

# Enabling Deep Reinforcement Learning on Energy Constrained Devices at the Edge of the Network

Jernej Hribar and Ivana Dusparic  
CONNECT-Trinity College Dublin, Ireland  
Emails: {jhribar, duspari}@tcd.ie

**Abstract**—Deep Reinforcement Learning (DRL) solutions are becoming pervasive at the edge of the network as they enable autonomous decision-making in a dynamic environment. However, to be able to adapt to the ever-changing environment, the DRL solution implemented on an embedded device has to continue to occasionally take exploratory actions even after initial convergence. In other words, the device has to occasionally take random actions and update the value function, i.e., re-train the Artificial Neural Network (ANN), to ensure its performance remains optimal. Unfortunately, embedded devices often lack processing power and energy required to train the ANN. The energy aspect is particularly challenging when the edge device is powered only by a means of Energy Harvesting (EH). To overcome this problem, we propose a two-part algorithm in which the DRL process is trained at the sink. Then the weights of the fully trained underlying ANN are periodically transferred to the EH-powered embedded device taking actions. Using an EH-powered sensor, real-world measurements dataset, and optimizing for Age of Information (AoI) metric, we demonstrate that such a DRL solution can operate without any degradation in the performance, with only a few ANN updates per day.

**Index Terms**—Age of Information, Edge computing, Energy Harvesting, Deep Reinforcement Learning

## I. INTRODUCTION

Recently, we have witnessed a rise in Artificial Neural Network (ANN) applications at the edge of the network. In most cases, the ANN is trained using a large quantity of data on a high-performance computing system, and only a fully trained ANN is executed on the resource-constrained embedded device at the edge. This approach is highly effective in a supervised learning setting as the device performs only an inference. However, for a Deep Reinforcement Learning (DRL)-based solution, which combines the Reinforcement Learning (RL) with ANN, the ANN has to be regularly updated due to continuous exploring in interaction with the environment, to avoid performance degradation.

Training an ANN on an embedded device is often challenging due to a lack of memory, processing power, and energy. The latter is especially problematic for embedded edge devices, as the energy used for training the ANN is enormous for such devices [1], e.g., back-propagation for the entire batch of experiences in a Deep Q-Network (DQN) algorithm [2]. Consequently, training on a constrained embedded device is practically impossible. An alternative solution is to perform training on another more resource-rich device, e.g., cloudlet server [3], and periodically transmit the ANN to the embedded device. In other words, the embedded device will make decisions and collect information, but the processing-intensive task

of training the ANN will be carried out by the unconstrained device. However, transmitting updated ANN weights to the embedded device also requires substantial energy. In this paper, we explore if an embedded device can support a DRL-based solution without performing training and rely only on a sporadic update of ANN weight values.

To test the feasibility of our approach, we focus on a system with a single sensor transmitting measurements to the sink, and measure the performance of the system using the Age of Information (AoI) metric [4]. The AoI measures the timeliness of collected information and is becoming ever more important in the Internet of Things (IoT) network [4]–[7]. By measuring the time passed since a source of information, e.g., a sensor, has generated new information in the form of a status update, e.g., a measurement, it is possible to assess the value of the collected information at the sink, e.g., monitor. The underlying assumption is that the more recent the received information is, the more relevant it is for the system. However, when the embedded device has limited energy available, e.g., a non-rechargeable battery or a reliance on Energy Harvesting (EH), there is a fundamental trade-off between the timeliness of collected information and the energy consumption [5]. In such a system, the device has to balance the frequency with which it transmits status updates and the available energy. In our work, we focus on a system in which a source of information can replenish its energy through EH; therefore finding the optimal update policy that minimises the average AoI is not trivial. Often, the optimal policy assumes that the amount of energy the EH-powered source will collect [6], [7] is known to the system. However, typically, such information is not available as the collected energy varies over time. Therefore, solutions capable of adapting to the dynamic environment are more suitable, as considered in our work.

In this paper, we are the first to propose a method that enables the adoption of DRL on energy-constrained devices. Our solution is a two-part algorithm. The first part is deployed on an energy-constrained device, making decisions and collecting information. The second part of the algorithm is on the resource-unconstrained device performing energy-intensive training. The weights of the fully trained ANN, crucial to the correct performance of DRL, are periodically transmitted to the device. However, each transfer of ANN weights consumes a high amount of the embedded device's energy. Therefore, we also balance the frequency of transmitting ANN weights on top of finding the optimal policy for transmitting status updates.

We show that our approach enables the constrained device to learn an updating policy that minimises the average AoI in the system of the EH-powered device. To provide realism, we use real illuminance measurements to model the time-varying energy arrival over ten days for a device relying on a solar panel to harvest energy. Additionally, we demonstrate that the proposed approach performance is comparable to an unconstrained DRL solution, i.e., an approach that ignores all device constraints and is able to perform all actions, including the training process, on the device itself. Finally, we demonstrate that our solution can outperform heuristic updating policies under three different energy profiles.

## II. RELATED WORK

In the literature, there are three prevailing methods of employing RL algorithms on energy-constrained devices. The first method is executing a computationally light RL algorithm directly on constrained devices [8], [9]. For example, the authors in [8] employed simple State-action-reward-state-action (SARSA) and  $\epsilon$ -greedy algorithms to maximise the sensor's throughput by controlling its transmission power and considering available energy. While the authors in [9] used an actor-critic method to manage power on a EH-powered sensor to maximise the quality of service. Unfortunately, the limitations of using such RL algorithms are their inability to perform well in a system with a large state space, as is the case in many real-world scenarios. Usually, an ANN with multiple hidden layers is used as a function approximation for the value function. Consequently, the second prevailing method is implementing a DRL algorithm on another resource unconstrained device that manages the energy-constrained device with control messages [10]–[12]. For example, DRL-based energy-aware scheduling schemes, implemented on a gateway, for battery-powered or EH-powered devices were proposed in [10]–[12]. At the same time, the third method was developed in which the ANN is trained in an offline manner and then deployed on the constrained device [13], [14]. In such a case, the constrained device exploits the pre-trained ANN to minimise training carried out on the constrained device. For example, such a scheme was proposed in [14] to adjust the low-power sensors' transmission power dynamically. The work in [13] proposed a design of a distributed DRL solution for controlling the device's power. In the latter, authors employed offline learning to mitigate the energy consumption of embedded devices. In contrast to these three methods, we enable the full *deep* RL, i.e., RL with value function approximated by the ANN, on energy-constrained embedded devices. Our solution relies on continuous updating rather than once-off transfer, with heavy training at each step offloaded to a more resource-rich device.

## III. SYSTEM MODEL

We consider a sensor network consisting of one wireless sensor node, i.e., information source, and one sink, e.g., monitor, as shown in Fig. 1. We assume that the source operates using sleep and wake-up cycles, a commonly employed practice to preserve low-power sensors' energy. For simplicity,

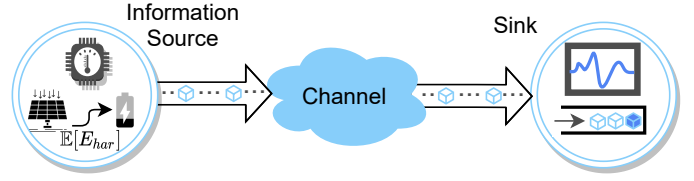


Fig. 1: System model illustration.

we consider a slotted time  $t \in \mathcal{T} = \{1, \dots, T\}$ . In each time step, the source consumes  $E_M$  amount of energy to support the essential operations of the source and to take the observation  $O_t$ , e.g., temperature measurement.

The source node harvests the energy from, e.g., light, vibrations, or wind. The harvested energy is proportional to the current  $I_{EH}(t)$  and voltage  $U_{EH}(t)$  originating from the EH unit. The source stores the collected energy in a capacitor of finite-size  $B$ . Consequently, the source's energy  $E(t)$  is limited to an interval;  $E(t) \in [0, B]$ . The  $E(t)$  varies over time due to the variable  $I_{EH}(t)$  and due to the energy cost  $E_{TR}(t)$  of transmitting a status update.

### A. Timeliness of Collected Information

The source generates a status update  $U_t$  at time step  $t$  if the source decides to transmit new information to the sink for collection. Hence, we can define the process  $\Delta(t)$ , representing the timeliness of collected information, i.e., status updates, as follows:

$$\Delta(t) := t - G(t). \quad (1)$$

where  $G(t)$  represents a time instance in which a status update is generated. We assume that the status update  $U_t$  also contains observations the source has obtained in the previous time steps, as is often the case in real deployments. The source employs buffer  $\mathcal{B}$  with length  $M$  to save past observations. Meaning, that the newly generated status update contains a set of multiple observations, i.e.,  $U_t = \{O_t, O_{t-1}, \dots, O_{t-M+1}\}$ .

The first metric we employ to measure the timeliness of information in the system is average AoI. In discrete systems it is defined as follows:

$$\bar{\Delta}(t) = \frac{1}{T} \lim_{T \rightarrow \infty} \sum_{t=1}^T \Delta(t). \quad (2)$$

The average value enables us to assess how the system performs as a whole in terms of AoI. However, the main limitation of the average AoI metric is that it is not possible to determine how are the AoI values of status updates spread around the average. To that end, the *peak* AoI was proposed by Costa et. al. [15] to provide a better characterization of a process  $\Delta(t)$ . First, we define  $P_i$  as a peak, i.e., the highest value of the process  $\Delta(t)$  before a new status updated is received at the sink. Then, the peak AoI is determined as:

$$\Delta_{peak}(t) = \frac{1}{K} \sum_{i=1}^K P_i, \quad (3)$$

where  $K$  represents the number of received status updates. In EH-powered sources, the peak AoI is connected with down-time, i.e.,  $T_{down}$ . We define  $T_{down}$  as a time during which the source does not have enough energy to transmit a new status update, i.e.,  $E(t) < E_{TR}(t)$ . In general, if a source experiences high  $T_{down}$ , the peak AoI is higher, a behaviour we analyse in more depth in the validation Section V.

### B. Minimising AoI in an Energy-constrained System

The system's main goal is to find a status updating policy  $\pi$  that will ensure that the EH-powered source will minimise the average and peak AoI. Therefore, we formulate the problem the system is facing as:

$$\begin{aligned} \min_{\pi} \quad & \bar{\Delta}(t), \Delta_{peak}(t) \\ \text{s.t.} \quad & E(t) \geq 0, \forall t = 1, \dots, T, \\ & T_{down} = 0, \forall t = 1, \dots, T. \end{aligned} \quad (4)$$

Due to the limited and randomly time-varying nature of  $E(t)$ , deriving an updating policy that will minimise the average peak AoI in the EH-powered system is a non-trivial task. To that end, we employ DRL and modify it to be able to overcome hardware constraints of a low-cost EH-powered source, as we describe in the next section.

## IV. ENABLING DRL ON ENERGY CONSTRAINED DEVICES

In each time step, the source has to decide whether to transmit a status update or wait. Each option has a different energy cost. If the source decides to transmit, it uses the energy required to transmit a status update; if it decides to wait, the value of the AoI will increase. Therefore, it is paramount that the source finds a policy capable of preserving the energy while also ensuring that the resulting average AoI will be minimal. To that end, we employ a DQN algorithm [2] to enable the source to decide dynamically. Alternatively, any other DRL algorithm, e.g., Deep Deterministic Policy Gradient (DDPG) [16], Soft Actor-Critic (SAC) [17], etc., could be applied with a minimal change to our design. Utilizing the ANN as a function approximation enables us to use real positive values, as obtained by the source, as an input to the decision-making process.

### A. States, Actions, and Reward Function

We define states, actions, and rewards for our proposed DRL-based approach as follows:

**State:** The state  $s_t$  consist of the three most critical parameters for the source's decision: the amount of energy currently available on the source, AoI value, and the harvesting current, i.e.,  $s_t = (E(t), \Delta(t), I_{EH}(t))$ . Due to the high variability in the values of the parameters in the state vector, the number of possible states results in millions, motivating the use of DRL.

**Action:** The source only has two actions available: transmit or do not transmit. Therefore, the action value ( $a_t \in \mathcal{A}$ ) can only take two different values, i.e.,  $\mathcal{A} = \{0, 1\}$ .

**Reward:** The reward  $r_t$  consist of two parts, i.e.,  $r_t(t) = r_a(t) + r_e(t)$ . The first part of the reward depends on the

action and average AoI achieved in the past day. When the source decides to not transmit a new status update the reward is proportional to one tenth of the average AoI, i.e.,  $r_a(t) = 1 - \frac{1}{10}\bar{\Delta}(t)$ . When the source transmits a new status update, the reward is much higher. However, to ensure stability we had to limit the reward as follows:

$$r_a(t) = \begin{cases} 2.5(40 - \bar{\Delta}(t)) + 2 - \frac{1}{10}\bar{\Delta}(t), & \text{if } \bar{\Delta}(t) \leq 40 \\ 2 - \frac{1}{10}\bar{\Delta}(t), & \text{otherwise} \end{cases} \quad (5)$$

The energy part of the reward function activates once the available energy in the source is below the threshold, i.e.,  $r_e(t) = -1000$ ; if  $E(t) \leq 0.15B$ .

### B. Two-part DQN for Energy Constrained Devices

We describe the proposed two-part algorithm in Algorithms 1 and 2. Algorithm 1 is executed on the EH-powered source, while Algorithm 2 runs on the sink. At the start, the ANNs, representing the Q-value function, on both the source and the sink are initialised with identical random weights ( $\theta_c, \theta_p$ , and  $\theta_i$ ). In practice, this can be easily achieved using the same random seed number.

**Algorithm 1** The first part of the proposed algorithm running on the energy-constrained device (source).

---

```

1: Init action-value function  $Q_c$  with weights  $\theta_c = \theta_r$ 
2: for  $t = 1, T$  do
3:   if  $E(t) > E_M$  then
4:     Take observation  $O_t$  and store it to buffer  $\mathcal{B}$ 
5:     With probability  $\epsilon$  select a random action  $a_t$ 
6:     Otherwise select  $a_t$  as  $\arg\max_a Q_c(s_t, a|\theta_c)$ 
7:     if  $a_t == \text{True}$  and  $E(t) \geq E_{TR}(t)$  then
8:       Transmit  $U_t$  to the sink for collection
9:     end if
10:    if  $T_u \geq T_{ANN}$  and  $E(t) \geq E_{ANN}(t)$  then
11:      Update  $Q_c$  using  $Q_p$  weights, i.e.,  $\theta_c = \theta_t$ 
12:       $T_u \leftarrow 0$   $\triangleright$  Set ANN update timer to zero.
13:    else
14:       $T_u \leftarrow T_u + 1$ 
15:    end if
16:  end if
17: end for

```

---

**Algorithm 1:** At the start of each time step, the EH-powered source wakes up, takes the new measurement  $O_t$  and adds it to buffer  $\mathcal{L}$ . In the next step, the source will, with probability  $\epsilon$ , select the action randomly or use cached ANN to decide on the action, i.e.,  $\arg\max_a Q_c(s_t, a|\theta_c)$ . The source will send a new status update, containing up to  $M$  observations, to the sink, provided that the source has enough energy to make the transmission. Before entering sleep mode again, the source checks if the weights  $Q_c$  should be updated. The  $T_{ANN}$  represents the minimum number of time steps that need to pass before the ANN is updated. While  $T_u$  counts time passed since the last update.

**Algorithm 2:** On the other hand, the sink will parse the received status update  $U_t$  and determine the reward for every

**Algorithm 2** The second part of the proposed algorithm running on the resource unconstrained device (sink).

```

1: Init policy action-value function  $Q_p$  with weights  $\theta_p = \theta_r$ 
2: Init target action-value function  $Q_t$  with weights  $\theta_t = \theta_p$ 
3: Init replay memory  $\mathcal{D}$  to capacity  $D$ 
4: for  $t = 1, T$  do
5:   if sink receives  $U_t$  then
6:     for  $O_i$  in  $U_t$  do
7:       Parse  $O_i$  to get  $s_t, s_{t+1}, r_t, a_t$  and store in  $\mathcal{D}$ 
8:     end for
9:   end if
10:  Sample random batch of  $N$  experiences from  $\mathcal{D}$ 
11:  for every experience  $\{s_j, s_{j+1}, r_j, a_j\}$  in batch do
12:    Set  $y_j = r_j + \gamma \max_{a_{j+1}} Q_t(s_{j+1}, a_{j+1})$ 
13:  end for
14:  Calculate the loss  $\mathcal{L} = \frac{1}{N} \sum_{j=0}^{N-1} (Q(s_j, a_j) - y_j)^2$ 
15:  Update  $Q_p$  by minimising the loss  $\mathcal{L}$ 
16:  Every  $C$  steps, update  $Q_t$  using  $Q_p$ , i.e.  $\theta_t = \theta_p$ 
17: end for

```

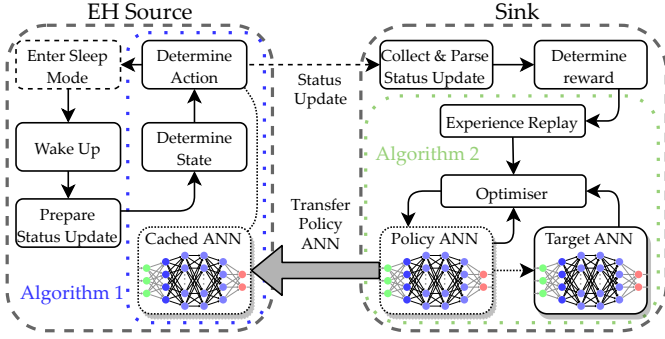


Fig. 2: The block diagram of the proposed solution.

received observation. Thus, the sink can reconstruct up to  $M$  past episodes necessary to obtain the *experience* (i.e., a tuple consisting of a state, action, reward, and future state,  $\{s_t, s_{t+1}, r_t, a_t\}$ ), essential for the training of any DRL algorithm, where  $M$  denotes the number overall measurements the device transmits. These experiences are then used to train the policy ANN using batch learning. The learning process, i.e., calculating the loss  $\mathcal{L}$  and updating policy ANN  $Q_p$  by minimising the loss  $\mathcal{L}$ , follows the standard DQN algorithm approach as described in [2].

Fig. 2 depicts the block diagram overview of the proposed solution and illustrates the interactions between the energy-constrained device and the sink (resource unconstrained device).

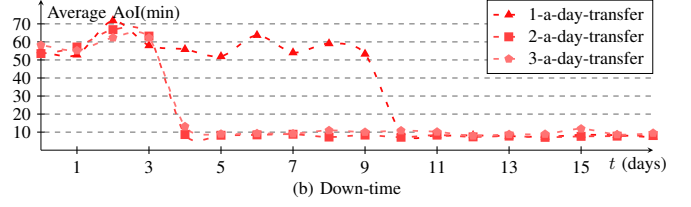
### C. Implementation

We employ DQN structure with four hidden layers. The first and the last hidden layer consist of four neurons, while the inner two hidden layers are implemented with eight neurons each. All layers use the ReLU activation function. To prevent over-fitting, we have added a ten per cent dropout layer between the hidden layers, and we also applied batch normalisation before the first hidden layer. We list the rest of the DQN algorithm hyperparameters in Table I.

TABLE I: DQN Hyperparameters

Hyperparameter	Value	Hyperparameter	Value
Batch size $N$	64	Memory size $D$	$10^5$
Optimizer	Adam	Loss Function	MSE

(a) Average AoI



(b) Down-time

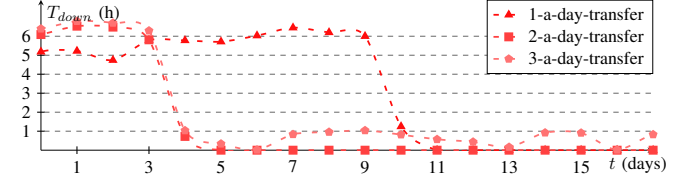


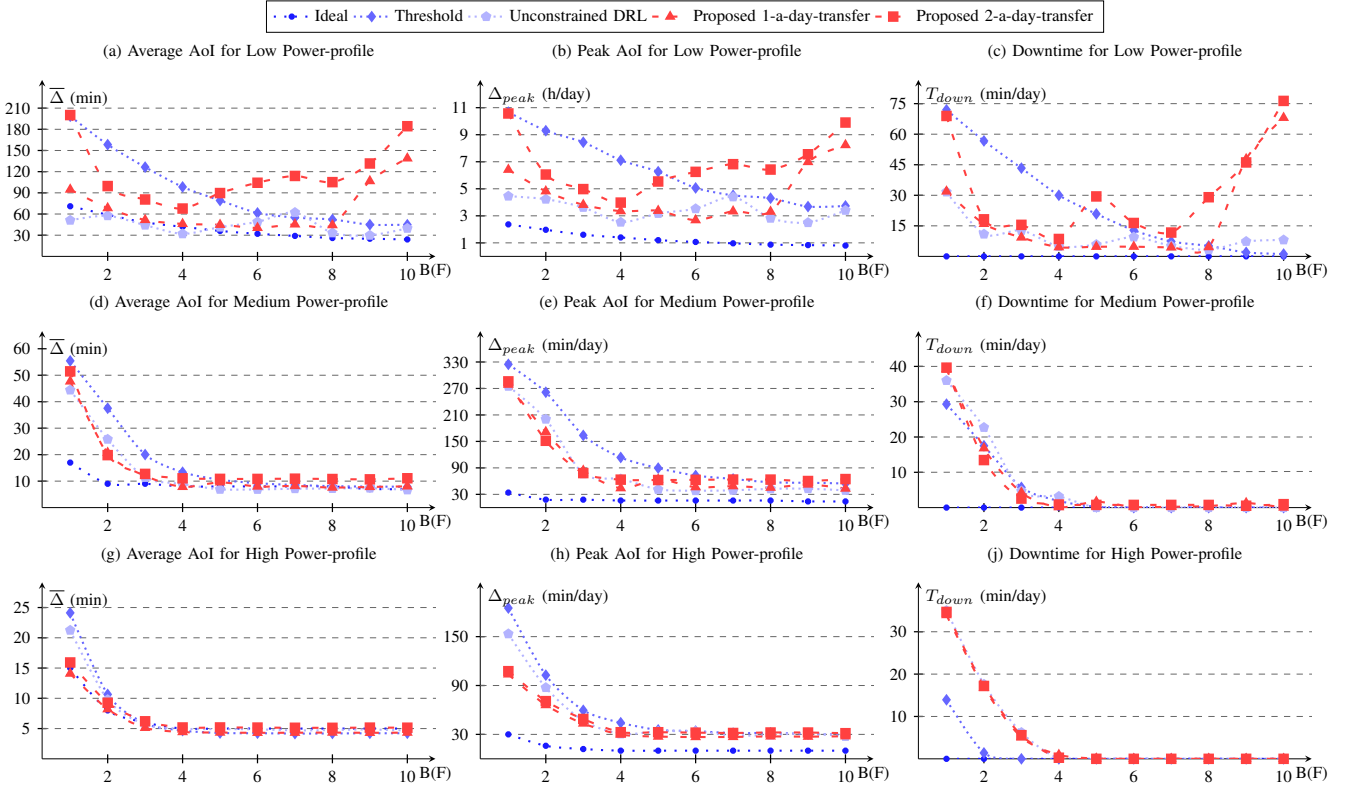
Fig. 3: Daily average AoI and down-time.

## V. VALIDATION

In this section, we first show the impact of the frequency of ANN weight updates on the performance. We continue by validating the feasibility of the proposed mechanism by comparing its performance, in terms of achieved average AoI, peak AoI, and downtime, to results obtained by three baseline updating policies. To add realism to our simulations, we model the  $I_{EH}(t)$  using illuminance measurements obtained in a real sensor deployment at Intel Berkeley Research laboratory [18]. By combining the illuminance measurements with the measurements performed by Yue et al. [19] we can determine the energy an EH-powered source collects in a real-world setting. Additionally, we divide the sources based on the amount of energy a source gathers per day into three power profiles: low, medium, and high. Testing the performance under different energy scenarios enables us to demonstrate the proposed solution generalisability. For illustration, a source with a low power profile collects on average  $6J$ , a medium one  $14J$ , and a high  $20J$  of energy per day.

We model the consumption of the source based on the LoRaWAN class B with spread factor  $8^1$ . The energy required to transmit the status updates varies depending on the number of measurements the source sends. Therefore, the energy for transmitting the status update  $E_{TR}$  varies between  $35mJ$  to  $100mJ$ . In our simulation, the source reports humidity and temperature readings as measured by a sensor in the Intel laboratory deployment. Furthermore, we assume that the source can successfully transmit a new status update with probability  $\eta = 0.9$ . In our implementation, the ANN's weights file size is around  $10kB$ . In our work, we assume that the sink transmits the file without any compression. Transferring a large amount of data to an embedded device is not unusual as devices are often updated, e.g., updating firmware on embedded device [20]. According to our estimates, the source would have to listen for around  $30s$  to receive the file. Consequently, the energy

<sup>1</sup>We rely on the following calculator to determine the energy consumption: [www.lorawebtools.nl](http://www.lorawebtools.nl).

Fig. 4: Average AoI, peak AoI, and  $T_{down}$  for three different power profiles.

cost of updating ANN  $E_{ANN}$  is in the range of  $0.9J$ . We list the remaining energy parameters in Table II.

TABLE II: Simulation Parameters

Parameter	Value	Parameter	Value	Parameter	Value
time-step size	120s	$E_{ANN}$	900mJ	$E_M$	1.5mJ
$U_{EH}(t)$	3V	$\eta$	0.9	$M$	4

#### A. The Impact of Frequency of ANN Weights Updates

In Fig. 3, we show how fast, depending on the number of ANN weight updates a system can make each day, the device converges to the desired behaviour, i.e., finds a policy that minimises the average AoI. At the beginning of our experiment, we randomly initialise weights in the ANN deployed on the device. The selected energy storage capacity is 4 farads, i.e.,  $B = 4F$ , and the device collects on average  $14J$  of energy each day. When ANN weights are updated only once per day, the source requires ten days to find the policy that minimises the average AoI. On the other hand, when we update ANN weights two or three times a day, the system will find the policy in only four days. Note that we achieved the numbers of transmissions per day by setting the  $T_{ANN}$ . For once-a-day ANN update, we set  $T_{ANN}$  to twenty-two hours, for two updates per day to eight hours, and three-a-day updates to six hours. Such a difference was necessary as the solar-powered source collects more energy in the afternoons. Interestingly, when we update the ANN weights more than two times per day, we can notice deterioration in the performance in the form of downtime ( $T_{down}$ ) as shown in Fig. 3 (b). The deterioration occurs as the energy required to update the ANN weights

three times a day is too high for the selected ANN size and the energy the source can collect in a day. Overall, the general behaviour of the proposed solution is as shown in Fig. 3. However, the best daily ANN weight update frequency depends on several factors, including ANN size and energy storage capacity.

#### B. Age of Information Performance

We show the performance in terms of achieved average AoI over one week of operation in Fig. 4 for the three power profiles. We define the three updating policies we use as baselines to compare the proposed solution to as:

- 1) **Unconstrained DRL**: A partially idealised approach which assumes that the source consumes zero energy to train the ANN and therefore trains directly on the device.
- 2) **Threshold**: The probability that the source will transmit is proportional to its energy level. For example, if the source has 25% of the battery left, the source has a 25% chance it will decide to transmit.
- 3) **Ideal Uniform**: This is a fully idealised approach as we assume that the source employs an oracle that provides the source with the exact information regarding the energy it will collect in the future. Using such information, the source can adopt an ideal strategy that will result in zero downtime.

The obtained simulated results of average AoI, i.e.,  $\bar{\Delta}$ , in Fig. 4 (a), (d), and (g) show that the smaller the storage capacity is and the less energy the source can collect, the more advantageous it is to use DRL. Additionally, updating



ANN weights only once per day results in performance close to the unconstrained DRL approach. The only exception is the low power-profile, where noticeable degradation occurs for capacitor sizes of 9 and 10 farads. Such degradation happens because the source does not have enough energy to transmit new ANN weights. Surprisingly, updating weights twice per day yields the worst performance in terms of average AoI, mostly due to the device consuming more energy in comparison to other approaches.

In Fig. 4 (b), (e), and (h) we show the performance in terms of *peak* AoI. The peak AoI provides a better characterisation of a process  $\Delta(t)$  as it focuses on the maximal AoI value, i.e., the worst case. Using DRL results in a lower peak AoI in comparison to other approaches, and its performance is close to that of the ideal updating policy. Such a result indicates that a DRL solution is much better at preserving the device's energy to ensure lower peak AoI. We observe similar behaviour in the downtime, which we present in Fig. 4 (c), (f), and (j). In general, the greater the  $T_{down}$ , the longer is the corresponding peak AoI. Interestingly, our DRL solution will result in a longer downtime than a threshold policy but will have lower peak AoI. Such behaviour indicates the DRL solution's superior ability to distribute the source's status updates throughout the day.

The average and peak AoI performance is linked to the number of the device's daily transmissions. For example, focusing on a case in which various updating policies achieve similar AoI performance, such as medium power profile for  $B = 5F$ , the source relying on the threshold policy will transmit 220 status updates each day. On the other hand, a source employing unconstrained DRL will send 170 status updates each day, 140 if it relies on the proposed approach with a single daily ANN weight update, and 120 transmissions if it uses two daily updates. However, a source will send only 90 status updates in an ideal policy. Such a result may come as a surprise, but as stated in [5], the source of information with limited energy might have to act lazy to achieve the optimal AoI performance.

## VI. CONCLUSION

In this paper, we have demonstrated that it is possible to facilitate a DRL-based solution on a resource-constrained embedded device powered by EH. In the proposed approach, the DRL agent that is implemented on an EH-powered device is only taking actions and sensing the environment. At the same time, the ANN training is performed on a distant unconstrained device, and the weights of the fully trained ANN are periodically transmitted to the constrained device. We have evaluated the system's performance using the average and peak AoI. We have demonstrated that the proposed periodical sending and updating of ANN weights is feasible, as the resulting performance can be comparable to the ideal solution.

In future work, we will explore the advantages that a system can gain by leveraging information from more devices. For example, as training is performed on a centralised unit, experiences from multiple sources could be combined. Moreover, a system could transmit trained ANN weights to numerous sources at the same time to preserve radio time.

## ACKNOWLEDGEMENTS

This work was funded in part by the European Regional Development Fund through the SFI Research Centres Programme under Grant No. 13/RC/2077\_P2 SFI CONNECT, the SFI-NSFC Partnership Programme Grant Number 17/NSFC/5224, and SFI Enable Grant Number 16/SP/3804.

## REFERENCES

- [1] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," *J. Parallel Distrib. Comput.*, vol. 134, pp. 75–88, Dec. 2019.
- [2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves *et al.*, "Playing Atari With Deep Reinforcement Learning," *arXiv preprint arXiv:1312.5602*, Dec. 2013.
- [3] J. Chen and X. Ran, "Deep Learning With Edge Computing: A Review," *Proced. IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019.
- [4] M. Abd-Elmagid, N. Pappas, and H. S. Dhillon, "On the Role of Age of Information in the Internet of Things," *IEEE Commun. Mag.*, vol. 57, no. 12, pp. 72–77, 2019.
- [5] R. D. Yates, "Lazy is Timely: Status Updates by an Energy Harvesting Source," in *Proc. ISIT*. Hong Kong, Jun. 2015, pp. 3008–3012.
- [6] X. Wu, J. Yang, and J. Wu, "Optimal Status Update for Age of Information Minimization with an Energy Harvesting Source," *IEEE Trans. Green Commun. Netw.*, vol. 2, no. 1, pp. 193–204, Mar. 2017.
- [7] J. Hribar, M. Costa, N. Kaminski, and L. A. DaSilva, "Using Correlated Information to Extend Device Lifetime," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2439–2448, Apr. 2019.
- [8] A. Masadeh, Z. Wang, and A. E. Kamal, "Reinforcement Learning Exploration Algorithms for Energy Harvesting Communications Systems," in *Proc. IEEE ICC*. Kansas City, KS, USA, May 2018, pp. 1–6.
- [9] F. A. Aoudia, M. Gautier, and O. Berder, "RLMan: An Energy Manager Based on Reinforcement Learning for Energy Harvesting Wireless Sensor Networks," *IEEE Trans. Green Commun. Netw.*, vol. 2, no. 2, pp. 408–417, Jun. 2018.
- [10] J. Hribar, R. Shinkuma, G. Iosifidis, and I. Dusparic, "Analyse or Transmit: Utilising Correlation at the Edge with Deep Reinforcement Learning," in *Proc. IEEE Globecom*. Madrid, Spain, Dec. 2021, pp. 1–6.
- [11] M. Chu, H. Li, X. Liao, and S. Cui, "Reinforcement Learning-based Multiaccess Control and Battery Prediction With Energy Harvesting in IoT Systems," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2009–2020, Apr. 2018.
- [12] J. Hribar, A. Marinescu, A. Chiumento, and L. A. DaSilva, "Energy Aware Deep Reinforcement Learning Scheduling for Sensors Correlated in Time and Space," *IEEE Internet Things J.*, 2021.
- [13] M. K. Sharma, A. Zappone, M. Assaad, M. Debbah *et al.*, "Distributed Power Control for Large Energy Harvesting Networks: A Multi-Agent Deep Reinforcement Learning Approach," *IEEE Trans. Cog. Commun. Netw.*, vol. 5, no. 4, pp. 1140–1154, Dec. 2019.
- [14] M. Li, X. Zhao, H. Liang, and F. Hu, "Deep Reinforcement Learning Optimal Transmission Policy for Communication Systems With Energy Harvesting and Adaptive MQAM," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 5782–5793, Jun. 2019.
- [15] M. Costa, M. Codreanu, and A. Ephremides, "On the Age of Information in Status Update Systems With Packet Management," *IEEE Trans. on Information Theory*, vol. 62, no. 4, pp. 1897–1910, Feb. 2016.
- [16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess *et al.*, "Continuous Control With Deep Reinforcement Learning," *arXiv preprint arXiv:1509.02971*, Sep. 2015.
- [17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *ICML*. Stockholm, Sweden, Jul. 2018, pp. 1861–1870.
- [18] P. Bodik, W. Hong, C. Guestrin, S. Madden *et al.*, "Intel Lab Data," *Online dataset*, Mar. 2004. [Online]. Available: <http://db.csail.mit.edu/labdata/labdata.htm>
- [19] X. Yue, M. Kauer, M. Bellanger, O. Beard *et al.*, "Development of an Indoor Photovoltaic Energy Harvesting Module for Autonomous Sensors in Building Air Quality Applications," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2092–2103, Dec. 2017.
- [20] K. Abdelfadeel, T. Farrell, D. McDonald, and D. Pesch, "How to Make Firmware Updates Over LoRaWAN Possible," in *Proc. IEEE WoWMoM*, Aug. 2020, pp. 16–25.