An Analysis of Double *Q*-Learning-Based Energy Management Strategies for TEG-Powered IoT Devices

Michal Prauzek, *Senior Member, IEEE*, Jaromir Konecny, *Member, IEEE*, and Tereza Paterova

***Abstract*—The study presents a self-learning controller for managing the energy in an Internet of Things (IoT) device powered by energy harvested from a thermoelectric generator (TEG). The device's controller is based on a double *Q*-learning (DQL) method; the hardware incorporates a TEG energy harvesting subsystem with a dc/dc converter, a load module with a microcontroller, and a LoRaWAN communications interface. The model is controlled according to adaptive measurements and transmission periods. The controller's reward policy evaluates the level of charge available to the device. The controller applies and evaluates various learning parameters and reduces the learning rate over time. Using four years of historical soil temperature data in an experimental simulation of several controller configurations, the DQL controller demonstrated correct operation, a low learning rate, and high cumulative rewards. The best energy management controller operated with a completed cycle and missed cycle ratio of 98.5%. The novelty of the presented approach is discussed in relation to state-of-the-art methods in adaptive ability, learning processes, and practical applications of the device.**

***Index Terms*—Energy harvesting, energy management, Internet of Things (IoT), reinforcement learning, thermoelectric generator (TEG).**

I. Introduction

THE APPLICATION of machine learning (ML) methods in combination with embedded Internet of Things (IoT) devices remains a challenging task due to the limited computational resources, low-power demands, and self-operating requirements of these devices. This study is an extended version of a pilot study [1] presented in the 2022 IEEE Symposium Series on Computational Intelligence and delivers a more detailed and complex analysis of *Q*-learning (QL) performance, an improved reward policy, and a double QL (DQL) policy tested over four years.

Manuscript received 27 October 2022; revised 21 April 2023 and 18 May 2023; accepted 5 June 2023. Date of publication 7 June 2023; date of current version 24 October 2023. This work was supported in part by the "Development of Algorithms and Systems for Control, Measurement and Safety Applications IX" of the Student Grant System, VSB-TU Ostrava under Project SP2023/009, in part by the "Development of a System for Monitoring and Evaluation of Selected Risk Factors of Physical Workload in the Context of Industry 4.0" of the Technology Agency of the Czech Republic under Project FW03010194, and in part by the European Union's Horizon 2020 Research and Innovation Programme under Grant 856670. *(Corresponding author: Michal Prauzek.)*

The authors are with the Department of Cybernetics and Biomedical Engineering, VSB—Technical University of Ostrava, 70800 Ostrava-Poruba, Czechia (e-mail: michal.prauzek@vsb.cz; jaromir.konecny@vsb.cz; tereza.paterova@vsb.cz).

Digital Object Identifier 10.1109/JIOT.2023.3283599

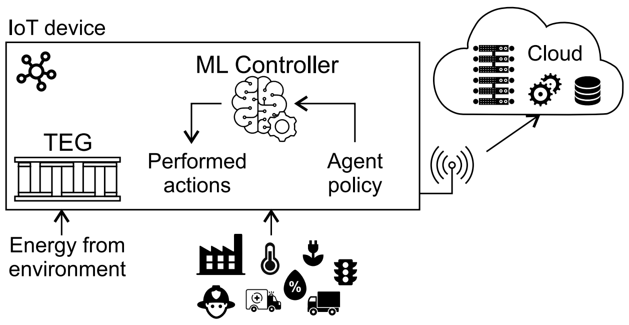

Fig. 1. Energy management principle applied by the ML controller in the energy harvesting TEG-powered IoT device.

The study investigated methods of powering IoT platforms with thermometric generators [thermoelectric generator (TEG)] [2] according to the scheme depicted in Fig. 1. The amount of energy harvested by a TEG is a dynamic parameter which depends on temperature in the surrounding environment [3]. An energy management system which specifies how an IoT device should behave at certain times should, therefore, be applied according to the energy which is available to the device [4]. The controller described in the study applied a DQL-based strategy to manage the duty cycle in a TEG-powered IoT device. The device itself was designed to monitor environmental parameters and transmit collected data via a wireless communications interface for storage in a cloud and subsequent advanced data processing.

The algorithm used by the controller applied real-time self-learning principles. The energy harvesting IoT sensor, therefore, did not require any energy system or hardware customization or modification (capacitor size, energy harvester type, replacement of aging hardware, etc.) to suit the device's application or deployment location. Self-learning also solved the many disadvantages of state-of-the-art methods, for example, the need for historical data to train a neural network, time consuming, and computationally extensive processes to optimize a fuzzy-based controller, or the need to predict ambient energy in a prediction-based controller. The proposed DQL implementation in a low-cost embedded IoT platform is very effective and has low computational and memory requirements in combination with the reinforcement learning algorithm.

This feature is designed for very demanding, low-cost, and low-power designs.

The study's contribution is summarized in the following.

1) A novel, self-learning DQL-based approach designed for low-power, low-cost TEG-powered IoT nodes managed with a wake-up scenario.
2) Comparison of the proposed solution's performance with a static energy management configuration and a state-of-the-art fuzzy-based controller tested with a simulation and soil environmental data.
3) Discussion of the features of the proposed approach in relation to the results obtained from the device's self-learning ability, model-free design, and computational cost requirements.

The article is organized as follows. Section I presents the aim, novelty, and benefits of the study; Section II summarizes related studies and the state-of-the-art; Section III describes the device's DQL principles, learning policy, and adapted DQL energy management algorithm; Section IV describes the study's experiment and input data and provides an evaluation of the device's performance; Section V discusses the experimental results in relation to the device's learning parameters, a time domain analysis, and comparison with a reference solution; Section VI discusses the results and the article's contribution; and Section VII concludes this article and outlines potential future work.

II. Related Works

Energy management policies in energy harvesting IoT sensors are designed to provide a continuous and uninterrupted supply of energy [5]. Due to the unpredictable and dynamic nature of the harvesting environments, successful operation of adaptive energy management algorithms in IoT sensors remains a challenge [6]. For adaptive energy management, ML methods can be applied. Table I summarizes the current research and state-of-the-art ML methods. These methods are categorized into offline and online learning approaches. Offline methods exploit a neural network or fuzzy logic to predict energy management system parameters. Online methods are based on self-learning algorithms, such as deep reinforcement learning or QL.

Neural networks are mainly used for predictive analysis. In neural network applications, the quantity of available energy can be predicted from energy harvesting nodes [7] or multiple energy harvesting sources [8]. Output power can be predicted from hybrid energy harvesting sources [9]. The high computational demands of neural networks mean it is not always feasible to deploy this approach with energy-constrained devices [23].

Fuzzy logic is a suitable method for building adaptive algorithms that are used to achieve a continuous energy source for sensor nodes and, thus, prolong sensor node lifetime [10]. Genetic algorithms are applied to optimize fuzzy rule-based controllers [11], forecast next-day solar energy availability using evolutionary fuzzy rules [12], or predict the quantity of available energy in IoT devices [13]. Algorithms based on fuzzy logic can also be applied to manage the operation of wireless sensor nodes equipped with energy harvesting devices [15]. Systems based on fuzzy logic can provide optimal operational strategies that assess the node's current resource requirements, current battery status, or expected energy charge [14]. However, because neural networks and fuzzy rule-based systems do not possess self-learning abilities, they are not suitable for adaptive energy management algorithms in dynamic environments.

Self-learning algorithms suitable for IoT-embedded platforms are commonly based on semi-supervised reinforcement learning approaches. One of these approaches is deep reinforcement learning, which combines neural network and reinforcement learning principles. Deep reinforcement learning algorithms can be used to manage energy in battery-less event detection sensors [16], manage resources in hybrid energy wireless networks [17], or jointly optimize data offloading and resource allocation in renewable-energy-aware IoT devices [18]. Although, this approach is self-learning and suitable for multidimensional environments, the algorithm is not fully optimized due to the computational complexity involved in real-time neural network training. Another online and self-learning approach is QL, which is suitable for coordinating the energy consumption in wireless sensor networks [19],

TABLE I
Overview of ML Methods Suitable for Adaptive Energy Harvesting Management in IoT Devices

Methods	Related studies
Neural networks	• Prediction of available energy in energy harvesting nodes [7] • Prediction of energy from multiple energy harvesting sources [8] • Prediction of output power from hybrid energy harvesting sources [9]
Fuzzy logic	• Adaptive sampling algorithm for a continuous source of energy for the sensor node [10] • Optimization of fuzzy rule-based controller prediction [11] • Forecasting of next-day solar energy availability using evolutionary fuzzy rules [12] • Prediction of the amount of battery energy in IoT devices [13] • Optimization of IoT node operation by managing energy consumption [14] • Fuzzy rule-based controllers for embedded sensors [15]
Deep reinforcement learning	• Energy management in battery-less event detection sensors [16] • Resource management in hybrid energy LoRa wireless networks [17] • Optimization of data offloading and resource allocation in renewable-energy-aware IoT devices [18]
Q-learning	• Coordination of energy consumption in a wireless sensor network [19] • Reinforcement learning-based resource allocation for energy harvesting device communications in IoT networks [20] • Management of energy in solar-powered environmental wireless sensor network nodes [21] • Energy and transmission management in network nodes [22]

allocating resources for energy harvesting device communications in IoT networks [20], and managing energy in solar-powered environmental wireless sensor network nodes [21]. This approach is suitable for managing energy in dynamic environments because it applies a self-learning algorithm that is not computationally intensive due to its semi-supervisory nature and a Q-table updated with single values according to the current reward. This can be demonstrated in the energy and transmission management of a node which has been programmed using QL and exhibits appropriate data and energy flushing in queues to achieve better throughput and fewer lost packets [22].

Hybrid ML approaches, such as the combination of fuzzy logic and reinforcement learning techniques, can also be applied. These fuzzy-based reinforcement learning mechanisms first prioritize tasks using fuzzy logic. A reinforcement learning mechanism is then used to solve the problem of tasks with high dimensionality in a dynamic environment [24]. Besides conventional computational models, task schedulers can also be used. This type of approach is able to intelligently schedule application tasks to avoid power failures and maintain forward progress in IoT devices [25].

III. Proposed Model

This section describes the DQL method, its learning policy and implementation for wake-up scheduling in an IoT device. The section includes a reference solution based on a fuzzy logic controller.

A. Double Q-Learning

The DQL method belongs to the reinforcement learning algorithm family. Hasselt [26] proposed this method as a modification to avoid overestimation of the action values produced by the QL algorithm. This approach differs by using two Q-tables instead of one. The Q-tables are denoted Q_A and Q_B and applied to each state/action pair. The DQL finds the action a^*, which is the maximal valued action in the next state s', according to the value function Q_A

$$a^* = \arg\max_a Q_A(s', a). \tag{1}$$

A similar process is applied to b^*, according to Q_B

$$b^* = \arg\max_a Q_B(s', a). \tag{2}$$

Each Q function is updated with a value from the other Q function for the next state. Q_B is used to update Q_A, according to the equation

$$Q_A(s,a) \leftarrow Q_A(s,a) + \alpha\big[R + \gamma Q_B(s', a^*) - Q_A(s,a)\big]. \tag{3}$$

This is performed conversely for Q_B, according to the equation

$$Q_B(s,a) \leftarrow Q_B(s,a) + \alpha\big[R + \gamma Q_A(s', b^*) - Q_B(s,a)\big]. \tag{4}$$

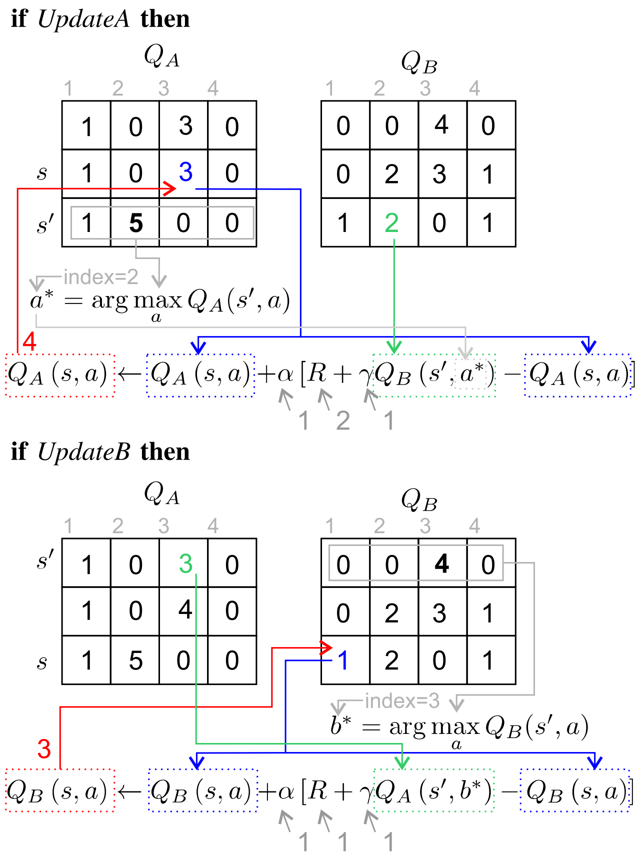
if UpdateA then
Q_A
Q_B
index=2
$a^* = \arg\max_a Q_A(s', a)$
$Q_A(s,a) \leftarrow Q_A(s,a) + \alpha[R + \gamma Q_B(s', a^*) - Q_A(s,a)]$
if UpdateB then
index=3
$b^* = \arg\max_a Q_B(s', a)$
$Q_B(s,a) \leftarrow Q_B(s,a) + \alpha[R + \gamma Q_A(s', b^*) - Q_B(s,a)]$

Fig. 2. DQL algorithm learning process: updating the Q-tables for Q_A and Q_B over two iterations.

The expected value (E) of Q_B for action a^* is mathematically proven to be less than or equal to the maximum value of $Q_A(s', a)$ [26]

$$E\{Q_B(s', a^*)\} \leq \max_a E\{Q_A(s', a)\}. \tag{5}$$

If a large number of iterations are executed, the expected value of $Q_B(s', a^*)$ will be less than the maximum value of $Q_A(s', a)$. It means that $Q_A(s', a)$ is never updated with a maximum value and, thus, never overestimated. This also applies conversely to $Q_B(s', a)$. To select the action for the next run, the appropriate Q-table (Q_A or Q_B) is used.

Fig. 2 shows the learning phase of the DQL algorithm. Generally, DQL algorithms use two estimators instead of one to eliminate any overestimation of rewards [26]. The advantage of two estimators is that the first is used to select an action and the second is used for evaluation. The estimators are switched after each iteration. When the Q-table for Q_A updates, the future reward value is taken from the Q-table for Q_B, and vice versa. Fig. 2 shows an example of two learning phases over two consecutive iterations. The Q-table for Q_A is updated during the first iteration, the Q-table for Q_B during the second.

B. Learning Policy

An integral part of DQL is the learning policy. Generally, the learning policy defines the actions, states, and the reward policy. The DQL controller's aim is to optimize the use of energy. Actions are defined for the next sleep time, i.e., the next period duration. The action set is defined as follows:

$$A = \{720,\ 480,\ 240,\ 120,\ 60,\ 10\}\ (\text{min}). \tag{6}$$

States are defined according to the energy stored in the supercapacitor. The maximum energy which can be stored is calculated from the equation

$$E_{\text{max}} = \frac{1}{2} \cdot C_{\text{store}} V_{\text{max}}^2 \tag{7}$$

where E_{max} is the maximum stored energy in joules, C_{store} is the electrical capacitance in farads, and V_{max} is the maximum supercapacitor voltage generated by the dc/dc converter. For the experiment in the current study, an LTC3109 dc/dc converter supplied electrical output until the supercapacitor voltage dropped below a desired output voltage. The supercapacitor was consequently charged using a minimum of energy, calculated according to the equation

$$E_{\text{min}} = \frac{1}{2} \cdot C_{\text{store}} V_{\text{out}}^2 \tag{8}$$

where E_{min} is the required stored energy in joules in the supercapacitor corresponding to the desired output voltage V_{out}. The state of energy storage (SoES) is computed according to

$$\text{SoES} = \begin{cases} E_{\text{store}} < E_{\text{min}}\text{: } 0 \\ \text{else:} \qquad \frac{E_{\text{store}} - E_{\text{min}}}{E_{\text{max}} - E_{\text{min}}}. \end{cases} \tag{9}$$

SoES is normalized to the interval $<0,\ 1>$ and divided into six states

$$S_i = \left\langle \frac{i-1}{6}, \frac{i}{6} \right\rangle;\ i = 1 \text{ to } 6. \tag{10}$$

If SoES is less than 1/6, then the state is S_1.

The reward policy is based on the current SoES value and the SoES value from the previous cycle; the reward is calculated from the equation

$$R = \text{SoES}_{\text{cycle}} - \text{SoES}_{\text{cycle-1}} \tag{11}$$

where $\text{SoES}_{\text{cycle}}$ is the supercapacitor's range normalized remaining energy, $\text{SoES}_{\text{cycle-1}}$ is the supercapacitor's range-normalized remaining energy from the previous cycle, and R is the reward. The policy is defined according to two conditions: 1) when SoES rises, the controller obtains a positive reward and 2) the action with a longer period increases the probability that the SoES is higher after the performed action.

The relationship between reward and incoming energy is straightforward. If a sudden temperature difference occurs on the TEG, caused by, for example, blowing wind, the reward will also be high. A high reward may cause overestimation, but using DQL instead of QL will decrease the probability of overestimation.

C. DQL Energy Management Algorithm

This section presents a DQL algorithm dedicated to controlling the behavior of an IoT node. The principle behind adapting DQL to this purpose is that the algorithm uses the current and previous step states instead of the current and future states.

Algorithm 1: DQL Algorithm Adapted to the Controller of an IoT Node

```
Initialize $Q(s,a)$, $Q_A(s,a)$ and $Q_B(s,a)$, for each $s \in S$,
$a \in A$
while $true$ do
    Wake up
    Observe $R$, $s'$
    if $UpdateA$ then
        Define $a^* = \arg\max_a Q_A(s',a)$
        $Q_A(s,a) \leftarrow Q_A(s,a)+$
            $+\alpha\big[R + \gamma Q_B(s',a^*) - Q_A(s,a)\big]$
        Set UpdateB
    else if $UpdateB$ then
        Define $b^* = \arg\max_a Q_B(s',a)$
        $Q_B(s,a) \leftarrow Q_B(s,a)+$
            $+\alpha\big[R + \gamma Q_A(s',b^*) - Q_B(s,a)\big]$
        Set UpdateA
    end
    Choose $a$, based on $Q_A$ or $Q_B$, and $s'$
    ($\epsilon$-greedy policy)
    Start action $a$
    $s \leftarrow s'$
    Sleep time according to executed action
end
```

Algorithm 1 defines the DQL process for controlling an IoT node's behavior. After initializing variables and the Q-tables, the IoT node is woken up. The reward R and current state s' are checked. The next step is a learning phase which updates one of the Q-tables (Q_A or Q_B). In the first iteration, the previous state s is unknown, therefore, the learning phase is omitted.

Action a is then selected from the appropriate Q-table (Q_A or Q_B). Finally, the selected action is performed, the current state is stored (as the previous state for the next learning phase), and the IoT node enters sleep mode. After a certain period has elapsed, the IoT node is again woken up, and the algorithm repeats.

D. Fuzzy Logic Controller Reference Solution

This section describes a state-of-the-art energy management solution based on a fuzzy controller, adapted from previously published research articles [10], [11], [14]. The reference solution uses a fuzzy logic controller instead of a DQL controller to schedule the next wake-up time.

The reference fuzzy logic controller has two inputs and a single output. The inputs have been designed to respond to the information available from the IoT node according to the DQL reward policy, where the first input contains the fuzzy sets of $\text{SoES}_{\text{cycle}}$ and the second contains $\text{SoES}_{\text{cycle-1}}$. The output is the next period duration discretized into $\{10,\ 20,\ 30, \ldots,\ 720\}$ min.

The shape of the input and output fuzzy sets depicted in Fig. 3 is triangular, and both inputs contain low, middle, and high sets. The output contains three fuzzy sets, representing slow, middle and fast operation.

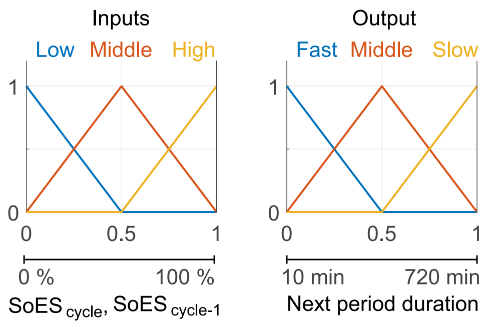

Fig. 3. Shape of the input and output fuzzy sets representing the current and previous SoES, with next period duration output.

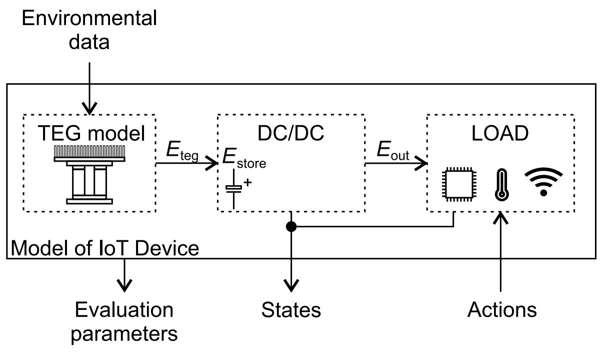

Fig. 4. Hardware model for an IoT device composed of a TEG, dc/dc converter, and load.

TABLE II
RULES FOR THE FUZZY CONTROLLER REFERENCE SOLUTION

SoES$_{\text{cycle}}$	**SoES$_{\text{cycle-1}}$**	**Next period duration**
low		slow
middle	low	fast
middle	middle	middle
middle	high	slow
high		fast

Table II lists the rules for the fuzzy logic controller. The knowledge base contains five rules according to the condition that if the SoES$_{\text{cycle}}$ is X and the SoES$_{\text{cycle-1}}$ is Y, then the next period duration is Z. To achieve the maximum comparable behavior to the proposed DQL approach, these rules follow the same DQL policy. When the current SoES is low, the fuzzy controller selects slow operation, when it is high, it selects fast operation. When the current SoES is in the middle, then the fuzzy logic controller's behavior depends on the previous SoES, where an increase in the SoES resulted in fast operation and a decrease in slow operation. When the SoES is balanced and both input sets are in the middle, then operation is also in the middle.

IV. Experimental Procedure

This section describes the experimental hardware parameters applied in the simulation, the input data, and the performance evaluation parameters for the input data.

A. Experimental Setup and Data

The experiment used a hardware model (Fig. 4) composed of three main modules: 1) a TEG; 2) a dc/dc converter; and 3) a load. The parameters of this device were applied in a simulation for analysis. The TEG hardware is a TEC1-12706 [27] module which generates electrical energy when it is exposed to temperature differences. The study in [28] described the properties of this TEG module through an experimental analysis of its current and voltage characteristics by exposing it to a range of temperature differences. A standalone TEG module is able to produce an open circuit voltage in the range of tens to hundreds of millivolts. This voltage range, however, is not sufficient to directly supply electrical devices, such as microcontrollers (MCUs) or transmission modules.

Dc/dc converters boost voltages. The hardware model's dc/dc module is based on an LTC3109 converter which converts electrical energy from extremely low input voltage sources such as TEGs [29]. The dc/dc converter module in the experiment was a mathematical model designed according to a physical LTC3109 module and respected its basic functionality.

Harvested energy is consumed by the load module. The load module is composed of an MCU, an environmental sensor, nonvolatile memory, and a wireless communications interface. An NXP KL25Z MCU [30] was selected for its low power consumption and wide range of integrated peripherals. The NXP KL25Z also fulfilled the requirements of the architecture and provides a number of low power modes,a feature which allows fine tuning of the energy profile.

Serving as a data collector, a Bosch BME688 4-in-1 [31] environmental sensor, which measures ambient temperature, air humidity and atmospheric pressure, is connected to the MCU via the I^2C bus. The experiment did not impose the necessity of any specific sensor model; the only requirement was an advanced sensor design. This device also integrates a gas sensor which detects volatile organic and sulfur compounds and various other gases.

The model performs two operations: 1) data measurement and 2) data transmission. A memory buffer synchronizes these two operations through a 24CW1280 EEPROM [32], although the FRAM technology was also considered. Currently, no FRAM device is capable of operation at low voltages (e.g., at 1.8 V). A LoRaWAN, currently one of the most popular types of communication tools for IoT devices and offering three communication classes (classes A, B, and C) to cover various use cases, provided communications. The communications link is established with a Semtech SX1261 [33] LoRa transceiver connected via a serial peripheral interface (SPI).

The experimental data contained air temperature and near-surface soil temperature measurements (measured at several depths 0.05, 0.5, 0.1, and 0.2 m) collected over a period of four years (2016–2019). These data were obtained from the Czech Hydrometeorological Institute [34] of the Ministry of the Environment of the Czech Republic and used as input for the experimental model. The data were measured at 10-min intervals at the Churanov Monitoring station, located in the

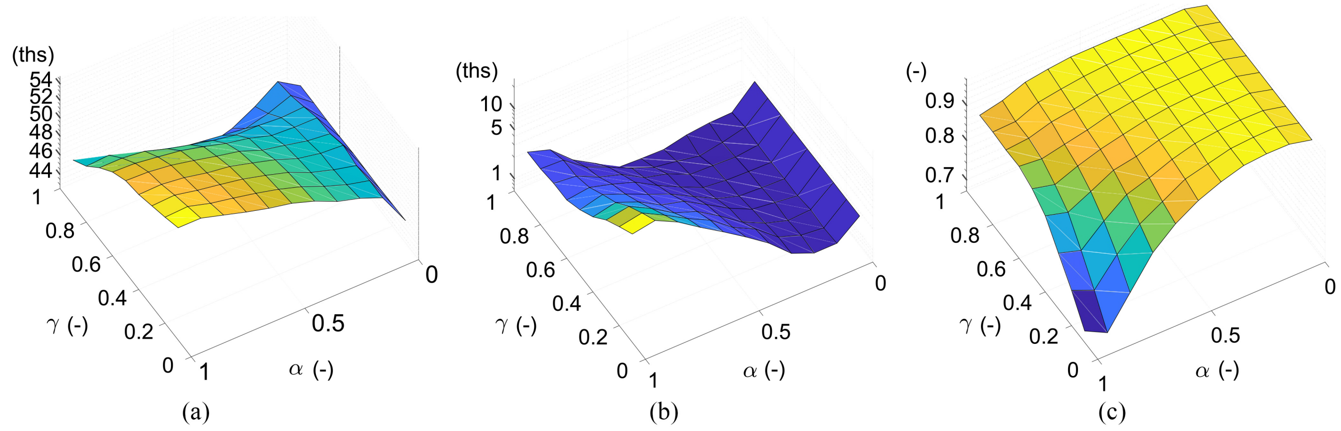

Fig. 5. Results for learning parameters performance by varying the learning parameters (α, γ); (a) represents the number of completed cycles; (b) represents the number of incomplete cycles (failures); and (c) represents the ratio of completed/missed cycles.

Czech Republic at the coordinates 49.0683° latitude, 13.615° longitude, and 1117.8-m elevation.

B. Performance Evaluation

To evaluate the hardware's performance in the simulation, several criteria and performance characteristics were analyzed. Performance was assessed according to successful/completed cycles and unsuccessful/missed cycles. A missed cycle is a period during which transmission is required but the available energy is insufficient. The ratio of both indicators (completed and missed cycles) is calculated according to the equation

$$\text{Ratio} = \frac{\text{Completed}}{\text{Completed} + \text{Missed}} \cdot 100\,\%. \quad (12)$$

The hardware model's energy consumption characteristics, especially unused energy, was evaluated. The quantity of unused energy E_U is the sum of the energy when the supercapacitor is fully charged and the load does not use all of the produced energy. This sum of unused energy is then measured as a ratio to the sum of produced energy.

The average SoES level ($\overline{\text{SoES}}$), average period between two successful transmissions ($\overline{\text{P}}$), percentage of power good pin is active ($\overline{\text{P}_{\text{GOOD}}}$), and average supercapacitor voltage ($\overline{\text{V}_{\text{STORE}}}$) were also monitored to allow an analysis of energy consumption, the average data availability in the cloud and status variables of the IoT node.

V. Results

This section evaluates the performance and variations in the learning parameters over time and discusses the best-performing controller. The learning parameters, represented by α and γ, determine the learning rate and cumulative reward preferences of the DQL algorithm. The ablation study demonstrates the performance of the solution without the ML control algorithm, considering different static settings of the wake-up period. The variations in the learning parameters over time are reflected in the α_R policy, which adjusts the algorithm's learning capability. The time domain analysis provides detailed insights into the behavior within a 200-day interval and the selected types of actions.

A. Learning Parameters Performance

Learning parameters performance was investigated by applying different values (α and γ) in the experimental model. To evaluate learning speed, α was set to a value in the range 0–1, with a step of 0.1. To test sensitivity to the cumulative reward, γ was set in the range 0.1–0.9, also with a step of 0.1. The ϵ-greedy policy was set to 0.98 to help reduce the number of random actions since the experimental model was a control system contained within a measurement device and not having deterministic functionality was undesirable. Each variant of the experiment was repeated 100 times to reduce any stochasticity caused by DQL behavior.

Fig. 5 depicts the results for learning parameters performance. Fig. 5(a) indicates a high number of completed cycles but also a high number of missed cycles. This is, especially, evident at a high α (high learning rate) and low γ (low cumulative reward). Fig. 5(b) indicates that a lower α (slow learning process) produced fewer successful cycles and also far fewer missed cycles. Fig. 5(c) shows the ratio of completed/missed cycles calculated from (12).

Table III lists the ten best-performing controllers according to the ratio of completed/missed cycles defined in Section IV-B. Each candidate differs in its configured learning parameters (α, γ). These candidates were evaluated according to the number and ratio of completed and missed cycles, unused energy (E_U), average SoES, and average period. It is clear that the majority of the best controllers had α parameters distributed in the interval 0.2–0.6, representing slower learning rates. In terms of γ, all the best controllers preferred a cumulative reward in the range 0.7–0.9.

In terms of average SoES, a relationship between the total number of cycles and average SoES is evident. The higher the number of cycles, the lower the average SoES, indicating better energy use. $\overline{\text{V}_{\text{STORE}}}$ parameter reflects the same principle and corresponds with the average SoES. A relationship is also evident between the number of cycles and the average transmission period. A higher number of cycles produces a shorter average transmission period and, thus, higher data availability in the cloud. However, the more aggressive strategy with a maximum number of cycles also produced a higher

TABLE III
CONTROLLER CANDIDATES WITH THE BEST PERFORMANCE BASED ON THE RATIO OF COMPLETED/MISSED CYCLES

*	α (-)	γ (-)	**Compl.** (-)	**Miss.** (-)	**Ratio** (%)	E_U (%)	$\overline{\text{SoES}}$ (%)	$\overline{\text{P}}$ **(min)**	$\overline{\text{P}_{\text{GOOD}}}$ (%)	$\overline{\text{V}_{\text{STORE}}}$ **(V)**
A	0.3	0.8	45,179	682	98.5	27.3	58.1	46.4	79.3	3.596
B	0.4	0.9	42,875	666	98.5	29.6	59.8	48.0	78.9	3.604
C	0.5	0.9	42,640	677	98.4	29.9	59.8	48.8	79.2	3.603
D	0.6	0.9	42,924	700	98.4	29.6	59.0	48.4	79.0	3.558
E	0.2	0.8	45,718	773	98.3	26.7	57.9	45.2	79.6	3.581
F	0.4	0.8	45,472	787	98.3	27.1	57.0	45.6	78.4	3.537
G	0.2	0.7	47,044	829	98.3	25.5	56.1	43.8	78.7	3.546
H	0.3	0.9	43,509	779	98.3	29.0	59.3	46.2	79.6	3.632
I	0.3	0.7	47,243	861	98.2	25.4	55.4	44.6	78.2	3.530
J	0.1	0.4	48,463	889	98.2	24.0	55.5	43.5	77.7	3.485

* – Case, Compl. – Completed cycles (4 years), Miss. – Missed cycles (4 years), $\overline{\text{SoES}}$ – Average SoES, $\overline{\text{P}}$ – Average successful period, $\overline{\text{P}_{\text{GOOD}}}$ – Average power good, $\overline{\text{V}_{\text{STORE}}}$ – Average supercapacitor voltage

TABLE IV
ABLATION STUDY AND REFERENCE SOLUTION BASED ON FUZZY LOGIC COMPARISON

*	**Compl.** (-)	**Miss.** (-)	**Ratio** (%)	E_U (%)	$\overline{\text{SoES}}$ (%)	$\overline{\text{P}}$ **(min)**	$\overline{\text{P}_{\text{GOOD}}}$ (%)	$\overline{\text{V}_{\text{STORE}}}$ **(V)**
10	71,392	138,852	34.0	14.4	15.7	29.4	34.0	1.183
20	51,316	53,804	48.8	33.5	26.1	41.0	40.7	1.701
30	37,336	32,744	53.3	43.3	32.1	56.3	48.6	2.107
40	30,132	22,428	57.3	49.1	36.5	69.8	54.5	2.390
50	25,548	16,500	60.8	53.0	40.0	82.3	58.8	2.595
60	22,352	12,688	63.8	55.8	42.7	94.1	62.4	2.751
120	12,904	4,616	73.7	64.1	53.5	162.9	73.2	3.307
180	8,988	2,692	77.0	67.6	58.0	233.9	76.6	3.518
240	6,892	1,868	78.7	69.5	60.2	305.0	78.5	3.625
360	4,724	1,116	80.9	71.5	62.6	445.0	80.6	3.737
480	3,600	780	82.2	72.5	63.8	584.0	81.6	3.793
600	2,896	608	82.6	73.2	64.6	726.0	82.3	3.829
720	2,432	488	83.3	73.6	65.0	864.5	82.7	3.853
Fuzzy	10,589	793	93.0	65.8	57.5	198.5	77.5	3.522

* – Case, Compl. – Completed cycles (4 years), Miss. – Missed cycles (4 years), $\overline{\text{SoES}}$ – Average SoES, $\overline{\text{P}}$ – Average period, $\overline{\text{P}_{\text{GOOD}}}$ – Average power good, $\overline{\text{V}_{\text{STORE}}}$ – Average supercapacitor voltage

number of missed cycles. $\overline{\text{P}_{\text{GOOD}}}$ indicates the percentage of the time, when IoT node works properly. The best-performing controllers have $\overline{\text{P}_{\text{GOOD}}}$ in range from 77.7% to 79.6%.

B. Ablation Study and Reference Algorithm

The purpose of the ablation study is to compare the DQL approach with control methods that do not rely on ML. The previous study [1] defined several reference controllers which applied static duty cycle periods. In the current study, the authors compared these solutions plus a reference solution based on a fuzzy logic controller to a solution containing a DQL controller. Table IV compares the performance results of reference static controllers and a reference solution based on fuzzy logic. The lowest average SoES and $\overline{\text{V}_{\text{STORE}}}$ corresponded to the expected static controller behavior. Short duty cycle controllers decreased the SoES, resulting in short average periods; the long-duty-cycle controllers did not use incoming energy, resulting in long average periods. The average SoES of the fuzzy reference solution was similar to the DQL controllers, however, the average period was approximately four times higher. In terms of $\overline{\text{P}_{\text{GOOD}}}$, configurations with longer wake-up periods represent higher $\overline{\text{P}_{\text{GOOD}}}$, which leads to more reliable operation. The fuzzy solution achieves $\overline{\text{P}_{\text{GOOD}}}$ of 77.5%, which corresponds to the range of 180 to 240 min in the static configurations.

Fig. 6 graphs the results for the reference solution in relation to the DQL controller's results. The reference solution is marked in blue and indicates static operating periods (720, 480, 240, 120, 60, or 10 min); the fuzzy controller is marked as yellow diamonds; the DQL controller and its dynamic operating periods are marked in brown. The DQL solution is not clearly visible in the upper graph in Fig. 6(a), therefore, Fig. 6(b) provides a scaled detail of this graph and indicates the best DQL cases with the points A–J. For complete cycles and missed cycles, the blue curve splits the graph area into two parts. Controllers (below the blue curve) achieved higher performance than the static controllers. The reference fuzzy controller's results are also below the blue curve and indicate that this method achieved higher performance than the static controllers. The results for the best-case DQLs fall below the blue curve to the right and indicate that the DQL algorithm's performance was greater than both the static controllers and fuzzy controller.

C. Changes in the Learning Policy

In this section, the algorithm's ability to learn in time by reducing the learning factor in each learning cycle (α_R principle) is discussed. This experiment used the controller ($\alpha = 0.3$ and $\gamma = 0.8$) which provided the best performance in the previous experiments. Reduction of the learning rate was based

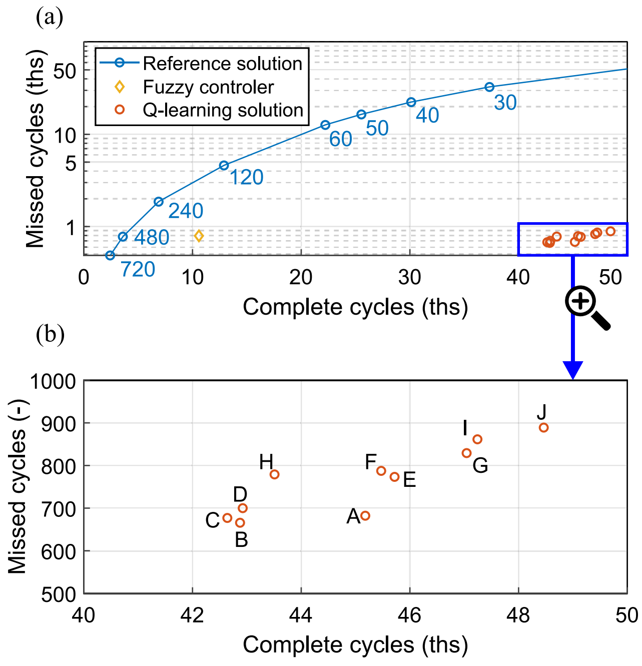

Fig. 6. (a) Performance comparison of the reference solutions [1] and DQL solution, indicating completed and missed cycles. (b) Detail of the upper graph indicating the best DQL controllers.

TABLE V
SEVEN α_R CONFIGURATIONS FOR DETERMINING THE ALGORITHM'S ABILITY TO LEARN IN TIME

Reduction coefficient r	Missed cycles (-)	Ratio (%)
500	1,334	97.14
1,000	1,066	97.65
2,000	1,056	97.89
4,000	880	98.18
8,000	814	98.28
16,000	749	98.38
32,000	811	98.27

on the hypothesis that in a repetitive and conservative environment, preserving obtained knowledge and reducing the ability to learn is advantageous. By contrast, when a controller operates in a dynamic environment, it is better to maintain the learning rate at the initial level.

The α_R principle is defined according to the equation

$$\alpha_r(n) = \alpha_i \cdot \left(\sqrt[r]{G}\right)^n \quad (13)$$

where α_i is the initial value of the learning factor, r is the reduction coefficient, G is the gradient coefficient, and n is the number of simulation steps.

Table V lists the numerical results for the α_R configurations which were applied to determine the algorithm's ability to learn over time. The policy was modified by the reduction coefficient, which reduced the time required for the learning rate. Parameter G was set to 0.5, resulting in a gradual decrease of the learning factor and eventual reduction to half in r cycles. In this experiment, the r coefficient was set to 500, 1000, 2000, 4000, 8000, 16 000, and 32 000. The best controller attained approximately 45 thousand learning cycles over four years of operation; a factor of around 11 300, therefore, indicates that the learning rate was reduced by half in approximately one year.

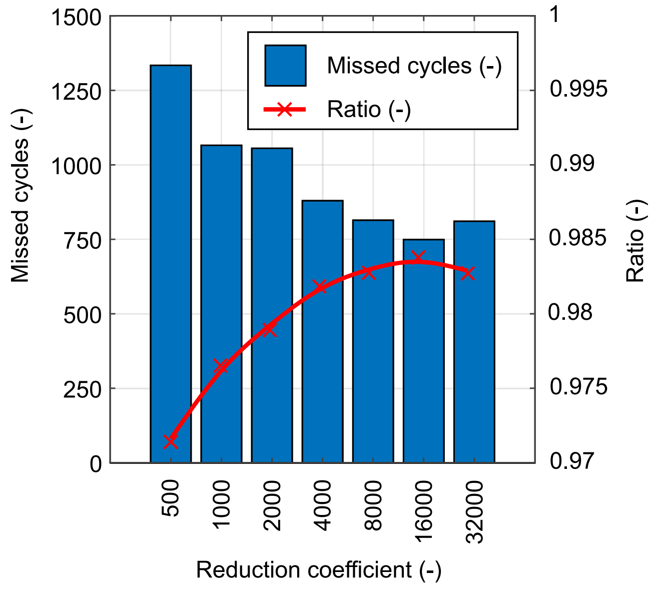

Fig. 7. Seven α_R configurations for determining the algorithm's ability to learn.

TABLE VI
NUMBER OF INDIVIDUAL PERIODS SELECTED DURING CONTROLLER OPERATION

Action type	Number	Percentage
10 min	42,909	91.60 %
60 min	714	1.52 %
120 min	318	0.68 %
240 min	705	1.50 %
480 min	671	1.43 %
720 min	1,529	3.26 %

Fig. 7 shows a graph of the seven α_R configurations for determining the algorithm's ability to learn over time. The results indicate that a high reduction in the learning rate by the reduction coefficient produced poorer performance in terms of missed cycles and the ratio of completed/missed cycles. The best ratios were achieved with low reduction coefficients (8000–32 000). This behavior demonstrates that a quick reduction (500–4000) in the learning rate is not suitable for energy management controllers based on a TEG. It is possible that the controller may benefit from a long-term reduction policy; for example, the best configuration, with 16 000 reduction cycles, required 1.42 years to reduce the learning rate to half. However, it should be noted that the ratio of completed/missed cycles was very close to the best result without a reduction policy, although the result clearly demonstrates that a continual learning rate is a suitable solution.

D. Time Domain Analysis

This section discusses the behavior of the best-performing controller, which used the settings $\alpha = 0.3$ and $\gamma = 0.8$.

Table VI lists the number of individual periods selected during controller operation. The controller operated according to defined output actions in periods of 10, 60, 120, 240, 480, or 720 min. The second column indicates the number of times this period was selected by the controller. The third column specifies the percentage of times this period was selected by

TABLE VII
COMPARISON OF FEATURES IN STATE-OF-THE-ART METHODS AND THE PROPOSED APPROACH

Energy management method	**Model-free design**	**Low computational complexity**	**Dynamic learning**	**On-site update**	**TEG harvesting compatibility**
Neural networks [7], [8], [9]	×	×	×	×	✓
Fuzzy logic [15], [11], [12], [13], [10], [14]	×	✓	×	×	✓
Deep reinforcement learning [16], [17], [18]	×	×	✓	×	×
Q-learning [19], [20], [21], [22]	✓	✓	✓	✓	×
Presented DQL approach	✓	✓	✓	✓	✓

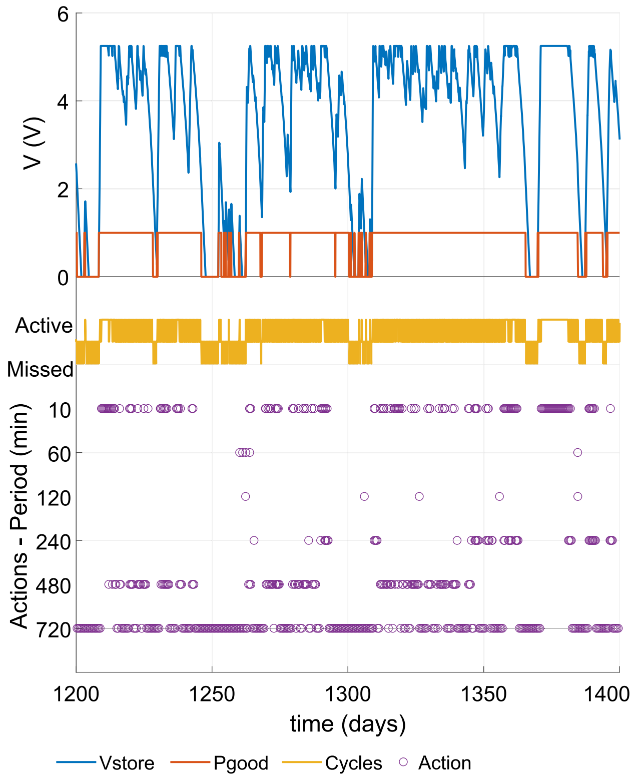

Fig. 8. Selected time window for IoT device operation with the best DQL candidate ($\alpha = 0.3$ and $\gamma = 0.8$). The upper part of the chart shows the time parameters V_{store}, P_{good}, and active and missed cycles. The lower part of the chart indicates the actions performed (period) over 1200–1400 days.

the controller and indicates that the controller selected the fastest action in 91.60% of cases. When energy was unavailable, the controller selected slower actions to obtain a better reward. In 3.26% of cases, the controller slowed down the operating period to 720 min to prevent an outage in the IoT device.

Fig. 8 graphs the results of the simulation for the best DQL candidate over 1200–1400 days. The blue and red curves represent the voltage waveforms of V_{store} and P_{good}, respectively. V_{store} is the supercapacitor voltage which corresponds to the SoES, and P_{good} is the output signal which indicates whether the output voltage is at a sufficient level. The yellow curve indicates whether a cycle was active (completed) or missed. The purple circles indicate executed actions (period).

At a high level of V_{store}, a high density of action was executed every 10 min; at a low level of V_{store}, a high density of action was executed every 720 min. These results correspond with the expected behavior.

VI. Discussion

This section compares the presented DQL approach with state-of-the-art methods and discusses the performance, features, and applications of DQL in TEG-powered IoT nodes.

A. *Comparison With State-of-the-Art Approaches*

The research from related studies and experiments open several discussion points. From a review of the literature on advanced methods, the current study is novel in three aspects. Table VII provides a comparison of the related studies listed in Section II with the proposed DQL approach. The individual ML methods are compared according the design needs of the models (model-free design), computational complexity, the ability to learn continuously (dynamic learning), the ability to learn without cloud assistance (on site updates), and compatibility with TEG-powered systems (TEG harvesting compatibility). Besides QL-based strategies, none of the methods are model-free and, therefore, require models for development. Approaches based on neural networks are characterized by high computational complexity and are, therefore, not suitable for implementation with low cost, low power IoT devices.

In general, approaches based on reinforcement learning are suitable for embedded applications. For IoT devices powered using energy harvesting methods, the embedded energy management algorithm must be able to adapt to the dynamic nature of the environment where the device is located by being able to learn at every step and adapt to the surrounding conditions. Algorithms based on reinforcement learning (deep reinforcement learning, QL, and DQL) satisfy this condition. Another significant parameter in a low-power IoT device is its ability to function with the cloud technology. ML approaches which learn by themselves without the assistance of the cloud technology belong to the reinforcement learning family. Methods based on neural networks and fuzzy logic lack this capability.

Energy harvesting based on the TEG technology is characterized by sudden incoming peaks of energy caused by changes in weather conditions. Energy management strategies must, therefore, be robust and eliminate overestimation

TABLE VIII
SUMMARY OF APPROACHES KEY PARAMETERS

	$\overline{\text{P}}$ (min)	**Ratio** (%)	$\overline{\text{P}_{\text{GOOD}}}$ (%)	**Miss.** (-)	E_{U} (%)
Static 20 min.	41.0	48.8	40.7	53,804	33.5
Static 180 min.	233.9	77.0	76.6	2,692	67.6
Fuzzy	198.5	93.0	77.5	793	65.8
DQL	46.4	98.5	79.3	682	58.1

$\overline{\text{P}}$ – Average period, $\overline{\text{P}_{\text{GOOD}}}$ – Average power good, Miss. – Missed cycles (4 years),

of such events. Neural network and fuzzy logic strategies are developed offline and, therefore, resistant to this type of adaptation in principle. The knowledge base created through reinforcement learning methods may also be compromised by the overestimation of external events. The proposed DQL approach using two Q-tables offers an effective solution to suppress overestimation during sudden changes in incoming energy.

Table VIII presents a summary of the key parameters for static, fuzzy, and DQL approaches. To compare ML approaches, two static controllers (20 and 180 min) are selected based on the comparable average period $\overline{\text{P}}$ parameter. The fuzzy controller has a comparable $\overline{\text{P}}$ with the 180-min static configuration, but the overall reliability, as indicated by the ratio parameter related to missed cycles, is significantly higher. In terms of $\overline{\text{P}_{\text{GOOD}}}$ and E_U, the fuzzy controller and the 180-min static configuration show negligible differences. These facts clearly demonstrate that a dynamic-oriented approach is more suitable for controlling TEG-powered IoT nodes. Similar observations can be made when comparing the DQL approach to the corresponding static 20-min approach. There is a significant difference between the ratio and missed cycles, despite the comparable average period. This finding further confirms that DQL is an appropriate solution for IoT energy management. The comparison of the DQL and fuzzy approaches reveals that DQL outperforms the fuzzy approach in terms of all key parameters.

B. DQL Performance and Features

The study's results demonstrate that the real-time self-learning algorithm designed for IoT devices deployed in environments with variable sources of energy is a suitable solution. This conclusion is based on the study's α_R experiment, which produced superior results without any reduced learning ability in the algorithm. This feature permits application to a wide range of IoT sensors and deployment scenarios. The controller's adaptability is an advantage with IoT sensors where the hardware configuration differs in energy harvester type, capacitor size, hardware age, and other factors as a consequence of DQL principles and a semi-supervised approach driven only by relative state variables from the reward policy.

Self-learning algorithms provide solutions for various ML methods which may require additional data sets (e.g., training data sets for neural networks). The proposed solution uses online self-learning principles and, therefore, performs semi-supervised learning within the deployed device itself. This feature not only eliminates the need for a training data set, it produces different learning results in each IoT device. This approach also eliminates the time-consuming and computationally demanding process of optimizing the design (e.g., fuzzy rule-based controllers) or providing ambient energy predictions (e.g., prediction-based controllers).

In terms of required computational resources, the DQL controller is suitable for IoT devices with hardware limitations. Memory implementation includes two data arrays representing Q-tables with floating point variables. In the each learning step, the Bellman equation updates only one variable selected from the data arrays. Finally, actions are selected by averaging and sorting the data arrays. This simple procedure is more effective than state-of-the-art approaches, such as fuzzy rule-based controllers or neural network evaluation. Overall, DQL provides the means to implement computationally limited, low-cost hardware with low-power specifications.

VII. CONCLUSION

The study presented a reinforcement learning principle designed to optimize energy management in IoT devices and experimentally tested a hardware model for such a device. The model consisted of a TEG energy harvesting subsystem with a dc/dc converter, a load module with an MCU, and a LoRaWAN communications interface. The device followed a reward strategy which compared its current charge status to the charge status in the previous learning step.

The study also presented a DQL-based approach with configurable learning parameters. The results showed that the best-performing DQL controller operated with a 98.5% success rate derived from the ratio of completed/missed operation cycles. The novelty of the solution was discussed in relation to state-of-the-art methods and their properties.

Future work includes two possible directions. In the first, and because the proposed approach demonstrated its ability to adapt to the surrounding environment and specific application, DQL-based methods applied in other domains could be evaluated with simulations that use other hardware or large data sets from a range of deployment locations. The second research opportunity involves long-term deployment and evaluation of an IoT device to study the differences between simulated data and real-life data which contains several observed parameters (e.g., disturbances, malfunctions, temperature changes, etc.)

REFERENCES

[1] M. Prauzek, J. Konecny, and T. Paterova, "Q-learning energy management strategy for TEG-powered environmental monitoring IoT devices: A pilot study," in *Proc. IEEE Symp. Series Comput. Intell.*, 2022, pp. 211–216.

[2] A. Chatterjee et al., "Powering Internet-of-Things from ambient energy: A review," *JPhys Energy*, vol. 5, no. 2, 2023, Art. no. 22001.

[3] F. Tohidi, S. G. Holagh, and A. Chitsaz, "Thermoelectric generators: A comprehensive review of characteristics and applications," *Appl. Thermal Eng.*, vol. 201, Jan. 2022, Art. no. 117793.

[4] P. V. B. C. Silva, C. Taconet, S. Chabridon, D. Conan, E. Cavalcante, and T. Batista, "Energy awareness and energy efficiency in Internet of Things middleware: A systematic literature review," *Ann. Telecommun.*, vol. 78, nos. 1–2, pp. 115–131, 2023.

[5] T. Chen, C. Gao, Z. Wang, H. Ming, M. Song, and X. Yan, "Intelligent energy management of low carbon hybrid energy system with solid oxide fuel cell and accurate battery model," *IET Smart Grid*, vol. 6, no. 1, pp. 28–37, 2023.

[6] A. Sabovic, M. Aernouts, D. Subotic, J. Fontaine, E. De Poorter, and J. Famaey, "Towards energy-aware TinyML on battery-less IoT devices," *Internet Things*, vol. 22, Jul. 2023, Art. no. 100736.

[7] W. Xiao, F. Liu, and J. Zhang, "Adaptive dynamic programming for multi-point scheduling in energy harvesting wireless sensor networks," in *Proc. IEEE 12th Int. Conf Ubiquitous Intell. Comput. IEEE 12th Int. Conf Auton. Trusted Comput. IEEE 15th Int. Conf Scalable Comput. Commun. Assoc. Workshops (UIC-ATC-ScalCom)*, 2015, pp. 1498–1502.

[8] Z. Qadir, E. Ever, and C. Batunlu, "Use of neural network based prediction algorithms for powering up smart portable accessories," *Neural Process. Lett.*, vol. 53, no. 1, pp. 721–756, 2021.

[9] Y. Park, K. Cho, and S. Kim, "Performance prediction of hybrid energy harvesting devices using machine learning," *ACS Appl. Mater. Interfaces*, vol. 14, no. 9, pp. 11248–11254, 2022.

[10] M. A. M. Yazid, A. Jazlan, M. Z. M. Rodzi, M. A. Husman, and A. R. Afif, "A method for preserving battery life in wireless sensor nodes for LoRa based IOT flood monitoring," *J. Commun.*, vol. 17, no. 4, pp. 230–238, 2022.

[11] M. Prauzek, P. Krömer, J. Rodway, and P. Musilek, "Differential evolution of fuzzy controller for environmentally-powered wireless sensors," *Appl. Soft Comput.*, vol. 48, pp. 193–206, Nov. 2016.

[12] P. Musilek, P. Krömer, J. Rodway, and M. Prauzek, "Pressure-based forecasting of next-day solar energy availability using evolutionary fuzzy rules," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, 2015, pp. 1–8.

[13] B. Al Kindhi and I. S. Pratama, "Fuzzy logic and IoT for smart city lighting maintenance management," in *Proc. 3rd East Indonesia Conf. Comput. Inf. Technol. (EIConCIT)*, 2021, pp. 369–373.

[14] V. Melo, G. Funchal, J. Queiroz, and P. Leitão, "A fuzzy logic approach for self-managing energy efficiency in IoT nodes," in *Proc. Internet Things. Technol. Appl. 4th IFIP Int. Cross Domain Conf.*, 2022, pp. 237–251.

[15] J. Rodway and P. Musilek, "Harvesting-aware energy management for environmental monitoring WSN," *Energies*, vol. 10, no. 5, p. 607, 2017.

[16] F. Fraternali, B. Balaji, D. Sengupta, D. Hong, and R. K. Gupta, "Ember: Energy management of batteryless event detection sensors with deep reinforcement learning," in *Proc. 18th Conf. Embedded Netw. Sens. Syst.*, 2020, pp. 503–516.

[17] R. Hamdi, E. Baccour, A. Erbad, M. Qaraqe, and M. Hamdi, "LoRa-RL: Deep reinforcement learning for resource management in hybrid energy LoRa wireless networks," *IEEE Internet Things J.*, vol. 9, no. 9, pp. 6458–6476, May 2022.

[18] H. Ke, J. Wang, H. Wang, and Y. Ge, "Joint optimization of data offloading and resource allocation with renewable energy aware for IoT devices: A deep reinforcement learning approach," *IEEE Access*, vol. 7, pp. 179349–179363, 2019.

[19] Y. Ge and Y. Nan, "Adaptive energy management by reinforcement learning in cluster-based solar powered WSNs," in *Proc. 7th Int. Conf. Inf. Sci. Control Eng. (ICISCE)*, 2020, pp. 2303–2307.

[20] A. Omidkar, A. Khalili, H. H. Nguyen, and H. Shafiei, "Reinforcement learning based resource allocation for energy-harvesting-aided D2D communications in IoT networks," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 16521–16531, Sep. 2022.

[21] M. Prauzek, J. Konecny, J. Hlavica, and P. Musilek, "Self-learning for day-night mode energy strategy for solar powered environmental WSN nodes," in *Proc. IEEE Can. Conf. Elect. Comput. Eng. (CCECE)*, 2020, pp. 1–5.

[22] Y. Kim and T.-J. Lee, "Learning nodes: Machine learning-based energy and data management strategy," *EURASIP J. Wireless Commun. Netw.*, vol. 2021, no. 1, pp. 1–16, 2021.

[23] M. Zawish, N. Ashraf, R. I. Ansari, and S. Davy, "Energy-aware AI-driven framework for edge computing-based IoT applications," *IEEE Internet Things J.*, vol. 10, no. 6, pp. 5013–5023, Mar. 2023.

[24] M. R. Raju and S. K. Mothku, "Delay and energy aware task scheduling mechanism for fog-enabled IoT applications: A reinforcement learning approach," *Comput. Netw.*, vol. 224, Apr. 2023, Art. no. 109603.

[25] A. Sabovic, A. K. Sultania, C. Delgado, L. De Roeck, and J. Famaey, "An energy-aware task scheduler for energy-harvesting batteryless IoT devices," *IEEE Internet Things J.*, vol. 9, no. 22, pp. 23097–23114, Nov. 2022.

[26] H. Hasselt, "Double *Q*-learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 23, 2010, pp. 2613–2621.

[27] "TEC1-12706—Thermoelectric cooler." 2021. [Online]. Available: https://peltiermodules.com/peltier.datasheet/TEC1-12706.pdf

[28] T. Paterova et al., "Environment-monitoring IoT devices powered by a TEG which converts thermal flux between air and near-surface soil into electrical energy," *Sensors*, vol. 21, no. 23, p. 8098, 2021.

[29] "LTC3109—Auto-polarity, ultralow voltage step-up converterand power manager." 2021. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/3109fb.pdf

[30] "NXP KL25Z MCU." 2014. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/KL25P80M48SF0.pdf

[31] "Bosch BME688 sensor." 2021. [Online]. Available: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme688-ds000.pdf

[32] "24CW1280 EEPROM." 2018. [Online]. Available: https://www.microchip.com/en-us/product/24CW1280

[33] "Semtech SX1261 LoRa transceiver." 2019. [Online]. Available: https://www.semtech.com/products/wireless-rf/lora-core/sx1261

[34] "Czech hydrometeorological institute." 2021. [Online]. Available: https://www.chmi.cz/

Michal Prauzek (Senior Member, IEEE) was born in Ostrava, Czech Republic, in 1983. He received the bachelor's degree in control and information systems, the master's degree in measurement and control systems, and the Ph.D. degree in technical cybernetics from VSB—Technical University of Ostrava, Ostrava, in 2006, 2008, and 2011, respectively.

He has been working with the Department of Cybernetics and Biomedical Engineering, VSB—Technical University of Ostrava since 2010, and he is currently an Associate Professor. He also worked as a Research Postdoctoral Fellow with the University of Alberta, Edmonton, AB, Canada, from 2013 to 2014. He has authored more than 90 articles and conference papers and has eight registered inventions. His research areas include embedded systems, data and signal analysis, control design, and machine learning.

Dr. Prauzek is an IEEE Senior Member active in the Systems, Man, and Cybernetics Society and the Engineering in Medicine and Biology Society.

Jaromir Konecny (Member, IEEE) was born in Frydek-Mistek, Czech Republic, in 1986. He received the bachelor's degree in control and information systems, the master's degree in measurement and control engineering, and the Ph.D. degree in technical cybernetics from the VSB—Technical University of Ostrava, Ostrava, Czech Republic, in 2008, 2010, and 2014, repectively.

He has been working with the Department of Cybernetics and Biomedical Engineering, VSB—Technical University of Ostrava since 2012 and is currently an Associate Professor. He has authored more than 50 articles and conference papers and has four registered inventions. His research areas include embedded systems, electronics, environmental monitoring systems, and localization systems in robotics.

Dr. Konecny is an IEEE Member active in the Systems, Man and Cybernetics Society, the Computational Intelligence Society and Internet of Things Community.

Tereza Paterova received the bachelor's degree in biomedical technology and the master's degree in biomedical engineering from the VSB—Technical University of Ostrava, Ostrava, Czech Republic, in 2017 and 2020, respectively, where she is currently pursuing the Ph.D. degree in cybernetics and research with the Department of Cybernetics and Biomedical Engineering.

She is also a part of the research team for the university's Czech and international projects. She has coauthored several articles published in journals and conferences. Her research areas are embedded systems, machine learning approaches, and environmental monitoring systems.