Multi-Attribute Monitoring for Anomaly Detection: a Reinforcement Learning Approach based on Unsupervised Reward

Mohamed Said Frikha, Sonia Mettali Gammar, Abdelkader Lahmadi

Multi-Attribute Monitoring for Anomaly Detection: a Reinforcement Learning Approach based on Unsupervised Reward

Mohamed Said Frikha*, Sonia Mettali Gammar*, Abdelkader Lahmadi†
*CRISTAL LAB, National School of Computer Sciences, ENSI, Manouba, Tunisia,
medsaid.frikha@ensi-uma.tn, sonia.gammar@ensi.rnu.tn
† Université de Lorraine, CNRS, Inria, Loria, F-54000 Nancy, France,
abdelkader.lahmadi@loria.fr

***Abstract*—This paper proposes a new method to solve the monitoring and anomaly detection problems of Low-power Internet of Things (IoT) devices. However, their performances are constrained by limited processing, memory, and communication, usually using battery-powered energy. Polling driven mechanisms for monitoring the security, performance, and quality of service of these networks should be efficient and with low overhead, which makes it particularly challenging. The present work proposes the design of a novel method based on a Deep Reinforcement Learning (DRL) algorithm coupled with an Unsupervised Learning reward technique to build a pooling monitoring of IoT networks. This combination makes the network more secure and optimizes predictions of the DRL agent in adaptive environments.**

***Index Terms*—Internet of Things, Deep Reinforcement Learning, Unsupervised Learning, Outlier detection**

I. Introduction

The Internet of Things (IoT) is blurring the line between the internet and our physical world, offering many advanced applications and services (e.g., smart cities, industrial automation, healthcare). Generally, IoT-enabled applications sense real-world assets and exchange their information among networked sensors, in particular Wireless Sensor Networks (WSNs). Including the technology of WSNs in IoT has opened several perspectives due to the ease of deploying a large number of nodes in harsh environments, as well as their features of low power consumption and low production cost. Technically, the use of unlicensed radio spectrum, for instance, the 2.4-2.48 GHz Industrial Scientific and Medical (ISM) bands, is a key feature of the current WSN solutions. These bands are then shared by many IoT networks based on wireless communication technologies such as ZigBee [1], Radio Frequency IDentification (RFID) [2], Bluetooth Low-Energy (BLE) [3], and WiFi [4].

A WSN is a self-organized network, built by a large number of distributed nodes using a multi-hop routing protocol. A typical WSN node consists of a small-size component with a limited computing unit, small memory, low communication capacity, and is battery-powered. Usually, but not limited to, WSNs applications are widely useful for environmental monitoring and data gathering [5], [6] to observe details about a local phenomenon (e.g., temperature, pressure) or analyze a detected event (e.g., intrusion detection in an interior space, object identification). Due to the limited capabilities of the sensors used in WSNs, these devices are more vulnerable to faults occurring during sensing [7]. Such faults cause disturbances that occurs due to hardware or software issues, and which may be internal or external default, such as environmental conditions. A dysfunction, an attack, or a not respected Quality of Services (QoS) requirement (e.g., delay, throughput) disrupts IoT device functionality and may have impacts on human safety.

The state of wireless sensor IoT networks, their reliability, and their security are important challenges to overcome by using an adapted network monitoring system. Outlier detection, also known as anomaly detection, is an important solution to detect any abnormal sensor behavior, such as sending or receiving a high volume of packets, rapid exhaustion of battery power, etc. Indeed, an outlier is an observation or measurement that is out from the expected range of values and which is discarded from the dataset. Several authors have attempted to formulate a definition to describe the term *'outlier'*. Hawkins et Douglas [8] define an outlier as *an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism*. Barnett et Lewis [9] propose an other definition describing an outlier as *an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data*. According to Grubbs et Frank [10] *an outlier observation is one that appears to deviate markedly from other members of the sample in which it occurs*. However, traditional anomaly detection techniques [11]–[13] cannot be suitable for sensor-based IoT networks due to their resource and communication constraints, large-scale and dynamic topology, etc.

In this paper, we propose a multi-attribute adaptive monitoring framework for detecting anomalies of low-power IoT device attributes. Our framework relies on a Reinforcement Learning (RL) method [14] which formulates network monitoring as a learning process of selecting the appropriate attributes to monitor, as well as the optimal surveillance period for each one to detect its anomalies. RL is a subset technique of Machine Learning (ML) [15], where an autonomous agent

performs a trial-and-error experiment to improve its choices in the future. Moreover, to improve the anomaly detection accuracy of our RL approach, we formulate a reward function based on an unsupervised outlier detection technique.

The rest of the paper is organized as follows. We start by listing some related work in section II. Then, section III provides a brief overview of RL and unsupervised anomaly detection algorithms. In Section IV, we describe our system model. Lastly, we conclude with some future research directions in section V.

II. Related Work

Monitoring low-power IoT networks has been always an interesting research topic over the last decade. Numerous studies have been conducted to develop smart monitoring frameworks able to address the features and constraints of IoT applications. Myridakis et al. [16] have proposed an approach considering only the deviation of the supply current as an indicator to detect security anomalies in IoT devices. This approach has been extended by Papafotikas et al. [17] using ML-based clustering algorithm for portable IoT devices. After training and clustering the normal operation mode, IoT devices become self-monitored to detect any suspicious behavior. Zhong et al. [18] have introduced an IoT enabled real-time machine status monitoring approach. Researchers use various sensors such as tags, RFID, and vibration sensors to capture machines' statuses. These IoT devices may generate enormous data of real-time operations and behaviors. Such collected data could be used to evaluate performance and generate statistical analysis reports. In [19], authors have presented a lightweight Adaptive Monitoring framework (AdaM) for smart battery-powered IoT devices. The main objective is to reduce the volumes of monitoring data generated by the IoT devices, as well as the network traffic to the management endpoints. AdaM consists of two algorithms, an adaptive sampling and an adaptive filtering algorithm, to dynamically adapt the monitoring intensity based on the variability of metrics. Results show that AdaM reduces data volume and energy consumption by more than 70% while preserving an accuracy near to 90%. The authors in [20] have developed *FailureSense* to detect sensor fail-stop, obstructed-view, and moved-location failures in smart homes. It monitors the interval between the usage of electrical appliances and the sensors trigger events, according to a Gaussian Mixture Model. However, the algorithm assumes that the resident has to be physically present to activate the electrical appliances. This hypothesis can be considered a drawback, that the system cannot be appropriate for some appliances such as boilers.

Recently, several research works have studied the problem of anomaly detection in IoT network by proposing RL-based approaches to adapt and recognize new outliers and behaviors. Gu et al. [21] consider the problem of novelty attack detection in IoT devices. To tackle the challenges of the heterogeneity of platforms, low-rate attacks, and the dynamic executed attack strategies, the authors propose a RL based IoT attack detection framework. The RL agent continuously attempts to adjust the attack detection threshold to optimize the detection rate and decrease the false alarm rate. The research study in [22] has proposed a deep Actor-Critic RL framework for anomaly detection. The system is made up of N independent processes, where each one is monitored by a different sensor that sends this information. The problem has been modeled as a partially observable Markov decision process (POMDP) since the RL agent can only observe a single sensor at a time. Therefore, the objective of the deep RL agent is to establish a policy to dynamically select the corresponding sensor in order to minimize the time slots needed for detecting the anomalies and maximize the average confidence level. In [23], the authors have developed an Intrusion Detection System (IDS) to monitor big network data generated from WSN and IoT networks. A deep RL has been combined on IDS schema to improve the decision performances of real-time monitoring.

III. Overview

In this section, we provide a brief overview about Markov Decision Processes (MDP) and Reinforcement Learning. Then we present some Unsupervised Learning methods for anomaly detection.

A. Reinforcement Learning

Reinforcement Learning is a sub-field of ML, whose goal is to determine the action sequences in order to find the optimal solution. The RL agent follows a trial-and-error process to improve its choices in the future based on past experience. Usually, RL problems are formulated as Markov Decision Process [24] model for discrete-time, stochastic, and sequential control environments. A MDP is defined as a tuple (S, A, p, r) where S is the state space of all situations that the agent can encounter; A is the action space for a state which allows the agent to change its environment; p is the transition probability of observing the state $s' \in S$ taking an action $a \in A$ from the state $s \in S$; and $r(s, a, s') \subset \mathbb{R}$ is the immediate reward value obtained, as a feedback from the environment, after an action a is performed in state s to search the state s'. The primary goal of the agent is to determine the optimal policy π^* that allows it to maximize the expected long-term cumulative discount R_t as follows:

$$R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1} \tag{1}$$

where $t = 0, 1, 2, ...$ is the time step, and $\gamma \in [0, 1)$ is the discount factor. It determines the weight of future rewards relative to those in the immediate one. Therefore, the larger γ is, the more important of the estimated future rewards will be concerned. As an example, AlphaGo is the most successful computer program playing the board game Go, one of the most complicated games, and which defeated 5-0 the European Go champion [25]. Based on RL, AlphaGo learns by itself the next move based on the current board position. After 21 days of self-learning, it reached the top human players level of all time, and defeated the previous human-champion-defeating version of AlphaGo in only 40 days by 100 games to 0.

B. Unsupervised Anomaly Detection

Unsupervised Learning [26] techniques use the data to make accurate classifications. Its tries to find the hidden structure from non-labeled input data and set it into groups of similar objects. Unlike supervised learning methods, unsupervised learning has no output associated with the input and does not receive any feedback signal during the learning process [27]. The objective is to learn how to cluster similar patterns, reduce the dimensionality, and detect outliers from data. In this work, we interest in unsupervised anomaly detection algorithms that can improve the RL agent prediction of the environment behavior and make better decisions in adaptive sampling and adaptive selecting of monitoring attributes. In the section below, we shortly introduce some unsupervised anomaly detection algorithms and their operating principle.

1) One Class Support Vector Machine (OCSVM): The OCSVM has been introduced by Schölkopf et al. [28] in 1999. It is an extension of the standard Support Vector Machine (SVM) [29] for unlabeled data sets by creating a splitting hyperplane that separates two categories in the feature space with the maximum margin. OCSVM allows gaining significant time and memory storage because only normal training samples (also called positive samples) are needed to classifier constructs [30]. Then, the new instances will be classifies as similar (inlier) or different (outlier) to the single class of the training set. Since only positive data is taken as input, the point of origin is considered as the only non-target class, and the hyperplane must be furthest from it, as shown in Fig. 1.

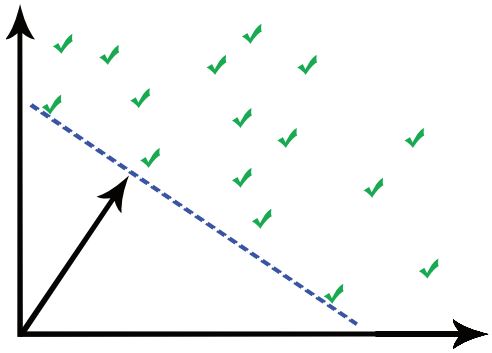

Fig. 1: Classification of the positive sampling data in the OCSVM: maximization of margin from the origin.

2) Isolation Forest (iForest): In an iForest [31], several decision trees are constructed to identify a typical values. In each tree, the easiest isolated valuers are classified as outliers, as illustrated in Fig. 2. Mathematically, an anomaly score based on branch depth for each leaf in the tree is introduced. If the new data point gets an average closer to 1, it is assessed as an outlier. With a hight dimensional data space, analyzing and eliminating the non useful information is a difficult operation. Moreover, the collected attribute values, from many applications, are unlabeled and usually include a minimal number of anomalies. Isolation Forest method is able to deal with such type of attributes, and it can be trained even with outliers in the training set.

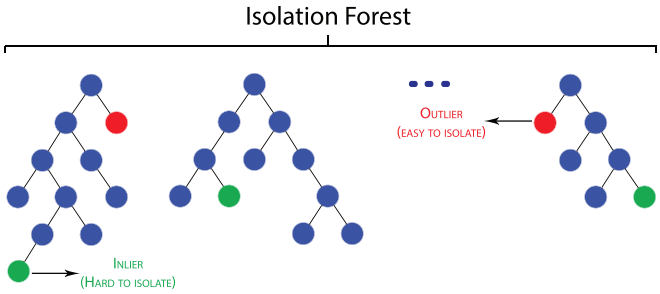

Fig. 2: Anomaly detection using iForest: different trees are generated with random partitioning of features, and outliers data are classified into leaf nodes with small paths.

3) Local Outlier Factor (LOF): The LOF [32] is an algorithm for detecting local anomalies based on the density of the neighborhood for a data object. A LOF score is computed for each data point, which reflects the degree of being an outlier. The value of the LOF score is calculated as the ratio of average local reachability density from the k-nearest neighbor. This numerical score reflects how isolated the data point is with respect to the surrounding neighborhood. As we can see in Fig. 3, a high LOF value indicates that the object is an outlier, whereas a data point with a low LOF value is expected to be normal. A LOF value greater than 1 means that the object has a lower density than neighbors and is located far from the normal samples, and thus a potential anomaly.

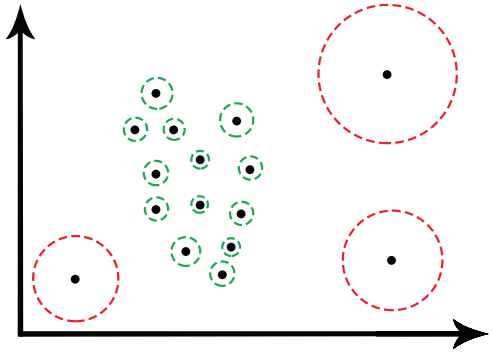

Fig. 3: Anomaly detection based on LOF scores: red circles represent outliers with high LOF values, and green circles represent normal data objects with low LOF values. The larger the radius of the circle, the more the data object is classified as an outlier.

IV. Monitoring Framework Design

The following section provides the design elements needed to understand the operation of the hybrid RL algorithm. The proposed approach is a combination of Unsupervised Learning (UL), Reinforcement Learning (RL), and Deep Learning (DL).

A. Network model

In this work, we consider a monitoring model based on a poller-pollee structure [33], applied to a low-power IoT networks. The poller node employs a monitoring process, using a DRL agent for each associated pollee, to adapt the polling interval of the monitored attributes. This allows dealing with constraints of low-power nodes (e.g., computing and energy resources) by deploying specific powerful devices as a poller in order to ensure monitoring features and respond to the deep calculation. The goal is to extend the network lifetime by avoiding oversampling, reduce the number and size of monitoring packets exchanged between pollers polles. Furthermore, the DRL agent must maintain the best results of anomalies detection and quickness reacting since the first disturbance instances.

B. State Space

Let S denote the state space, a state $s^t \in S$ of an IoT node at time t is a vector of tuples of the recuperated monitoring attributes, and is given by:

$$\begin{gathered} s^t = [(\theta^t_{att_1}, SI^t_{att_1}), \cdots, (\theta^t_{att_N}, SI^t_{att_N})] \\ \forall \theta^t_{att_n} \neq \emptyset,\ n \in [1, N] \end{gathered} \tag{2}$$

The tuples within the vector include the actual quality $\theta^t_{att_n}$ of a monitored attribute, and its current monitoring interval $SI^t_{att_n}$ selected by the RL agent, in seconds. The monitoring interval SI_{att_n} takes discrete integer values in the range $[T^{min}, T^{max}]$ defined by the administrator during the deployment of the IoT network. θ_{att_n} is a variable that indicates the quality of the polled monitoring value specified by the monitoring requirements and is given by:

- 1 : if no anomaly is detected
- -1: if one or more anomalies was detected
- 0 : if the RL agent does not collect the attribute

C. Action Space

At each decision step, the RL agent determines an action vector denoted $a \in A$ with a set of $n \in [1, N]$ elements, where n is the number of recuperated monitoring attributes. Each element is defined by the decided action taken to a specific attribute, so as to change the frequency of monitoring. Specifically, we define five possible actions as follows:

$$A = \{a_D, a_K, a_I, a_S, a_A\} \tag{3}$$

where a_D is the action to decrease the monitoring interval, a_K to keep the current interval , a_I to increase the monitoring interval, a_S to stop collecting an attribute, and a_A the action to reactivate the collect of an attribute such as $SI_{att_n} = T^{min}$.

Depending on each attribute state, the RL agent has the ability to take one action according to the following rules:

- $SI_{att_n} = T^{min} \implies A = \{a_K, a_I\}$
- $SI_{att_n} = T^{max} \implies A = \{a_D, a_K, a_S\}$
- $SI_{att_n} = \infty \implies A = \{a_K, a_A\}$
- *Otherwise* $\implies A = \{a_D, a_K, a_I\}$
- *The RL agent can deactivate the collection of at most $N-1$ attributes at the same time.*

The last rule will avoid the RL system from falling into the blocking case where there is no attribute to be collected, and therefore, no new states to continue the trial-error process. For that, the agent must keep at least one active monitoring attribute (e.g., a state vector with one tuple). Furthermore, the RL agent will learn the optimal attribute(s), which must be always monitored. It will be able to detect faster node anomalies by reactivating the monitoring of one or more attributes based on the optimal activate attribute status. Furthermore, to avoid abrupt changes, actions can only move to neighboring values of the monitoring interval.

D. Reward Function

The reward function is the numerical fed back, which refers to the evaluation of being in a new state s^{t+1} taken an action $a \in A$ given a state s^t according to the following policy. The main objective of the proposed reward function is to found the trade-off between the cost, in terms of energy consumption by the monitoring system, and the efficiency of detecting anomalies in attribute values. The RL rewards $r(s^t, a)$ is given by the following equation:

$$\begin{aligned} r(s^t, a) = \sum_{j=1}^{N} \sum_{\substack{k=1 \\ k \neq j}}^{N} & \underbrace{\beta_{r_j}}_{\mathbf{r_1}} + \underbrace{((\omega_{att_{j,k}} \times \theta_{att_k}) \times \frac{SI_j}{T^{min}})}_{\mathbf{r_2}} \\ & - \underbrace{(\theta_{att_j} \times \frac{SI_j}{T^{min}})}_{\mathbf{r_3}} \end{aligned} \tag{4}$$

The reward function can be disintegrated into three main parts as under-braced above:

- $\mathbf{r_1}$: is the state of the monitored attribute during SI_j
- $\mathbf{r_2}$: is the expected future state of attribute j based on the state of the other attributes.
- $\mathbf{r_3}$: is a punishment value of how bad to collect or not this attribute.

Since the values of $\mathbf{r_2}$ and $\mathbf{r_3}$ are approximate, we increase their weights by multiplying them at the monitoring interval selected by the RL agent.

To recover the β_{r_j} value, the RL agent exploits an unsupervised anomaly detection technique. During the trained phase, the unsupervised learning model learns to identify anomalies based on the percentage of the variance between the new instance value and its predecessor. The training model takes as input the attribute values at instant $t-1$ and t and returns as output -1 for outliers and 1 for inliers.

In our approach, the current state of the attribute as well as the future quality estimation, based on the trends in attributes' dependency between them, are mainly used to calculate the immediate reward function. To this end, the RL agent maintains the estimation weight values for each attribute tuple in a vector and puts them update in each new state for an attribute as described in algorithm 1. The main idea is to make the agent

Algorithm 1 Attributes' weight update function for $attr_N$

Require:
$\Theta = \{\theta_{attr_1}, \ldots, \theta_{attr_N}\}$: attributes' new state
$\Omega_N = \{\omega_{attr_{N,1}}, \ldots, \omega_{attr_{N,N-1}}\}$: attributes' weight

if $\theta_{attr_N} = 1$ **and** $values\ of\ \Theta \setminus \{\theta_{attr_N}\}\ are\ not\ equal$ **then**
  $index_{min} \leftarrow index\ of\ the\ minimum\ value\ in\ \Omega$
  $\Omega_{anom} \leftarrow \{\omega_{attr_{N,n}}\}\ ;\ \forall n \in [1, N-1]$ **and** $\theta_{attr_n} = 1$

  // Compute change percentage;
  if $\Theta[index_{min}] \neq 1$ **then**
    $\Delta \leftarrow \dfrac{\Omega[index_{min}]}{(N-1) \times length(\Omega_{anom})} \times \alpha'$
  else
    $\Delta \leftarrow \dfrac{\Omega[index_{min}]}{[(N-1) \times length(\Omega_{anom})] - 1} \times \alpha'$
  end if

  // Update weight values;
  for $i \leftarrow 1$ **to** $N-1$ **do**
    if $\Theta[i] = 1$ **then**
      $\omega_{attr_{N,i}} \leftarrow \omega_{attr_{N,i}} + \Delta$
    else
      $\omega_{attr_{N,i}} \leftarrow \omega_{attr_{N,i}} - \dfrac{\Delta \times length(\Omega_{anom})}{(N-1) - length(\Omega_{anom})}$
    end if
  end for
end if
return Ω_N

able to approximate the operating link between the different couple monitoring attributes (e.g., the possibility that a specific attribute will be in an anomaly state if another attribute is already in an anomaly state.). Initially, the RL agent assigns the same link probability values for each couple attributes in its specific vector, denoted Ω, as follows:

$$\forall (attr_i, attr_j)\ ,\ \omega_{attr_{i,j}} = \frac{1}{N-1} \tag{5}$$

where N is the number of the monitoring attributes, $i, j \in [1, N]$, and $i \neq j$. At each step, the RL agent collects the new state of the attributes and executes the following algorithm only for those identified as in an anomaly state. The RL agent compute the change percentage based on the minimal attributes' weight in the Ω vector. If the attribute in the vector Θ is in an anomaly state, the RL agent updates the weight by increasing its link probability value; Otherwise, the weight decreases. To reduce the update estimation error rate, we multiply the change percentage by a small value $\alpha' \in [0, 1]$. This hyperparameter controls the speed, on time, at which the agent promotes a specific link relative to the others. After a long run, the link weight of the independent attributes will converge to zero.

E. Neural Network Architecture

The DRL agent takes an environment observation as a vector of the state of each monitoring attribute. This makes the state space very large and grows exponentially with the number of monitoring attributes, with a complexity in order of $\mathcal{O}(m^N)$, where m is the number of valid possible status tuple combinations of a monitoring attribute and N is the number of the monitoring attributes. In this case, it's not practical to represent the Q-function as a table containing values for each combination of state s and action a. To deal with such a situation, Deep RL trains an approximation function using a neural network (NN) to approximate the Q values. Fig. 4 represents the structure of the modeled DRL with multiple layers, where the NN takes the observation environment state as input and returns as output a vector of actions.

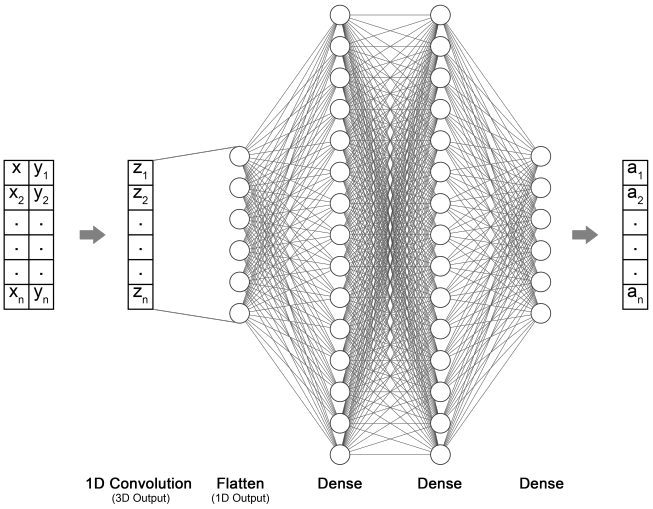

Fig. 4: A DRL diagram with One-Dimension (1D) Convolutional layer, a Flatten layer, and two hidden layers. This figure was drawn in part using models in https://alexlenail.me/NN-SVG/ .

In order to assume the hight dependency between θ_{att_i} and SI_{att_i} for each input attribute state on the final decision, we use the 1D convolution layer. Mathematically, a convolution is an integral transform between two functions f and g, denoted as $f * g$, where one represents the input feature, and the other denotes the kernel. The output function expresses the weighting sum of the input elements shifting over the kernel. To transform the 3D output of the convolution 1D layer into a 1D array of elements, a flatten layer is used as a dimension reduction process. It reshapes the input data by removing all dimensions, except for one, and preserving the same number of elements. The hidden fully connected layer receives a single vector from the previous layer, such as all the output are connected to every activation unit of the next layer.

V. Conclusion and Future Work

In this paper, we proposed a new multi-attribute adaptive monitoring technique based on DRL for detecting anomalies of low-power IoT device. We then introduced a novel

reward function based on unsupervised learning to improve environment behavior prediction. Our future work consists of conducting a series of experiments to study Neural Network parameters settings and their effects on the RL agent monitoring operation. Choosing the unsupervised learning technique and the appropriate DRL agent (DQN, DDPG, etc.) also requires an assessment to understand which configuration adapted to the proposed solution.

REFERENCES

[1] "Zigbee - zigbee alliance," [Accessed on June 2021]. [Online]. Available: https://zigbeealliance.org/solution/zigbee/

[2] C. M. Roberts, "Radio frequency identification (rfid)," *Computers & security*, vol. 25, no. 1, pp. 18–26, 2006.

[3] K. Townsend, C. Cufí, R. Davidson *et al.*, *Getting started with Bluetooth low energy: tools and techniques for low-power networking*. " O'Reilly Media, Inc.", 2014.

[4] M. Park, "Ieee 802.11 ah: sub-1-ghz license-exempt operation for the internet of things," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 145–151, 2015.

[5] M. T. Lazarescu, "Design of a wsn platform for long-term environmental monitoring for iot applications," *IEEE Journal on emerging and selected topics in circuits and systems*, vol. 3, no. 1, pp. 45–54, 2013.

[6] P. P. Raj, A. M. Khedr, and Z. Al Aghbari, "Data gathering via mobile sink in wsns using game theory and enhanced ant colony optimization," *Wireless Networks*, vol. 26, no. 4, pp. 2983–2998, 2020.

[7] J. Chen, S. Kher, and A. Somani, "Distributed fault detection of wireless sensor networks," in *Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks*, 2006, pp. 65–72.

[8] D. M. Hawkins, *Identification of outliers*. Springer, 1980, vol. 11.

[9] V. Barnett and T. Lewis, "Outliers in statistical data," *osd*, 1984.

[10] F. E. Grubbs, "Procedures for detecting outlying observations in samples," *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969.

[11] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, 2009, pp. 117–132.

[12] M. V. Mahoney and P. K. Chan, "Learning nonstationary models of normal network traffic for detecting novel attacks," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 376–385.

[13] P. Barford and D. Plonka, "Characteristics of network traffic flow anomalies," in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, 2001, pp. 69–73.

[14] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2017, http://incompleteideas.net/book/RLbook2018.pdf.

[15] T. M. Mitchell *et al.*, *Machine learning*. McGraw-Hill, 1997, no. 432.

[16] D. Myridakis, G. Spathoulas, A. Kakarountas, D. Schoinianakis, and J. Lueken, "Anomaly detection in iot devices via monitoring of supply current," in *2018 IEEE 8th International Conference on Consumer Electronics-Berlin (ICCE-Berlin)*. IEEE, 2018, pp. 1–4.

[17] S. Papafotikas and A. Kakarountas, "A machine-learning clustering approach for intrusion detection to iot devices," in *2019 4th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*. IEEE, 2019, pp. 1–6.

[18] R. Y. Zhong, L. Wang, and X. Xu, "An iot-enabled real-time machine status monitoring approach for cloud manufacturing," *Procedia CIRP*, vol. 63, pp. 709–714, 2017.

[19] D. Trihinas, G. Pallis, and M. D. Dikaiakos, "Adam: An adaptive monitoring framework for sampling and filtering on iot devices," in *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, 2015, pp. 717–726.

[20] S. Munir and J. A. Stankovic, "Failuresense: Detecting sensor failure using electrical appliances in the home," in *2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems*. IEEE, 2014, pp. 73–81.

[21] T. Gu, A. Abhishek, H. Fu, H. Zhang, D. Basu, and P. Mohapatra, "Towards learning-automation iot attack detection through reinforcement learning," in *2020 IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2020, pp. 88–97.

[22] C. Zhong, M. C. Gursoy, and S. Velipasalar, "Deep actor-critic reinforcement learning for anomaly detection," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[23] H. Benaddi, K. Ibrahimi, A. Benslimane, and J. Qadir, "A deep reinforcement learning based intrusion detection system (drl-ids) for securing wireless sensor networks and internet of things," in *International Wireless Internet Conference*. Springer, 2019, pp. 73–87.

[24] R. A. Howard, "Dynamic programming and markov processes." 1960.

[25] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[26] Z. Ghahramani, "Unsupervised learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 72–112.

[27] Y.-Z. Lu, *Industrial intelligent control: fundamentals and applications*. John Wiley & Sons, 1996.

[28] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, J. C. Platt *et al.*, "Support vector method for novelty detection." in *NIPS*, vol. 12. Citeseer, 1999, pp. 582–588.

[29] N. Cristianini, J. Shawe-Taylor *et al.*, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[30] W. Li, Q. Guo, and C. Elkan, "A positive and unlabeled learning algorithm for one-class classification of remote-sensing data," *IEEE transactions on geoscience and remote sensing*, vol. 49, no. 2, pp. 717–725, 2010.

[31] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.

[32] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104.

[33] L. Li, M. Thottan, B. Yao, and S. Paul, "Distributed network monitoring with bounded link utilization in ip networks," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, vol. 2. IEEE, 2003, pp. 1189–1198.