goal of the agent is to maximize energy-saving without losing accuracy.

The contribution of this paper can be summarized as follows:

- Proposing an LSTM architecture capable of predicting the future sequence of sensor events;
- Modeling the decision about sensor states in each time slot as a reinforcement learning problem to maximize power saving while preserving accuracy;
- Solving the formulated problem using an exact model-based solution, as well as and an approximated model-free solution for scalability.
- Testing the proposed model using a real publicly available dataset and compare it with four other solutions.

The remainder of the paper is organized as follows. Sec.II summarizes the related work. Sec.III presents the problem formulation and the proposed solution, while Sec.IV shows the experiment results. Sec.V discusses and analyses the results. The work is concluded, and future work is presented in Sec.VI.

TABLE I: Current contribution vs. existing solutions

Solution	Monitoring Type	Reading estimation	Energy preservation	Term
[2]–[4]	Query-based/continuous	part	part	long
[5], [6]	Query-based	all nodes	all nodes	long
[7]	Events	all nodes	no	short
Current solution	Events	all nodes	all nodes	long

II. Related Work

The battery lifespan of IoT devices remains the main obstacle for applications at large scales. Duty cycling is one of the most adopted techniques, which consists of scheduling the devices (nodes) radios, to continuously switching between active and sleeping modes while reducing the active periods to preserve energy. Several duty cycling protocols have been proposed in the literature [1]. This technique has some drawbacks, e.g., it introduces latency in communication [8], which is critical for real-time applications. It also starts reaching its limit in terms of energy preservation, which calls for alternatives for future applications. Few works, such as Liu et al. [9], [10], considered duty-cycling while focusing on the sensing coverage to ensure "frequent" coverage of the points of interest. These solutions focus on the positions of the nodes and their sensing ranges to divide the network into different sets of sensors. At every time slot, the monitoring set is selected to cover each target to the level required by the application. While this approach is more adapted to event-based applications than the previous one, it does not set the frequency of the coverage dynamically as a function of the readings of the sensors. This may lead to missing events (events occurring in inactive periods), as well as energy-saving potential (turning the sensor "on" in the absence of events).

Another approach consists in the use of energy harvesting and energy transfer technologies from the ambient resources [11], and proposing solutions based on new energy models [12]. While having the potentials for facing the finite batteries problem and shifting from fossil energy (traditional batteries) towards green resources in the future, this approach is facing technical challenges for implementing efficient hardware at the micro-scale level. Other solutions consider energy constrains for specific applications, including sensors deployment [13], crowd-sensing [14], and activity context extraction [15].

In this paper, we follow a different path and consider the prediction of sensor readings as a power-saving alternative for both the radio and the sensing device. Solutions of this category explore the advances made in merging IoT architectures with machine learning techniques [16]. In Dias et al. [17], the authors tested the feasibility of forecasting data in sensors. Experimental evaluations on four datasets showed that it is possible to reduce transmissions without affecting the data quality. Dias et al. [18] disaggregated prediction-based data reduction techniques in wireless networks into two categories, 1) "single prediction schemes", vs. 2) "dual prediction schemes". A single point of the network makes the predictions in the first category, either a sensor node or a data collection point. In the second category, network clustering is used, and both the Clusters Heads (CHs) (or the gateway) and sensor nodes simultaneously make predictions. The general idea in the second category is that the sensor nodes and the data collection point share the same prediction model and produce the same predictions. After every prediction, the sensor nodes locally check the predictions' accuracy, and the node transmits its data only if the difference between the predicted and the actual value is beneath a predefined threshold. The success of techniques that run prediction models on node levels depends highly on the node's capacity, which is limited for sensor nodes. Further, these solutions consider only communications and do not reduce energy consumption in sensing, while this may cause a significant percentage of the node's energy consumption in some applications [19]–[21].

In this paper, we are interested in reducing energy– both in sensing and communications– by predicting all sensor values in a centralized way and consequently limiting the need for physical sensing and communication to the occurrences of irregular data. In the remainder of this section, we discuss solutions where a central point holds a model of the spatiotemporal relationships between data to reduce sensing and transmission in sensor nodes.

Wei et al. [22] proposed a solution based on LSTM to predict missing data in underwater WSN, i.e., lost packets due to limitations in underwater communications. This solution inspires by a solution used for missing clinical data ([23]). The prediction model avoids *retransmissions* of lost data but does not reduce the sensing or transmission of new packets. Some other solutions exploit the spatiotemporal correlation between sensor readings to create sets of "trustful" nodes for regular sampling instead of sampling the whole network. The prediction model then uses data collected from a subset to complete the measurements of the non-sampled sensors. The Binocular framework [3] defines a scheduler to schedule sensors into "sleeping" and "working". Next, the measurements from active nodes are used by linear transformations to predict sleeping sensors' predictions. The linear model combines temporal, spatial, and spatiotemporal correlations among sensor readings. Malik et al. [2] analyzed the historical data using the feature extraction method PCA (Principal Component Analysis). This method was adopted to define for each query a subset of sensor nodes with most of the observed variance, which becomes the only sensors to send their data. The data is next treated by an association rule algorithm (a priori) to deduce the reading of the non-connected sensors.

In both Deshpande et al. [5], and Al-Hoqani et al. [6], probabilistic models were used to represent the correlation between different types of data the sensors measure and then to respond to queries. In both solutions, the authors assume sensor values follow a Gaussian distribution, and the sensors send the response only if the model's confidence is below a threshold. Deshpande et al. [5] conditioned the probability distribution on the value of a set of observed sensors to increase the answer confidence for the query. Papataxiarhis and Hadjiefthymiades [7] followed a similar approach and proposed an online technique to infer sensor measurements. Their approach is more generalized and not limited to query answering. The authors proposed three heuristics to dynamically select a set of monitors and used the correlations to estimate the remaining sensor measurements using Gaussian distributed variables. They also proposed a method to automatically detect changes in the environment, which requires a new training phase. These models based on Gaussian distributions are limited to continuous monitoring and inappropriate for event prediction, which is more complex and requires advanced knowledge about the domain [18].

Silvestri et al. [4] proposed a framework for event detection and prediction. The solution includes four steps. First, the stream of raw data is transformed into a binary stream in which the value 1 represents an event. Next, the event stream is used to generates rules about events correlation. The resulted rules are next associated with probabilities to allow events prediction. Finally, the derived rules are filtered in order to eliminate the old ones. The correlation scheme is derived using Markov models to represent frequent event patterns. The authors reported that "since these models capture sequences of events that take place in subsequent steps, they have certain limitations in representing direct dependencies among events that occur within a larger time frame." In many situations, correlated events do not happen one after the other. For example, the increase in temperature reported by the temperature sensor may be separated by several timeslots to the event of the existence of smoke that is reported by a smoke detector. Since this solution supposes that events occur in a sequence (one after another), it can not link two events spaced by several timeslots. We use this approach for comparison.

In summary, the solutions proposed in this paper fill the following gaps in the literature:

- Most duty-cycling-based approaches are limited to scheduling communication activities. Duty-cycling the whole sensor node based on such approaches is only feasible in periodic applications but inappropriate for event-based applications. If used for event-based applications, it would require keeping the sensing board on all the time while following the schedules defined by the protocol for the radios. This only reduces the energy consumed by the radio but not by the sensing. The approach proposed in this paper optimizes both sensing and communication energy usage (radio and sensing-board) for event-based applications.
- Works that consider the sensing coverage for duty-cycling do not schedule the sensors as a function of their readings, i.e., the events occurring in patterns. The solutions of this paper set the policy after learning spatiotemporal correlations from the readings, which increases energy saving potential and reduces the events missing.
- Solutions selecting monitoring sensors and dividing the network into active and sleeping nodes suffer from two problems. First, the active sensors do not benefit from any energy management, but on the contrary, their batteries get depleted very fast. Second, long sleeping periods may cause deterioration of passive sensors' batteries [24]. The RL agent proposed herein guarantees a better balance of working and sleeping periods for all the sensors. It is based on accurate sensors' reading estimation, which provides intervals for all sensors instead of targeting stopless coverage of the area.

- Using known functions (e.g., Gaussian) to approach sensor readings is proper for analog values but not for event-based applications. The proposed LSTM model is capable of learning more complicated distributions.
- Finally, the LSTM network holds on to short and long-term past information to predict several timeslots in the future. Contrary to other approaches that use only recent information.

III. Solution Description

A. Problem Statement and Solution Overview

The ultimate goal of this work is to generate a model capable of learning long/short term spatiotemporal relationships in sensorial data and predict its future values. This model enables turning the sensors off for as long as possible to preserve energy. Short term relationships characterize sequences of events that are close in time, e.g., while the occupant is working inside his office. The long term relationship detects relations between hours throughout the day, between days, seasons, or holiday periods, e.g., several timeslots may separate the increase in temperature and a smoke detection event. Spatial relationships are related to the deployment of the sensors and allow detecting events that spread through the area, e.g., a related to moving object. In this work, the events are considered as binary data. Analog sensors can also be used to report such data, e.g., the increase of temperature beyond a threshold is an indication for the existence of fire. We make abstraction of the signal processing phase throughout the paper. In particular, data is considered pre-processed and presented under a binary form.

Our architecture is composed of three main components: (1) the physical sensors, (2) the LSTM network, and (3) the RL agent. The LSTM network and the sensors work collaboratively to monitor the environment. While physical sensors are "on", the system relies on their readings. However, the LSTM output substitutes the readings as long as the sensors are turned off. The readings are predicted step-by-step, where the outputs from one timeslot are plugged as input for the next one. The process stops when the sensors are "on" again. The current work considers the sensors to always be at the same state, i.e., either all "on" or "off". The RL agent plays the supervisor's role to decide about the use of physical readings vs. LSTM estimation. It observes the LSTM model predictions and dynamically decides when turning the physical sensor on to preserve accuracy. It continuously balances two conflicting goals: power preservation and detection accuracy. Fig.1 presents the general framework of the solution. It shows 1) the training phase of the LSTM and the RL agent and 2) the deployment phase when they operate with the sensors. The process as described above represents the deployment phase. On the other hand, the training phase is composed of two steps. First, training of the LSTM to predict a sequence of sensor readings. Second, using the trained LSTM to train the RL agent. We propose two solutions for the RL, model-based and model-free. The model-based solution uses a transition table that contains the probability of transition from one state to another, and which is generated using the trained LSTM. For every state, the table provides the probability of being predicted by the LSTM. It is used by a dynamic programming algorithm to find the optimal policy. The later assigns the optimal action (on/off) for each state. The model-free solution does not require this table. It uses a deep Q-network to approximate the Q-function. The action is then selected following an ϵ-greedy strategy.

We consider a star topology that guarantees that a sensor's energy is only consumed for sensing and transmitting their readings and not for routing other sensors' packets. Therefore, all the sensors have the same state at any given time, and every sensor transmits only the data it generates. This guarantees the balance in the energy consumption between sensors. The current solution is implemented on the data collector level. Depending on the IoT architecture and the available resources, the data collector may be a base station, an intelligent-edge, or the cloud. A cluster-based topology [25] can also be considered to extend the scalability, where the cluster heads (CHs) run the solution. Rotating the CHs to balance the load (clustering protocol) is out of the scope of this work, but there are plenty of protocols that can be explored, e.g., [26]. The LSTM model and the RL agent are presented in the following.

B. LSTM Model for Predicting Sensor Readings

Fig.1 (top left side) illustrates the LSTM network architecture. Let (X, Y) be a source-target pair where, $X = x_1, x_2, x_3, \cdots x_m$, $Y = y_1, y_2, y_3, \cdots y_m$ are sequences of variables, of length m. x_i is a vector of length N that represents the sensors reading at time step i, where N is the number of sensors. The probability of the sequence $P(Y \mid X)$ (Eq.1) is decomposed using *the chain rule*.

$$\begin{aligned} P(Y \mid X) &= P(Y \mid x_1, x_2, x_3, \cdots x_m) \\ &= \prod_{i=1}^{m} P(y_i \mid y_1, y_2, y_3, \cdots y_{i-1}; x_1, x_2, x_3, \cdots x_m). \end{aligned} \tag{1}$$

In our architecture, the output of one step is the input of the next step. That is, $x_2 = y_1, x_3 = y_2, \cdots x_i = y_{i-1}$. This case is represented by Eq.(2),

$$P(Y \mid X) = \prod_{i=1}^{m} P(y_i \mid y_1, y_2, y_3, \cdots y_{i-1}; x_1). \tag{2}$$

The network estimates the probability of the next state given the input and the sequence generated so far as given by Eq.(3).

$$P(y_i \mid y_1, y_2, y_3, \cdots y_{i-1}; x_1) = P(y_i \mid y_{i-1}; h_i), \tag{3}$$

where, h_i, represents the learned context, which is the hidden state of the network at time step, i.

We use a *Softmax* layer that generates a probability distribution over discrete candidate outputs, i.e., $P(y_i = j \mid y_{i-1}; h_i) = softmax(z)j = exp(z_j)/\sum_{k=1}^{2^N} exp(z_k)$, where z is the output of the linear layer. As event-based sensors are considered, the space of sensor readings is discrete and includes all possible combinations, i.e., 2^N classes. The binary

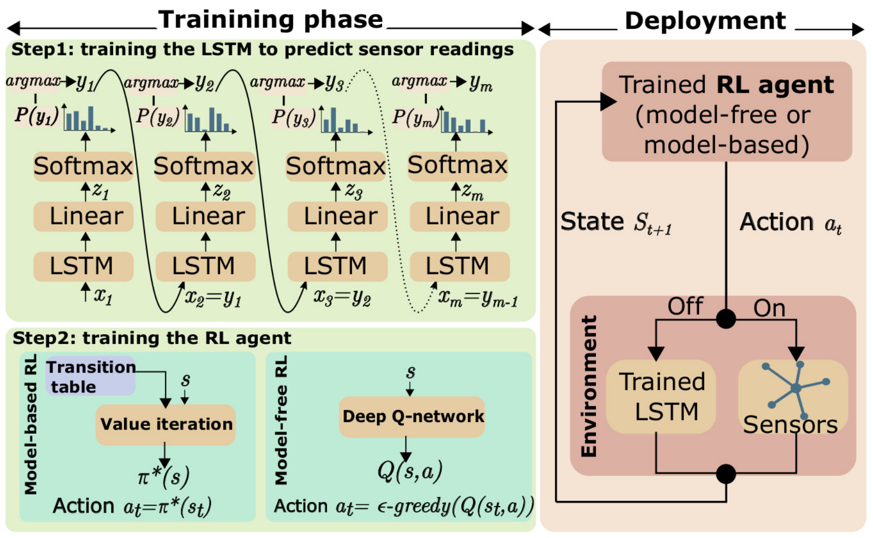


Fig. 1: General framework of the solution

vector is converted into decimal values and then into a one-hot vector[1] of size 2^N. The Softmax pushes the probability mass onto one of the 2^N classes. The output, which is also a one-hot vector of the selected class, receives the inverse operations to retrieve the vector's binary form. The probability distributions generated by the Softmax function represent the transition probabilities in the model-based RL approach (see Sec.III-C1). Note that it is possible to apply hierarchical Softmax [27], [28] for an extended number of classes– for large networks. However, this function is replaced by a Sigmoid function when the state space is vast. In this case, the model-free approach is applied, which does not require the transition probabilities.

At this step, the model can predict the sensor values for a period equal to the sequence length, i.e., m. However, the accuracy of sensors prediction is limited to the data encountered in its training phase. To effectively select the sensors on/off state in a way that ensures power-saving while preserving accuracy in estimating readings, we introduce the RL agent that orchestrates this process.

C. RL Agent for Optimal Prediction Sequence Length

1) Problem Formulation: We use an RL model in which the problem is formulated as the optimal control of incompletely know Markov Decision Process (MPD). The latter is composed of; 1) set of *states:* $s \in S$, 2) set of *actions:* $a \in A$, 3) *state transition probabilities:* $p(s'|s,a) = P\{S_{t+1} = s'|S_t = s, A_t = a\}$, 4) *expected reward*, $r(s', s, a)$, for transition from state s to s' with action a. At each time t, the agent perceives a representation of the environment's *state*, $S_t \in S$ and it selects an action $A_t \in A(s)$, for which it receives a *reward*, $R_{t+1} \in \mathbb{R}$. A reward is a numerical value sent by the environment to the agent in response to the taken action to describe how good it is.

Similarly to the LSTM modeling, the states are vectors representing all the possible values of sensors and the state space

[1]in a one-hot vector all bits are 0, except the bit representing the state

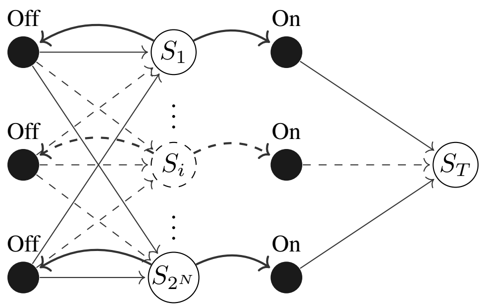


Fig. 2: Graphical model of the MDP

is discrete and includes all possible combinations, i.e., 2^N states. Besides, we add a state S_T to represent the end of the episode (sequence length). The agent reaches this state when it decides to wake the sensors up. The total number of states in our model is then $|S| = 2^N + 1$. The actions set includes two actions "on" and "off". Fig.2 graphically illustrates the MDP. The agent starts from the state representing the last value given by the sensors. If it selects action "off", the LSTM network makes a prediction, and the agent moves to another state within the set $\{S_1, ..., S_{2^N}\}$, which corresponds to the predicted value. The process continues until reaching the state S_T when deciding to turn "on" the sensors.

We define the reward as given by Eq.(4) in which the accuracy is measured by the number of correctly predicted events while the consumed energy is reflected by the number of sensors turned "on". Both are natural values between 0 and N (the number of sensors). The two values are then from the same scale. The gains G_a and G_e are respectively, the accuracy gain and the energy gain for a single sensor. They are real values that balance accuracy vs. energy usage according to the application needs. When taking "on" action (the first line of Eq.(4)), the agent receives accuracy gain proportional to the sum of detected events (i.e., number of y_i that equals 1); but it loses the energy gain for all (N) sensors. When taking "off" action, the agent receives the total gain for energy (for all nodes). The accuracy gain, in this case, depends on the accuracy of the prediction of events ($\hat{y}_i$) compared to the label y_i. G_a is added for all correct predictions (when both $\hat{y}_i$ and y_i equals 1, i.e. the second term in the second line of Eq.(4)), while $(-G_a)$ is received for all incorrect predictions (third term).

$$r_t = \begin{cases} G_a . \sum_{i=1}^{N} y_i - N.G_e, \text{if } a_t = \text{on} \\ N.G_e + G_a . \sum_{i=1}^{N} (y_i * \hat{y}_i) - G_a . \sum_{i=1}^{N} (y_i - \hat{y}_i)^2, \\ \text{if } a_t = \text{off}. \end{cases} \tag{4}$$

The transition probabilities $p(s'|s,a)$ are estimated with Eq.(5). In case the action is "on", the agent reaches the state S_T with probability 1. If the action is "off", the probability is calculated using $p(y_i \mid y_{i-1}; h_i)$, i.e., the probability estimated by the LSTM network as seen in Eq.(3) by setting $s' = y_i$ and $s = y_{i-1}$. These probabilities are the output of the trained LSTM and are saved for all states in the transition table.

$$p(s'|s,a) = \begin{cases} p(y_i \mid y_{i-1}; h_i) & \text{if } a = \text{off} \\ 1, & \text{if } s' = S_T \text{ and } a = \text{on} \\ 0, & \text{else.} \end{cases} \tag{5}$$

2) The model-based solution: The goal for an RL agent is to find the optimal action given the current state. This problem sum up to finding the optimal policy π^*, which is a mapping from the current state to the action to be taken by the agent. It is the one that maximizes the discounted, cumulative reward $R_{t_0} = \sum_{t=t_0}^{\infty} \gamma^{t-t_0} r_t$. The discount, $\gamma \in [0,1]$ makes rewards from the uncertain future less important than the near future. We define the function $Q^* : State \times Action \rightarrow \mathbb{R}$, that outputs the return of an action given a state. The optimal policy is thus defined by,

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \; Q^*_{\pi^*}(s,a). \tag{6}$$

We use *value iteration* algorithm [29] based on Dynamic Programming (DP) to solve this problem. It relays on the *Bellman equation* to find the optimal policy. The action-value function is iteratively updated, $Q_{i+1}(s,a) = \mathbb{E}[r + \gamma Q_i(s',a')]$, until convergence to the optimal action-value function,$Q_i \rightarrow Q^*, i \rightarrow \infty$. The algorithm estimates Eq.(7) to find the policy $\pi(s)$ to be optimized.

$$\begin{aligned} &\pi(s) = \underset{a}{\operatorname{argmax}} \; Q_\pi(s,a), \\ &\text{and } Q_\pi(s,a) = \sum_{s'} p(s'|s,a) \left[r + \gamma V_\pi(s') \right], \end{aligned} \tag{7}$$

where $V_\pi(s') = \sum_a \pi(s,a) Q_\pi(s,a)$ is the value function of state s' under the policy π, calculated by the algorithm. It is the expected outcome when the agent starts from state s' and follows the policy π. The value γ is the *discount factor*, r is the reward received from the environment measured using Eq.(4), and $p(s'|s,a)$ is given by Eq.(5).

3) The model-free solution: DP algorithms scale in time $O(|S|^2)$ [30], and the transition table capturing the probabilities is of size $|S|^2$. Nevertheless, as the number of states is proportional to the number of sensors in our model, the scalability of the model-based approach becomes a problem. That approach can be useful for application with limited, e.g., occupancy monitoring in small spaces where few sensors are used, but not in areas with tens of sensors. To tackle this problem, we propose a model-free solution based on the *Deep Q-Network (DQN)* algorithm [31].

Instead of iteratively estimating $Q(s,a)$, a function approximator is used to estimate the action-value function, $Q(s,a;\theta) \approx Q^*(s,a)$. The Q-network we use is a neural network function approximator with weights θ, which tries to minimize the loss function defined in Eq.(8),

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim U(D)} [(\underbrace{r + \gamma \max_a Q(s',a';\theta_{i-1})}_{\text{target}} - \underbrace{Q(s,a;\theta_i)}_{\text{prediction}})^2], \tag{8}$$

where θ is the weights vector of the network, and $U(D)$ is the experience replay history. Notice the Q-Network also keeps track of some observation in a $memory$ for training the network (the technique is known as *replay memory* technique). The network predicts the expected return of taking every action given the current state. We define a neural network in which the inputs are the outputs of the LSTM, while the outputs are the Q values of the actions.

In this approach, the states are presented as binary vectors of size N instead of one-hot vectors of size 2^N like in the model-based one. It implies that the states set increases linearly (not exponentially), which enables higher scalability.

In the original work of mnih et al. [31], the authors defined a Convolutional Neural Network (CNN) in order to extract the current state from images. However, in our case, the states are vectors of size N. We hence define a standard deep neural network composed of a linear input layer of size N and a linear output layer of size 2 (the size of action space). The hidden layers are two *ReLU* (Rectifier Linear Units) layers separated by a fully connected linear layer with fixed sizes defined as a hyperparameter. Fig.3 shows this architecture.

The action selection follows the ε-greedy method: most of the time, it selects the action greedily maximizing $Q(s,a)$, but with small probability, ε, it randomly selects an action (with equal probability among all the possible actions). To encourage the exploration in the first episodes of learning, and then the exploitation, we use a weight decay strategy for ϵ values. We set a multiplicative factor per episode for decreasing epsilon by 0.995 after each episode. Minimal and maximal values of ϵ are respectively 0.01 and 1.

D. Baseline Derivation for the RL Agent

In the following, we define the baseline (lower bound in energy consumption, and upper bound in precision) for the RL. Let x be a binary vector presenting optimal decision through t

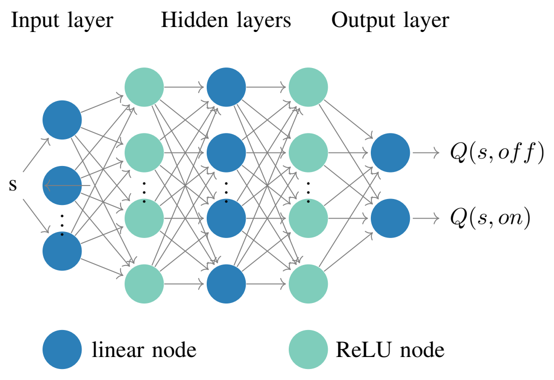


Fig. 3: Architecture of the Q-network

TABLE II: Simulation hyperparameters

Component	Hyperparameter	Settings		
Network Architecture and Training	Network type	LSTM	DNN	DQN
	Learning rate	19.10^{-3}	1.10^{-4}	5.10^{-4}
	Batch size	160	1024	64
	Weight decay	1.10^{-5}	NA	NA
	Hidden layer size	480	512	64
	# layers	2	2	2
	Seq lenght	128	NA	NA
DQN Optimisation	Discount rate γ	0.9		
	Steps before update	4		
	Replay buffer size	10^5		
ϵ-greedy Exploration	Initial ϵ value	1		
	ϵ Decay factor	0.995		
	Minimum ϵ Value	0.01		

time steps. $x_i = 1$ implies the action 1 is optimal for the time step i. Optimal values of this vector are obtained by solving the optimization problem in Eq.(9). The problem minimizes the number of active timeslots while keeping prediction error below a threshold S. This problem is transformed into Eq.(10), which is an integer programming problem equivalent to the famous *Knap sac* optimization problem that can be solved using a dynamic programming algorithm. Notice that the predicted and the ground truth data are both required to solve this problem. Since the ground truth data is not available in practice, this solution serves only as a baseline for the tests to evaluate the solutions proposed earlier for the RL actions.

$$\text{minimize} \sum_{j=1}^{t} x_j$$
$$\text{s.t.} \sum_{j=1}^{t}(1-x_j)\sum_{i=1}^{N}(y_i^{<j>} - \hat{y}_i^{<j>})^2 \leq S \text{ and } x_j \in \{0,1\}. \tag{9}$$

We notate $z_j = 1 - x_j$ and $w_j = \sum_{i=1}^{N}(y_i^{<j>} - \hat{y}_i^{<j>})^2$, Eq.(9) is the transformed into,

$$\begin{aligned} &\text{maximize} \sum_{j=1}^{t} z_j \\ &\text{s.t.} \sum_{j=1}^{t} w_j z_j \leq S \text{ and } z_j \in \{0,1\}. \end{aligned} \tag{10}$$

IV. Experiments and Performance Evaluation

The following experiments compare the model-based and model-free approaches, respectively LSTM-DP (DP for Dynamic Programming) and LSTM-DQN (DQN for Deep Q-network), with two state-of-the-art solutions:

- JGD [7] (*"Jointly Gaussian Distribution"*): it uses a heuristic to split the network's nodes into two groups, "monitoring" sensors, and "sleeping" sensors. Data from monitoring sensors are used to estimate the readings of the sleeping sensors. It is supposed that sensor readings fellow a Gaussian distribution.
- EC [4] (*"Event Correlation"*): it proposes a stepwise approach. First, each sensor's stream is transformed into a binary stream. Then, a tree from events correlation is generated, from which rules are then used to make event prediction using *probabilistic temporal logic*.

We are also interested in investigating the impact of learning the spatiotemporal correlations on the model's performance. To this end, we compare it against a DNN (Deep Neural Network) variant which has no access to previous inputs. The difference between the two models shows the advantage brought by learning spatial and temporal patterns in sensor readings. To this purpose, we create two other variants, DNN-DP and DNN-DQN, by replacing the LSTM cells with a DNN (in the architecture illustrated in Fig.1, top left side). It receives a vector of sensors state at time, t, and returns the estimated vector for time $t+1$. The hidden layers are similar to the ones in Fig.3. Finally, the comparison also includes a baseline that is used for the RL agent's performance evaluation presented in Sec.III-D, noted LSTM-IP (IP for Integer Programming). We selected the smart building context as the case study for the evaluation, which is one of the emerging IoT applications calling for the usage of advanced machine learning tools [32], [33]. The *MERL* [34] real dataset is used in this study, which has been collected in a laboratory for the test. It includes over 50 million motion sensor events spanning two years, with milliseconds granularity. The models have been implemented in a GPU environment using Pytorch [35] deep learning framework.

We consider a star topology in the simulations, i.e., all sensors directly connect to a data collector. The latter has limitless energy and enough calculation capacities to run deep learning models. Wake up/sleeping mechanism consumes neglectable energy and occurs within a short time. Tab.II shows the settings and the training hyperparameters considered during the simulation.

A. Prediction Accuracy

We varied the ratio of activation time from 0 to 1 (with 0.01 granularity) and measured the accuracy. The former is the ratio of the number of active time slots to the total number of slots. In the presented solution, it is controlled using G_a and G_e parameters in the reward function (Eq.(4)). A similar definition for the ratio of activation time is used for *"EC"* [4]. In *"JGD"* [7], it is defined as the ratio between the number of sensors left "on" to the total number of sensors.

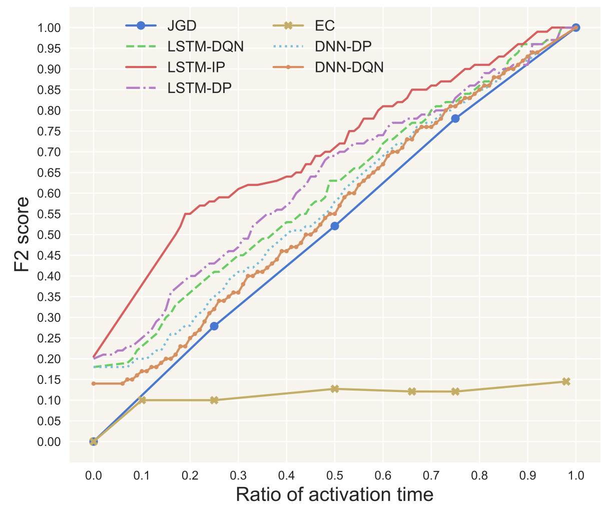


Fig. 4: F2-score on the test set for 4 sensors as a function of the ratio of energy

We dedicate 10% of dataset for the test, which is a period of 1.04×10^6 *seconds*. For each value of activation time, the average accuracy is calculated for the whole sequence. The accuracy metric is the *F2-score* [36], which is the harmonic weight of the recall and the precision that weights the recall higher. Precision may be defined as the likelihood that an event is relevant, given that it is predicted by the system, while the recall is the likelihood of correctly predicting events.

Fig.4 illustrates the results in a network of 4 sensors. It is clear that the proposed solutions (both model-based and free) outperform the two state-of-the-art solutions (JGD and EC). The proposed solutions achieve about 50% of accuracy while being active 30% of the time, 72% of accuracy with 50% of activation time. They achieve 20% of average accuracy without turning the sensors "on" for more than 27 consecutive hours (the length of the test set). JGD has an increasing and better performance than EC since it relies on the part of the network that is awake. The results confirm that EC is incapable of learning long term dependencies and that even with more data, the prediction performance remains stable. These results are in line with those reported in [4].

Notice also that the difference between the optimal decision (LSTM-IP) and the proposed approaches is more visible when less energy is consumed, but this difference gradually reduces with the increase of the consumed energy. The results also show that LSTM-based variants perform better than DNN-based variants, which confirms our first intuition about the need for learning spatiotemporal dependencies. The model-based approach has a slight advantage compared to the model-free approach. This can be justified by the fact that the former is based on the full knowledge of the environment.

B. Energy Consumption

We measured the average energy in *watt second* consumed by one sensor while varying the accuracy rates. We used as reference a hardware manufacturer datasheets of a wireless

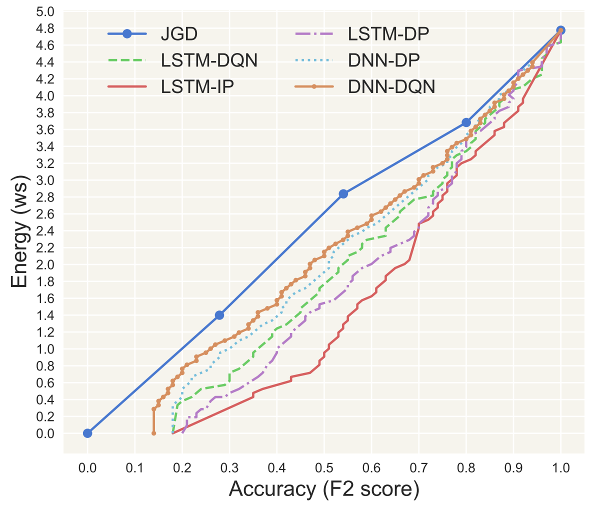


Fig. 5: Average consumed energy (w.s) by one sensor as a function of the F2-score

motion detector. The mote is powered with a *CR2032* coin cell battery ($3V$) of 240 mAh lifetime. The entire system, including the CPU, consumes 1.57 mA in active mode for 56.66 ms, 3.45 μA in standby mode, and 2.16 μA in shutdown [37]. Eq.(11) shows the power measurement p in one second. The energy is then derived using the equation $E = p \times t$, and $p = I \times V$.

$$p = \underbrace{[(1.57 * 56.66.10^{-3} + 3.45.10^{-3} * 0.94334) \sum_{i=1}^{N} y_i +}_{\text{event}} \underbrace{2.16..10^{-3}.(N - \sum_{i=1}^{N} y_i)] \times 3}_{\text{no event}} \quad (11)$$

We compare the energy consumed by LSTM-IP, LSTM-DQN, DNN-DQN, LSTM-DP, DNN-DP, and JGD [7]. EC [4] is not included for this metric since it has stable accuracy levels. Fig.5 shows the results that are in line with the previous ones, i.e., the LSTM-DP reaches better energy-saving followed by the LSTM-DQN, and both outperformed JGD [7].

The same energy model was used to measure the battery's lifetime using the Eq.(12).

$$\text{Battery Life} = \frac{\text{Battery Capacity (mAh)}}{\text{Load Current (mA)}} \times \text{Derating factor}, \quad (12)$$

where the derating factor depends on external factors that affect batteries' lifetime. Fig.6 shows the results and confirms that the proposed solutions improve the battery lifetime, which becomes almost optimal for F2-score values beyond 0.7.

C. Scalability

We varied the number of sensors to investigate the scalability of both the LSTM network and the RL agent. The

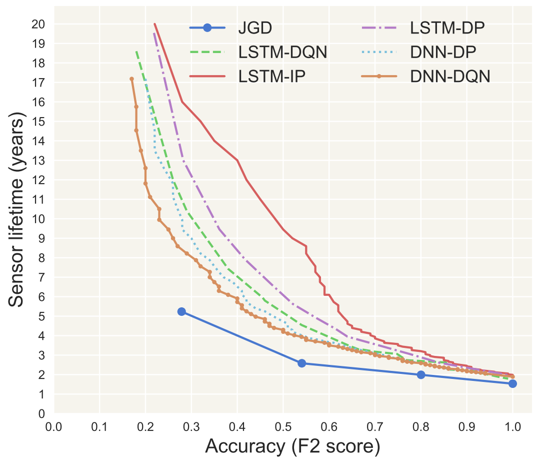


Fig. 6: Average one sensor battery lifetime (year) as a function of the F2-score

activation time was set to 50%, which represents the average usage of the LSTM. We used the range $\{4, 8, 16, 29\}$ sensors. 29 was the maximum number of sensors deployed in the used dataset within a communication range of 10 m [38] while considering the star topology. The purpose of considering a $10m$ power range was to increase the number of sensors and investigate the scalability. However, realistic range is typically shorter due to physical constraints in indoor environments. We evaluate the performance of LSTM-IP, LSTM-DQN, DNN-DQN, LSTM-DP, DNN-DP, JGD [7], and EC [4] according to five metrics: F-score, the average energy (in ws) consumed by an active sensor, the lifetime of the network in years, the latency (in ms), and the ratio of missed events. The network lifetime is the average lifetime of all the sensors–active and inactive confined. LSTM-IP is not tested for latency because it is a theoretical baseline and does not run in a real scenario. We noticed that for more than 8 sensors, the Softmax performance drops. Therefore, it has been replaced with a Sigmoid function. A weighted Sigmoid loss was used to penalize "false-negatives". Furthermore, LSTM-DP evaluation is limited to 8 sensors, as it uses transition probabilities generated by the Softmax function.

Fig.7 shows that extending the network's size increases the F2-score and reduces the ratio of missed events. It also reduces energy consumption and elongates the lifetime. This endorses the role of learning spatial correlations in improving the LSTM prediction. On the other hand, increasing the number of sensors rises the latency. Notice that for JGD, the network's lifetime increases despite the increase on the average consumed energy of active sensors. The solution does not balance the load between sensors and provides a false impression of elongating the network's lifetime (average lifetime for active and inactive sensors) while draining the energy of the most valuable sensors.

We tabulated data values of all the results in the supplementary material.

V. Analysis and Discussion

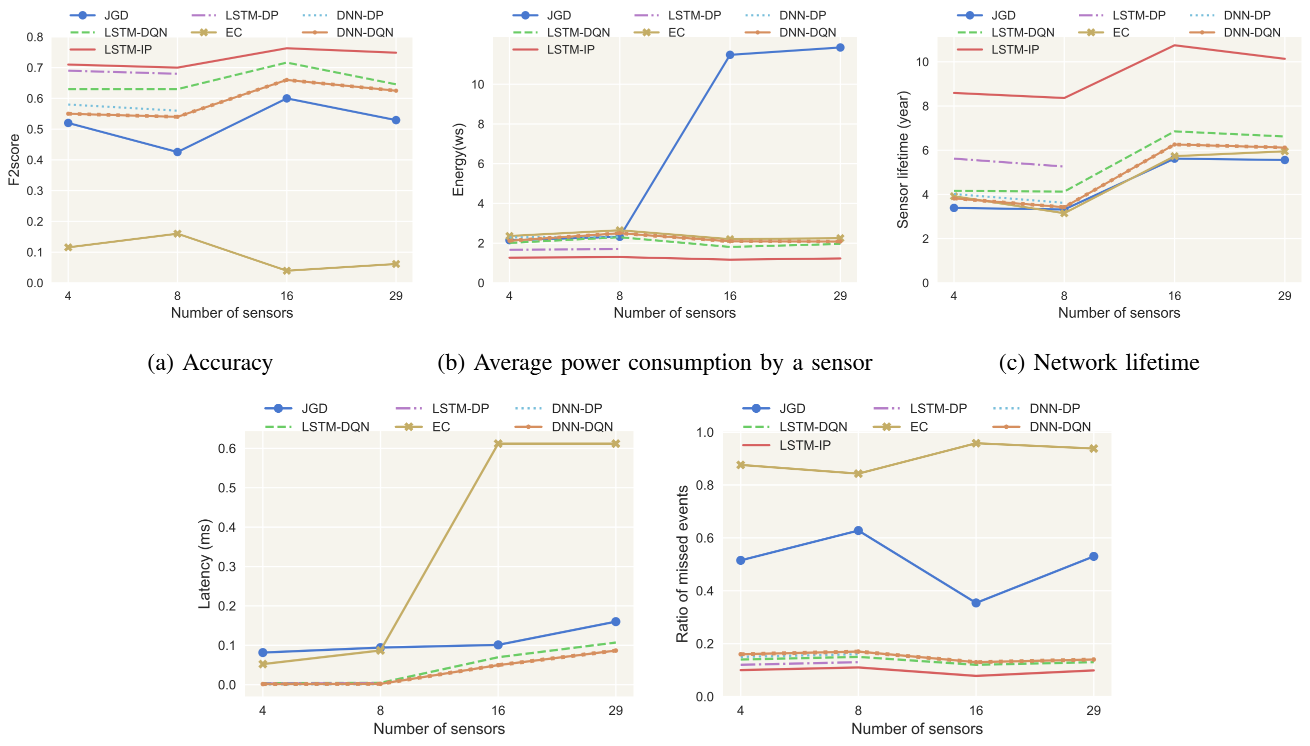
A. *Novelty and Improvement:*

Unlike traditional methods, this solution reduces the energy spent for both sensing and communications as it predicts sensor readings and allow long sleep periods for both the radio and sensing-board. We focused on event-based applications that are more challenging than periodic monitoring in which continuous functions can approach the real readings. The solution uses sequence modeling with LSTM to learn spatiotemporal correlation, while an RL agent balances the accuracy/energy. The prediction is for all the sensors and possible for several future timeslots. It achieved better results than JGD [7] that suffers from two problems (similarely to many state-of-the art approaches). First, it only save the less informative sensors' energy while putting all the network's charge on the most indicative sensors. Consequently, the latter do not benefit from sleeping periods. Second, long periods of battery disuse of sleeping nodes leads to battery self-discharge and deterioration [24]. Therefore, besides the reported improvement, the proposed solution provides a better load balancing between sensors and better network management.

Our solution also outperformes EC [4], which is a solution designed to foresee events in a large stream of data. EC holds on short term data (data from few prior seconds) to make the prediction. The test results show that this is insufficient. The proposed LSTM approach can learn short and long term temporal and spatial patterns in the data through long periods. The decrease of performance after replacing the LSTM by a DNN mainly shows the benefit of learning spatiotemporal correlations. On the other side, we tested the model-based and model-free RL agents' performances vs. the optimal decision (based on knapsack problem solver). The results show the proposed approaches for RL are close to optimal and converge with the increasing in power usage. The model-based agent tends to be more accurate that the model-free, which is more scalable.

B. *Complexity*

The overall complexity is the summation of the costs related to: i) the forward pass of the LSTM model, ii) and the RL agent algorithm. Hochreiter and Schmidhuber [39] show that the LSTM complexity in time and space is equal to $\mathcal{O}(\omega)$, where ω is the number of the LSTM weights. It depends on the number of output units, the number of memory cell blocks and their size, the number of hidden units, the number of forward units that are connected to memory cells, gate units, and hidden units. For the model-based RL agent, DP algorithms scale in time $\mathcal{O}(|S|^2)$ [30], and the transition table is of size $|S|^2$, where S is the size of the state space. On the other side, the complexity of the model-free solution is equivalent to the forward pass of the Q-network. This is equal to $\Theta\left(nh_1 + h_L u + \sum_{i=1}^{L-1} h_i h_{i+1}\right)$, for a network of n inputs, u output neurons, and L hidden layers, where the $i^t h$ hidden layer contains h_i hidden neurons.



(a) Accuracy (b) Average power consumption by a sensor (c) Network lifetime

(d) Latency (e) Missed events ratio

Fig. 7: Scalability tests

C. Scalability

The results reveal that extending the network's size increases the accuracy. They also confirms that, besides the temporal correlations, spatial correlations between sensors enhance the LSTM's predictions. On the other hand, increasing the number of sensors increases the computational time. This raises three issues that should be considered in future work. First, replace traditional LSTM by Convolutional LSTM [40], which has better performance in tackling spatiotemporal sequence forecasting problems. Second, appraise the improvement of a cluster-based topology on computational efficiency, where cluster-heads can perform prediction in parallel. Third, consider sensors' optimal deployment [41] to divide the sensors into independent clusters without losing spatial correlation information. This can potentially decrease the training time as well.

D. Non-Stationarity and Unfamiliar Cases

In this work, we assumed a stationary environment, i.e., the environment's rules (the state transition probabilities and reward distributions) do not change over time. However, the Q-learning algorithm considered in the model-free approach is an online RL algorithm that can adjust policies to match non-stationary environments when the changes do not happen too often. Reinforcement learning in highly dynamic environments is a more challenging problem and a research trend that we consider tackling in future work. For data non-stationarity, we can consider re-training the system periodically. Part of the future work is also to set a mechanism for detecting changes and launching a new training phase whenever the old models do not match the new data. To guarantee the system's robustness against infrequently faced states (during the training phase) we avoided overfitting, i.e., the LSTM properly generalizes to states outside the training set. Besides, the improvement brought by including the RL agent proves that the latter learns to distinguish unfamiliar cases for the LSTM.

E. Discount Factor Values

Common discount factor values in literature are between 0.9 and 0.99 [31], [42], [43]. We noticed that varying values between 0.9 and 0.99 has no big impact on the policy's performance in our case. For values below 0.5, the agent performs poorly and the policy is either always "on" or always "off". We accordingly set the discount factor to 0.9.

VI. Conclusion and Future Work

A new approach has been proposed in this paper that enables accurate prediction of sensor readings in event-based IoT applications. It allows to turn "off" sensors in periods when their readings may be predicted and thus preserving the energy consumed for sensing and communications. The solution uses an LSTM network for the prediction and an RL agent for accuracy-energy balancing. We evaluated the solutions using real datasets and found that the LSTM model was capable of learning spatiotemporal dependencies better than a finely tuned DNN and statistical approaches. On the other hand, the RL agent was able to achieve near-optimal performances for both the model-based and the model-free solution.

This work takes a step toward jointly applying deep learning and reinforcement learning to face one of IoT limitations; battery lifetime of the devices (things). To our knowledge, this is the first that considers predicting sensor readings to reduce events sensing and transmissions while setting a mechanism to preserve accuracy. A principal direction for future work is to investigate the solutions' scalability on a high number of sensors. One option would be to divide the sensors into independent clusters. We also plan to propose a solution that considers different states for the sensors at the same timeslot.

VII. Acknowledgement

This work has been carried out in part at CERIST research center Algeria, in part at UWE Bristol, UK, and in part at NTNU Trondheim, Norway. This project is also partly supported by the Arab-German Young Academy of Sciences and Humanities (AGYA) that has been funded under the German Ministry of Education and Research (BMBF) grant 01DL16002.

References

[1] R. C. Carrano, D. Passos, L. C. S. Magalhaes, and C. V. N. Albuquerque, "Survey and taxonomy of duty cycling mechanisms in wireless sensor networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 181–194, First 2014.

[2] H. Malik, A. S. Malik, and C. K. Roy, "A methodology to optimize query in wireless sensor networks using historical data," *Journal of Ambient Intelligence and Humanized Computing*, vol. 2, no. 3, p. 227, Jun 2011. [Online]. Available: https://doi.org/10.1007/s12652-011-0059-x

[3] F. Emekci, S. E. Tuna, D. Agrawal, and A. E. Abbadi, "Binocular: A system monitoring framework," in *Proceeedings of the 1st International Workshop on Data Management for Sensor Networks: In Conjunction with VLDB 2004*, ser. DMSN '04. New York, NY, USA: ACM, 2004, pp. 5–9. [Online]. Available: http://doi.acm.org/10.1145/1052199.1052201

[4] S. Silvestri, R. Urgaonkar, M. Zafer, and B. J. Ko, "A framework for the inference of sensing measurements based on correlation," *ACM Trans. Sen. Netw.*, vol. 15, no. 1, pp. 4:1–4:28, Dec. 2018. [Online]. Available: http://doi.acm.org/10.1145/3272035

[5] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," in *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, ser. VLDB '04. VLDB Endowment, 2004, pp. 588–599. [Online]. Available: http://dl.acm.org/citation.cfm?id=1316689.1316741

[6] N. Al-Hoqani, S. Yang, and D. P. Fiadzeawu, "Probability model-based on-demand query processing for wireless sensor networks database," in *2017 23rd International Conference on Automation and Computing (ICAC)*, Sep. 2017, pp. 1–6.

[7] V. Papataxiarhis and S. Hadjiefthymiades, "Event correlation and forecasting over high-dimensional streaming sensor data," in *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct 2018, pp. 1–8.

[8] M. Doudou, D. Djenouri, and N. Badache, "Survey on latency issues of asynchronous mac protocols in delay-sensitive wireless sensor networks," *IEEE Communications Surveys Tutorials*, vol. 15, no. 2, pp. 528–550, Second 2013.

[9] K. S. Liu, T. Mayer, H. T. Yang, E. Arkin, J. Gao, M. Goswami, M. P. Johnson, N. Kumar, and S. Lin, "Joint sensing duty cycle scheduling for heterogeneous coverage guarantee," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, May 2017, pp. 1–9.

[10] K. S. Liu, J. Gao, S. Lin, H. Huang, and B. Schiller, "Joint sensor duty cycle scheduling with coverage guarantee," in *Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '16. New York, NY, USA: ACM, 2016, pp. 11–20. [Online]. Available: http://doi.acm.org/10.1145/2942358.2942379

[11] F. Engmann, F. A. Katsriku, J.-D. Abdulai, K. S. Adu-Manu, and F. K. Banaseka, "Prolonging the lifetime of wireless sensor networks: A review of current techniques," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[12] M. Khiati and D. Djenouri, "Adaptive learning-enforced broadcast policy for solar energy harvesting wireless sensor networks," *Computer Networks*, vol. 143, pp. 263–274, 2018. [Online]. Available: https://doi.org/10.1016/j.comnet.2018.07.016

[13] Nojeong Heo and P. K. Varshney, "Energy-efficient deployment of intelligent mobile sensor networks," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 35, no. 1, pp. 78–92, Jan 2005.

[14] L. Wang, D. Zhang, Z. Yan, H. Xiong, and B. Xie, "effsense: A novel mobile crowd-sensing framework for energy-efficient and cost-effective data uploading," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 12, pp. 1549–1563, Dec 2015.

[15] H. Park, J. Park, H. Kim, J. Jun, S. Hyuk Son, T. Park, and J. Ko, "Relisce: Utilizing resource-limited sensors for office activity context extraction," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 8, pp. 1151–1164, Aug 2015.

[16] F. Samie, L. Bauer, and J. Henkel, "From cloud down to things: An overview of machine learning in internet of things," *IEEE Internet of Things Journal*, pp. 1–1, 2019.

[17] G. M. Dias, B. Bellalta, and S. Oechsner, "On the importance and feasibility of forecasting data in sensors," *CoRR*, vol. abs/1604.01275, 2016. [Online]. Available: http://arxiv.org/abs/1604.01275

[18] ——, "A survey about prediction-based data reduction in wireless sensor networks," *ACM Comput. Surv.*, vol. 49, no. 3, pp. 58:1–58:35, Nov. 2016. [Online]. Available: http://doi.acm.org/10.1145/2996356

[19] C. Alippi, G. Anastasi, M. Di Francesco, and M. Roveri, "Energy management in wireless sensor networks with energy-hungry sensors," *IEEE Instrumentation & Measurement Magazine*, vol. 12, no. 2, pp. 16–23, 2009.

[20] M. Razzaque and S. Dobson, "Energy-efficient sensing in wireless sensor networks using compressed sensing," *Sensors*, vol. 14, no. 2, pp. 2822–2859, 2014.

[21] V. Raghunathan, S. Ganeriwal, and M. Srivastava, "Emerging techniques for long lived wireless sensor networks," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 108–114, 2006.

[22] X. Wei, Y. Liu, S. Gao, X. Wang, and H. Yue, "An rnn-based delay-guaranteed monitoring framework in underwater wireless sensor networks," *IEEE Access*, vol. 7, pp. 25 959–25 971, 2019.

[23] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific reports*, vol. 8, no. 1, p. 6085, 2018.

[24] X. Fan, X. Liu, W. Hu, C. Zhong, and J. Lu, "Advances in the development of power supplies for the internet of everything," *InfoMat*, vol. 1, no. 2, pp. 130–139, 2019.

[25] W. B. Heinzelman, "Application-specific protocol architectures for wireless networks," Ph.D. dissertation, Massachusetts Institute of Technology, 2000.

[26] N. Merabtine, D. Djenouri, D.-E. Zegour, B. Boumessaidia, and A. Boutahraoui, "Balanced clustering approach with energy prediction and round-time adaptation in wireless sensor networks," *International Journal of Communication Networks and Distributed Systems*, vol. 22, no. 3, pp. 245–274, 2019.

[27] A. Mnih and G. E. Hinton, "A scalable hierarchical distributed language model," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2009, pp. 1081–1088. [Online]. Available: http://papers.nips.cc/paper/3583-a-scalable-hierarchical-distributed-language-model.pdf

[28] A. A. Mohammed and V. Umaashankar, "Effectiveness of hierarchical softmax in large scale classification tasks," *CoRR*, vol. abs/1812.05737, 2018. [Online]. Available: http://arxiv.org/abs/1812.05737

[29] R. Sutton, A. Barto, and F. Bach, *Reinforcement Learning: An Introduction*. MIT Press, 2018. [Online]. Available: https://books.google.dz/books?id=sWV0DwAAQBAJ

[30] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, April 1967.

[31] 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, p. 529, 2015.

[32] D. Minoli, K. Sohraby, and B. Occhiogrosso, "Iot considerations, requirements, and architectures for smart buildings—energy optimization and next-generation building management systems," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 269–283, Feb 2017.

[33] D. Djenouri, R. Laidi, Y. Djenouri, and I. Balasingham, "Machine learning for smart building applications: Review and taxonomy," *ACM Comput. Surv.*, vol. 52, no. 2, pp. 24:1–24:36, Mar. 2019. [Online]. Available: http://doi.acm.org/10.1145/3311950

[34] C. Wren, Y. Ivanov, D. Leigh, and J. Westhues, "The merl motion detector dataset," in *Workshop on Massive Datasets (MD)*, Nov. 2007, pp. 10–14. [Online]. Available: https://www.merl.com/publications/TR2007-069

[35] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[36] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and f-score, with implication for evaluation," in *Advances in Information Retrieval*, D. E. Losada and J. M. Fernández-Luna, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 345–359.

[37] "Ultra-low-power wireless pir motion detector for cost-optimized systems reference design," https://www.ti.com/lit/ug/tiducu5/tiducu5.pdf, accessed: 2019-08-06.

[38] I. Kuzminykh, A. Snihurov, and A. Carlsson, "Testing of communication range in zigbee technology," in *2017 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM)*. IEEE, 2017, pp. 133–136.

[39] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[40] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[41] R. Laidi and D. Djenouri, "Udeploy: User-driven learning for occupancy sensors deployment in smart buildings," in *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, March 2018, pp. 209–214.

[42] G. Palmer, K. Tuyls, D. Bloembergen, and R. Savani, "Lenient multi-agent deep reinforcement learning," *arXiv preprint arXiv:1707.04402*, 2017.

[43] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3389–3396.