# A Comparative Study of AI-Based Intrusion Detection Techniques in Critical Infrastructures

SAFA OTOUM, College of Technological Innovation, Zayed University, UAE
BURAK KANTARCI and HUSSEIN MOUFTAH, University of Ottawa

Volunteer computing uses Internet-connected devices (laptops, PCs, smart devices, etc.), in which their owners volunteer them as storage and computing power resources, has become an essential mechanism for resource management in numerous applications. The growth of the volume and variety of data traffic on the Internet leads to concerns on the robustness of cyberphysical systems especially for critical infrastructures. Therefore, the implementation of an efficient Intrusion Detection System for gathering such sensory data has gained vital importance. In this article, we present a comparative study of Artificial Intelligence (AI)-driven intrusion detection systems for wirelessly connected sensors that track crucial applications. Specifically, we present an in-depth analysis of the use of machine learning, deep learning and reinforcement learning solutions to recognise intrusive behavior in the collected traffic. We evaluate the proposed mechanisms by using KDD'99 as real attack dataset in our simulations. Results present the performance metrics for three different IDSs, namely the Adaptively Supervised and Clustered Hybrid IDS (ASCH-IDS), Restricted Boltzmann Machine-based Clustered IDS (RBC-IDS), and Q-learning based IDS (Q-IDS), to detect malicious behaviors. We also present the performance of different reinforcement learning techniques such as State-Action-Reward-State-Action Learning (SARSA) and the Temporal Difference learning (TD). Through simulations, we show that Q-IDS performs with 100% detection rate while SARSA-IDS and TD-IDS perform at the order of 99.5%.

CCS Concepts: • **Security and privacy → Intrusion detection systems**; • **Networks → Network simulations**; **Network performance analysis;**

Additional Key Words and Phrases: Intrusion detection, machine learning, deep learning, restricted Boltzmann machine, reinforcement learning, wireless sensor networks

**ACM Reference format:**
Safa Otoum, Burak Kantarci, and Hussein Mouftah. 2021. A Comparative Study of AI-Based Intrusion Detection Techniques in Critical Infrastructures. *ACM Trans. Internet Technol.* 21, 4, Article 81 (July 2021), 22 pages.
https://doi.org/10.1145/3406093

## 1 INTRODUCTION

Smart cities exploit the flexibility, self-deployment and low-cost benefits of wireless sensors to monitor their critical infrastructures [1]. Monitoring critical infrastructures involves various types

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) DISCOVERY Program.
Authors' addresses: S. Otoum, College of Technological Innovation, Zayed University, UAE, Abu Dhabi, UAE; email: Safa.Otoum@zu.ac.ae; B. Kantarci and H. Mouftah, University of Ottawa, 800 King Edward Ave, Ottawa, ON, K1N6N5; emails: {Burak.Kantarci, Mouftah}@uottawa.ca.

1533-5399/2021/07-ART81 $15.00
https://doi.org/10.1145/3406093

of sensors under different circumstances [2, 3]. Efficiency and reliability of critical infrastructures such as medical infrastructure can be met by using reliable and secure data aggregation and transmission [4, 5]. Sensors and the communication lines between them are vulnerable to numerous intruders that may interrupt and manipulate with the communication and transmitted data [6, 7]. IDS is an essential component for network security to detect different types of intrusions. At the very high level, the validity of any IDS is determined by its ability of raising an alarm with the detection of any malicious node that exhibits intrusive behavior [8]. IDSs are categorised in two sets: Anomaly-based IDS and Signature-based IDS. The former aims to detect abnormal traffic patterns that deviate from the normal behaviour. To this end, it creates profiles of the features to seize the needed patterns. The variance between the extracted patterns and noticed activities may lead to an alarm. Thus, ideally Anomaly-based IDSs are able to detect unknown attacks. However, It is worth noting that most IDS systems still suffer from False Positive (FP) decisions, i.e., a non-malicious activity's being marked as an intrusive event [6]. Apart from the anomaly-based IDS, signature IDSs uses the rule complement mechanism to detect malicious behavior by comparing system activities with specific rules. An intrusive event is detected when the noticed activities match a known malicious pattern. The risk of solely adopting a signature-based IDS is that its possible low precision in the likelihood of an unknown attack [6, 8]. In our previous work, we proposed several artificial intelligence (AI)-based IDS methods for WSNs monitoring critical infrastructures [7, 8]. All of these approaches have the following aim in common: signature detection and anomaly detection which can be combined in an IDS by following an adaptive supervision procedure. In this article, we provide a comparative study of intrusion detection in critical infrastructures which compares the usage of machine learning (random forest and E-DBSCAN), deep learning (restricted Boltzmann machine) and reinforcement learning approaches (Q-learning). We also develop our work presented in [6–8] by introducing Q-learning, SARSA-learning and TD-learning based reinforcement procedures to our previously proposed IDS. The main contributions of this work are mentioned as follows.

(1) This work directly target the volunteer computing by applying the previously proposed IDSs to the distributed data and evaluating the IDS performance.
(2) It presents an intrusion detection mechanism, in which reinforcement learning using Q-learning, SARSA learning, and TD learning methods are introduced to estimate the IDS performance.
(3) For the first time, we present a comprehensive and comparative study of IDS evaluation using machine, deep and reinforcement learning methods.

## 2 MOTIVATIONS

WSNs are major technology in critical infrastructures, especially in monitoring and control processes, due to their self-deployment, flexibility and low-cost benefits. Monitoring critical infrastructures involve numerous sensors under various phenomena. In most of critical infrastructures, WSNs are deployed in an open nature environment, thus, nodes could be vulnerable to different attacks which can manipulate the aggregated sensory data. The communication lines are also vulnerable to numerous intruders that may interrupt and manipulate with the communication and transmitted data. WSNs need to provide an acceptable level of security to achieve attacks free environments, reliable and secure data aggregation and transmission [4]. Intrusion Detection System (IDS) is an essential component for network security especially for the sensor networks to detect different types of intrusions. At the very high level, the validity of any IDS is determined by its ability to raise an alarm with the detection of any malicious node that exhibits intrusive behavior [8].

The rest of this article is organised as follows: Section 3 layouts this study's related works. Section 4 introduces the IDS topology settings. Section 5 describes the methods used to test the previously proposed IDS. Experimental and results analysis are presented in Section 6. Finally, we present the conclusion in Section 7.

## 3 RELATED WORKS

Integration of machine learning with IDS have been proposed by many researchers. Some of them tackled the unknown attacks [9] while others tackled the known ones [10]. Lately, deep learning-based techniques had been used in IDSs and proved their effectiveness in detection-intrusive behaviors [11, 12].

### 3.1 Machine Learning-Based IDS

The goal of any IDS is to identify two different behavior patterns: normal behavior and malicious behavior [13]. Popular machine learning algorithms that aim to cluster behavior patterns around a centroid are K-nearest neighbour and K-means [14]. As a supervised approach, **Support Vector Machines (SVM)** have also been be considered [15], which partitions the data plane into smaller hyperplanes and also automates feature selection.

Adaptive machine learning-based techniques for IDS have been tested in many studies to improve the accuracy of classification such as the work presented in [16] where the researchers presented a validation procedure by presenting the automated and adaptive testing prototype. the work in [17] proposed an **Adaptive Model Generation (AMG)** as a model generator which would work adaptively in real time to perform IDSs. AMG enables evaluation of data in real time, and automates the data collection, as well as the generation and deployment of detection models.

To the best of our knowledge, an adaptive-IDS solution for critical infrastructures that deal with both known and unknown intruders remains an open issue. In ASCH-IDS, we present a dynamic adjustment technique for the proportion of the aggregated sensory data that forwarded to the anomaly and misuse detection subsystems.

### 3.2 Deep Learning-Based IDS

Deep learning methods have been applied to IDS and achieved highly accurate results [18–20]. In [21], Alom et al. tested the abilities of a **Deep Belief Network (DBN)** in the detection of intrusive patterns. Salama et al. [18] combined DBN and SVMs, and introduced a hybrid IDS methodology where DBN served as a feature selector and SVM as the classifier. The hybrid approach resulted in ≈92% accuracy rate. Fiore et al. [22] presented a partially supervised learning approach where the classifier was trained with normal traffic only so any knowledge about malicious behaviors could evolve dynamically. Fiore et al. applied **Discriminative Restricted Boltzmann Machine (DRBM)** to anomaly detection as an energy-based classifier.

High dimensionality is a grand challenge in big data applications; hence, Li et al. [23] applied an autoencoder in the first stage of an IDS in order to reduce dimensionality and extract the features for a DBN to classify anomalous and normal behavior patterns.

Deep learning has been introduced to distributed computing such as in the works presented in [24] and [25]. In [24], Abeshu and Chilamkurti proposed a cyber-attack detection deep learning-based mechanism in fog-to-things computing. The work in [25] proved the flexibility of cloud-based infrastructures along with the enhancements in the large-scale machine learning area for moving the highest computationally and storage tasks to the cloud to benefit the edge computing capabilities.

### 3.3 Reinforcement Learning Based IDS

**Reinforcement Learning (RL)** is considered as an extension to machine learning, and involves agents that take actions to maximise the notion of rewards. Previously, the reinforcement learning process was applied to IDSs by several studies. In [26], Servin and Kudenko proposed a distributed reinforcement learning approach in which each agent (i.e., sensor) analyzes state observations and communicates them to a central agent. Agents that are higher in the hierarchy are equipped with the knowledge for analyzing the collected data, and they broadcast an overall abnormal state to the network operator [26]. Xu and Xie [27] presented reinforcement learning-based technique for host-based IDS through a series of system calls by presenting a **Markovian reward process (MRP)** to replicate the behavior of system call series where the intrusion detection issue is transfigured to predict the MRP value functions. The work presented in [28] proposed an approach to detect intrusion with online clustering by using **Pursuit Reinforcement Competitive Learning (PRCL)**. Georgia [29] presented an approach of an adaptive neural networks to intrusion detection that detect new attacks autonomously with the use of reinforcement learning method. The work presented in [30] exploits the use of data flow information, Reinforcement Learning and tile coding to detect flooding-based **Distributed Denial of Service (DDoS)** attacks.

To this end, a comprehensive comparison/evaluation of IDS for critical monitoring infrastructures that works for both known and unknown attacks using Q-based, **State-Action-Reward-State-Action Learning (SARSA)** and the **Temporal Difference learning (TD)** reinforcement learning remains an open issue.

## 4 SYSTEM MODEL

We adopt the IDS model that was proposed for WSN-based critical infrastructure monitoring in our previous work [6] to the volunteer computing architecture.

The IDS consists of $N$ clusters with $C$ sensors in each cluster. Aggregating the sensory data is the responsibility of the Cluster Head (CH) which collects the data from sensors and directs it to the IDS which is installed in a central node namely the server. The aggregated data then undergoes the intrusion detection method. The IDS consists of (1) Weighted cluster head selection in Section 4.1, (2) Data aggregation in Section 4.2, and (3) The detection methods in Section 5. The notations used all through the article are listed in Table 1.

### 4.1 Weighted Cluster Head (CH) Selection

A **Cluster-Head (CH)** election method has been adopted for data collection and processing reasons. CH election method is fulfilled by using the election technique that calculates the weight of each sensor node and compares it to the weights of the other nodes [31]. Each node has a weight $G_n$ that is a function of its ($RSSI_{sum}$) which refers to its received signal strength, mobility, and degree. Following the weight computation, the node broadcasts its weight along with its unique **identifier (ID)**. It then proceeds to compare it with the weights of the neighbour nodes such as the sensor node that achieves the lowest weight is elected as the CH [6, 8, 31].

The weight depends on the following four elements: mobility and degree difference (which refers to the node degree, excluding the CH capacity), cumulative time, and $RSSI_{sum}$. The election procedure goes through steps as shown in Algorithm 1, where $D_n$ refers to the degree of node $n$, $\Gamma$ represents the CH Capacity *(*i.e., *number of nodes)*, $\partial n$ refers to the degree difference for $n$, $SRS_{strength}(n)$ represents the sum of $n$'s received signal strength, $\check{M}_n$ represents the mobility factor for node $n$, $\hbar_n$ refers to the cumulative time for node $n$, and $G_n$ represents the combined weight for node $n$. In Algorithm 1, $n$ refers to any node, $g_{\hbar_n}$, $g_m$, $g_{sum}$, and $g_d$ refer to the weight factors of $\hbar_n$, $\check{M}_n$, $RSSI_{sum}$, and $\partial n$, accordingly.

### 4.2 Data Aggregation Procedure

Each CH gathers data from its corresponding sensors and sends them to the sink. The method presented in [32] has been adopted in our previous and ongoing research as the aggregation procedure. The method depends on assessing the trust score of the aggregator by tracking the sensors' trust scores with the estimated trust between sensors and the aggregator [32]. Equation (1) computes the trust score of a CH [32] where $T_{CH}$ denotes the CH trust value, $T_n$ is node $n$ trust value, and $T_{CH}^n$ is the CH and node $n$ trust estimation

$$T_{CH} = \frac{(\sum_{n=0}^{n-1}(T_n + 1) \cdot T_{CH}^n)}{\sum_{n=0}^{n-1}(T_n + 1)} \tag{1}$$

**ALGORITHM 1:** Weighted Cluster Head (CH) selection pseudo-code

```
procedure WEIGHTED CH SELECTION
Input: D_n, Γ, |1/SRS_strength(n)|, M̌_n, ħ_n
Output: G_n
for each node n do
        ∂n = |D_n − Γ|
|1/SRS_strength(n)|: normalised value of SRS_strength
G_n = g_d ∂n + g_sum / |1/SRS_strength(n)| + g_m M̌_n + g_ħn ħ_n
    EndFor
Return G_n
Select the node with a minimum G_n as CH
Remove CH from the nodes list
Repeat all steps for the remaining nodes
End
end procedure
```

## 5 INTRUSION DETECTION METHODS

### 5.1 Adaptive Machine Learning Based the Adaptively Supervised and Clustered Hybrid Intrusion Detection System (ASCH-IDS)

In our previously proposed ASCH-IDS [7] and CHH-IDS [6], the collected sensed data experiences two machine learning-based subsystems working in parallel for intrusion detection: Anomaly Detection Subsystems (*ADSs*) and Misuse Detection Subsystems (*MDSs*). This translates into a hybrid system as shown in Algorithm 2 for detecting intrusive sensors. *ADSs* follows the E-DBSCAN algorithm which refers to the Enhanced-Density Based Spatial Clustering of Applications with Noise while *MDSs* follows the random forest technique. DBSCAN returns to a density based clustering method where clusters are considered as dense areas of objects where the data space is distributed into areas of objects with low densities [34] while the E-DBSCAN algorithm retains the track of the variation of local density in the cluster and computes any core objects' density variance with considering its e-neighbourhood [35]. On the other hand, when random forest used as a classification mechanism in which each tree checks each input for the frequent classes and delivers a vote if it happened [36], it operates in two stages: the training and the classification stages [37]. In [6], the collected data is partitioned into IDS subsystems (IDSs) in a round-robin fashion following a time-slotted method, as shown in Figure 1.

Figure 2 represents the previously proposed CHH-IDS [6] where CHH-IDS runs on a WSN which contains $N$ clusters where the CH handles the aggregation procedure of the data forwarded by the sensors.

Table 1. Notations Used in the Article

| Notation | Description |
|---|---|
| $V$ | Visible element |
| $H$ | Hidden element |
| $O$ | Outputs |
| $X$ | Number of visible nodes |
| $Y$ | Number of hidden nodes |
| $a_x$ | Visible bias |
| $b_y$ | Hidden bias |
| $W$ | The weights between visible and hidden layers |
| $W_{xy}$ | Combined weights of visible $V_x$ and hidden $H_y$ units |
| $\Theta$ | RBM parameters ($W_{xy}$, $a_x$, $b_y$) |
| $E(V, H\|\Theta)$ | The energy function of the RBM |
| $P(V, H)$ | The probability of $(V, H)$ formation |
| $\sum_{X,Y} e^{-E(V,H)}$ | The normalisation factor (all possible configurations including the visible and hidden elements) |
| $P(V)$ | The probability allocated to any visible element $V$, The network sets probability score to each case in hidden and visible elements [19, 33] |
| $P(H)$ | The probability allocated to any hidden element $H$ |
| $P(V\|H)$ | The probability of $V$ independently with $H$ |
| $P(H\|V)$ | The probability of $H$ independently with $V$ |
| $P(Intrusive)$ | The binary output for intrusive detection |
| $Normal$ | The binary output for normal detection |
| $rec$ | Number of records |
| $Var$ | Number of variables |
| $T_r$ | Number of trees |
| $TP$ | True Positive |
| $TN$ | True Negative |
| $FP$ | False Positive |
| $FN$ | False Negative |
| $S_t$ | State at time $t$ |
| $A_t$ | Action at time $t$ |
| $R_t$ | Reward at time $t$ |
| $V^{\pi}(S)$ | Value estimation of $R$ that at initial state $S$ |
| $\pi(S, A)$ | Probability of $A$ in $S$ |
| $P_{SS^+}(A)$ | Transitional probability from state $S$ to $S^+$ at $A$ |
| $R(S, S^+, A)$ | Reward returned from transition from state $S$ to $S^+$ at $A$ |
| $r$ | Discount factor weight from future rewards to current rewards |
| $V_I^{\pi}(S^+)$ | Value estimation of $R$ at state $S^+$ at the initial iteration $I$ |
| $V_{I+1}^{\pi}(S)$ | Value estimation of $R$ at state $S$ at the updated iteration $I + 1$ |
| $T_{CH}$ | CH trust value |
| $T_n$ | Node $n$ trust value |
| $T_{CH}^n$ | CH and node $n$ trust estimation |
| $\alpha$ | The weight of the recently calculated $TP/FP$ and its value during the $(\triangle\tau)$ |

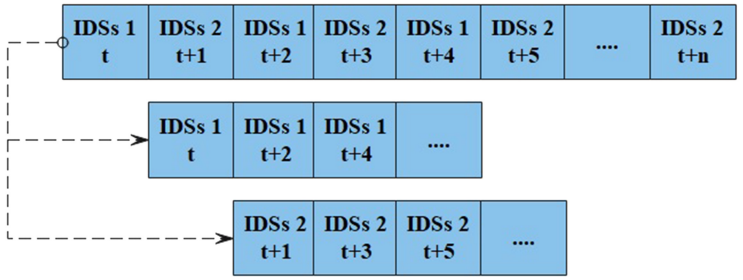


Fig. 1. Aggregated sensed data distribution mechanism between the IDSs1 and IDSs2.

**ALGORITHM 2:** Detection of intrusive sensors

```
procedure INTRUSIVE SENSORS DETECTION
Cluster Head (CH) selection: Weighted CH selection
Data aggregation: Trust-based aggregation of sensed data
Data distribution: Aggregated data distribution between the two sub-systems
if (Intrusive behavior detected) then
    ALARM
END
else if (End of sensing data) then
    END
else
    Go to Data distribution
end procedure
```

When random forest is used for misuse detection, each tree is developed as described below [37]:

- If the training set size is denoted by $Y$, randomly extracted data-points $y$ from the original dataset turn into the training dataset for the developing tree.
- If the input variables are denoted by $X$, randomly extracted data-points $x$ from $X$ can be utilised to split the node. $x$ is considered as a constant while the forest develops where each tree will be built to its most size.

On the one hand, E-DBSCAN is used as a clustering algorithm that has been utilised to extract the $\epsilon$ which represents the distance threshold, $\epsilon$ is a critical factor in E-DBSCAN algorithm [38]. DBSCAN can realise the adjacent clusters which belong to the same density as well as clusters of random shapes [34]. DBSCAN consists of two factors, the $\epsilon$ and the $MinPts$ which are considered as input parameters. It follows the rules described below:

- $N_\epsilon(x) = y \in X | d(x, y)| \leq \epsilon$ is the $\epsilon - neighbourhood$ of point $x$.
- Neighbourhoods of the core object has a size $> MinPts$.
- A point $j$ is density accessible from $i$ as a core object.
- $i$ and $j$ considered as density-base -connected when $i$ and $j$ are density accessible from a core object.

On the other hand, the ASCH-IDS [6] retains the track of *ADSs* and *MDSs* deviations in the **Receiver Operating Characteristics (ROC)** and changes the segment of data directed to them adaptively.

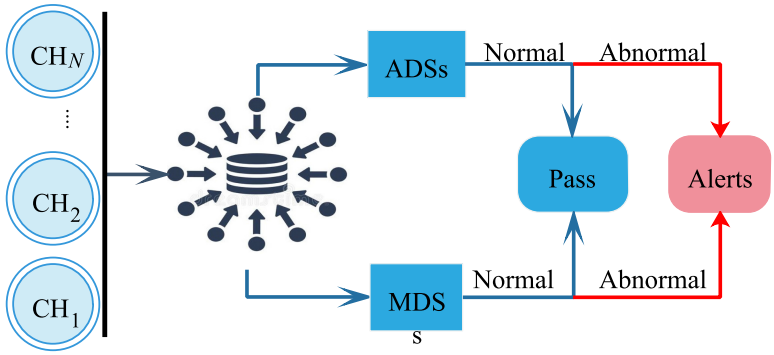


Fig. 2. CHH-IDS system model. The collected sensed data is distributed between the ADSs and MDSs.

Equations (2)–(3) represent the ($TP$) to ($FP$) for the two subsystems at $\tau_i$ which are represented by $\mu_1(\tau_i)$ and $\mu_2(\tau_i)$ [7].

$$\mu_1(\tau_i) = \frac{TP_1(\tau_i)}{FP_1(\tau_i)} \tag{2}$$

$$\mu_2(\tau_i) = \frac{TP_2(\tau_i)}{FP_2(\tau_i)} \tag{3}$$

Equations (4)–(5) represent the $TP$ to $FP$ ratios at time ($\triangle\tau$) where $\triangle\tau$=$\tau_{i+1} - \tau_i$ [7].

$$\mu_1(\triangle\tau) = \frac{TP_1(\triangle\tau)}{FP_1(\triangle\tau)} \tag{4}$$

$$\mu_2(\triangle\tau) = \frac{TP_2(\triangle\tau)}{FP_2(\triangle\tau)} \tag{5}$$

When the ROC behaviour during $\triangle\tau$ is extracted, the ROC behaviour of each subsystem can be obtained as the sum of the ROC behaviour at $\tau$ and the behaviour at $\triangle\tau$ as represented in Equations (6) and (7) [7], where $\alpha$ represents the weight of the recently calculated $TP/FP$ and its value during the ($\triangle\tau$) where $\tau_{i+1} = \tau_i + \triangle\tau$.

$$\mu_1(\tau_{i+1}) = \alpha\mu_1(\tau_i) + (1-\alpha)\mu_1(\triangle\tau) \tag{6}$$

$$\mu_2(\tau_{i+1}) = \alpha\mu_2(\tau_i) + (1-\alpha)\mu_2(\triangle\tau) \tag{7}$$

Equation (8) represents an indicator $I(\tau_i)$ which tracks the ROC behavior of MDSs and ADSs at any time $\tau_i$:

$$I(\tau_i) = \frac{\mu_1(\tau_i)}{\mu_2(\tau_i)} \tag{8}$$

The extracted indicator $I$ is used in the decision of directing the collected data such as if $I(\tau_i) > I(\tau_{i-1})$, ASCH-IDS decides that $ADSs$ is performing better than $MDSs$, so any increasing of data proportion on $ADSs$ is enhancing the overall performance of the system. On the other hand, if $I(\tau_i) < I(\tau_{i-1})$, the intrusion system determines that $MDSs$ is performing better than $ADSs$, such as, any increase in the data proportion on $MDSs$ is expected to improve the overall performance of the system. wherever, if $I(\tau_{i+1}) > I(\tau_i)$, the ASCH-IDS increases the data proportion on $\mu_1$ and decreases on $\mu_2$ such as: $D_a(\tau_{i+1}) = D_a(\tau_i) + \triangle D$ and $D_m(\tau_{i+1}) = D_m(\tau_i) - \triangle D$ as formulated in (9)–(10) where $\triangle D$ denotes the data adjustment for each subsystem. Algorithm 3 presents an overview of the previous steps. Where $\tau_i$,$\tau_{i+1}$ and $\triangle\tau$ refer to any time, $\tau_i + \triangle\tau$ and the time difference between ($\tau_{i+1} - \tau_i$), respectively. $\alpha$ refers to the ROC characteristics weight in the evaluation of

$\mu_1(\tau_i)$ and $\mu_2(\tau_i)$; $D_a(\tau_i)$ and $D_m(\tau_i)$ refer to the segments of data forwarded to the anomaly (ADSs) and the misuse (MDSs) at $\tau_i$:

$$D_a(\tau_{i+1}) = D_a(\tau_i) \pm \triangle D \tag{9}$$

$$D_m(\tau_{i+1}) = D_m(\tau_i) \pm \triangle D. \tag{10}$$

**ALGORITHM 3:** Adaptive IDS: (ASCH-IDS)

**procedure** ROC Tracking ▷ Tracking the deviations in (ROC) of the anomaly and the misuse detection subsystems
**Input:** $TP_1(\tau_0)$ , $FP_1(\tau_0)$ ,$TP_2(\tau_0)$ , $FP_2(\tau_0)$
**Output:** $\mu_1(\tau_i)$, $\mu_2(\tau_i)$
▷ $\mu_1(\tau_i)$: Ratio of $TP$ to $FP$ at $\tau_i$ for the anomaly detection subsystem
▷ $\mu_2(\tau_i)$ : Ratio of $TP$ to $FP$ at $\tau_i$ for the misuse detection subsystem

**Initiate:** $\mu_1(\tau_0) \leftarrow$INIT , $\mu_2(\tau_0) \leftarrow$INIT , $t_i \leftarrow \tau_0$ , $I(\tau_0) \leftarrow 1$ ▷ $\mu_1(\tau_0)=TP_1(\tau_0)/FP_1(\tau_0)$
▷ $\mu_2(\tau_0)=TP_2(\tau_0)/FP_2(\tau_0)$
**for** *Allnetwork* **do**
    **if** *HALT* **then**
    END
    **else**
        Check for *NextAggregation*
**if** *NextAggregation* **then**
    $\tau_i \leftarrow \tau_i + \Delta\tau$
    $\mu_1(\tau_i) \leftarrow \alpha\mu_1(\tau_i - \Delta\tau) + (1-\alpha)\frac{TP_1(\tau_i) - TP_1(\tau_i - \Delta\tau)}{FP_1(\tau_i) - FP_1(\tau_i - \Delta\tau)}$
    $\mu_2(\tau_i) \leftarrow \alpha\mu_2(\tau_i - \Delta\tau) + (1-\alpha)\frac{TP_2(\tau_i) - TP_2(\tau_i - \Delta\tau)}{FP_2(\tau_i) - FP_2(\tau_i - \Delta\tau)}$
**else**
    Go to *HALT*
**if** $I_(\tau_i) > I_(\tau_i - \Delta\tau)$ **then**
    $D_a(\tau_i) \leftarrow D_a(\tau_i - \Delta\tau) + \Delta D$
    $D_m(\tau_i) \leftarrow D_m(\tau_i - \Delta\tau) - \Delta D$
    Go to Initiate
**if** $I_(\tau_i) < I_(\tau_i - \Delta\tau)$ **then**
    $D_a(\tau_i) \leftarrow D_a(\tau_i - \Delta\tau) - \Delta D$
    $D_m(\tau_i) \leftarrow D_m(\tau_i - \Delta\tau) + \Delta D$
    Go to Initiate **else**
        Go to Initiate
**END**
**end procedure**

As the proposed system consists of two subsystems, its complexity is a function of the two algorithms complexities. Since the random Forest is a special model of decision trees, its complexity can be extracted from decision tree, such as the complexity $C(Randomforest)$ for building a decision tree with $r$ records and $v$ variables is $O(v + rec * \log(v))$, while for multi trees as in our random forest, the complexity is $O(Tr * Var * rec * log(Var))$ where $T_r$ is the number of trees, $Var$ is the number of variables. For E-DBSCAN as the second subsystem, the complexity is directed by the query requests' number. Since each point is operates with only one query, the run-time complexity is $O(n)<(n * \log(n))$. In our E-DBSCAN, the initialisation step is executed 1 time, the comparison step is executed $(m + 1)$ times, and the incremental step is executed $(m)$ times. Thus, the complexity will be as $O(2 + (2m)) = O(m)$. To this end, the overall complexity of the system can be calculated by $O((T_r * Var * rec * \log(Var)) + (2 + (2m))$.

### 5.2 Deep Learning Based Restricted Boltzmann Machine-Based Clustered IDS (RBC-IDS)

Restricted Boltzmann machine (RBM) is a neural, energetic network containing two types of layers: (V) and (H), which refer to the visible and the hidden layers, respectively, where the learning procedure is steered by an unsupervised fashion [19]. The RBM permits connections between neurons of the same layer, which makes it restricted, and the procedure is presented in the pseudocode in Algorithm 4.

**ALGORITHM 4:** Restricted Boltzmann Machine (RBM) Procedure

**procedure** RBC-IDS
**Input:** $W_{xy}$ , $a_x$, $b_y$
**Output:** $P(Intrusive), P(Normal)$
**Initiate:** $W_{xy}$ , $a_x$, $b_y$, $X$ , $Y$
**for** $y = 1, 2, 3, \ldots, Y$ *(All hidden layers)* **do**
  **for** $x = 1, 2, 3, \ldots, X$ *(All visible layers)* **do**
    **Compute** $E(V, H|\Theta) = -\sum_{x=1}^{X} a_x V_x - \sum_{y=1}^{Y} b_y H_y - \sum_{x=1}^{X}\sum_{y=1}^{Y} V_x H_y W_{xy}$
    **Compute** $P(V, H) = \frac{e^{-E(V,H)}}{\sum_{X,Y} e^{-E(V,H)}}$
    **Compute** $P(V) = \sum_Y P(V, H) = \frac{\sum_Y e^{-E(V,H)}}{\sum_X \sum_Y e^{-E(V,H)}}$
    **Compute** $P(H) = \sum_X P(V, H) = \frac{\sum_X e^{-E(V,H)}}{\sum_X \sum_Y e^{-E(V,H)}}$
    **Compute** $P(V|H) = \prod_{x=1}^{V} P(V_x|H)$
    **Compute** $P(H|V) = \prod_{y=1}^{H} P(H_y|V)$
    **Compute** $P(Intrusive) = P(O = 0|V)$
    **Compute** $P(Normal) = P(O = 1|V)$
  End for
End for
**end procedure**

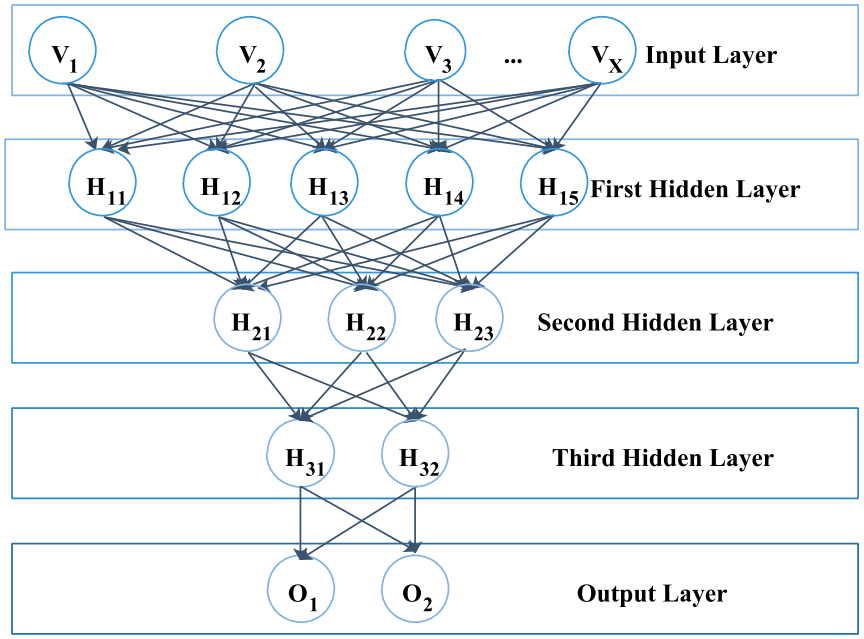


Fig. 3. RBM used in RBC-IDS which contains one visible layer and three hidden layers and one output layer. $W_{11}$ refers to the weight between the visible layer and the first hidden layer, $W_{12}$ refers to the weight between the first and second hidden layers and $W_{23}$ refers to the weight between the second and third hidden layers. $O_1$ and $O_2$ are the *Intrusive* and *Normal* outputs.

Table 1 contains the RBM parameters used in Algorithm 4.

The network sets probability score to each case in hidden and visible elements [19, 33].

Figure 3 represents the used RBM setting in RBC-IDS. The RBC-IDS consists of an input layer contains $X$ visible nodes, three hidden layers and output layer with two outputs $O_1$ and $O_2$ for *Intrusive* and *Normal* outputs respectively. $W_{11}$ represents the weight between the first visible layer and the first hidden layer while $W_{12}$ refers to the weight of the first and the second hidden layers and $W_{23}$ is the weight between the second and the third hidden layers.

In the RBC-IDS, each CH is responsible for aggregating the sensed data from the sensors in the same cluster and forwards them to the server by adopting the procedure in [32].

### 5.3 Reinforcement Learning

In reinforcement learning, the studied environment is defined as Markov Decision Process (MDP) which comes up with an optimum policy to achieve maximum rewards over time [39]. The fundamentals of reinforcement learning concept are as follows:

- The agent interacts with the environment and takes actions $A_t$ in each state $S_t$ and observes the feedback from the environment.
- The environment supplies a reward $R_t$ for the actions performed in form of $R^+$ or $R^-$ which refer to positive or negative rewards respectively.
- The agent observes the environment for any changes and optimise the received rewards by updating the policies.
- Different reinforcement learning techniques are called in order to maximise the expected value of the total reward, starting from the current state.

*5.3.1 Q-Learning.* Q Learning builds on the concept of value iteration where the agent aims to estimate the value function to update all states $s$ and actions $A$ every iteration in order to know which action $A$ leads to higher reward $R$. In the Q table, the rows represent the states while the columns represent the actions. In a state (say state $S$), the agent takes an action (i.e., action $A$), observes the reward ($R$) for this action, as well as the next state ($S'$), and re-estimates the Q value.

Equation (11) represents the estimated $Q$ value [40] where $S_t$, $A_t$ and $R_t$ stand for the state, action and reward at time $t$, respectively. In addition, $\alpha$ and $\gamma$ represent the learning rate and a constant regarding the relative value of rewards, respectively. Indeed, the following conditions hold for both parameters: $0 < \alpha < 1$ and $0 < \gamma < 1$.

$$\begin{aligned} Q^+(S_t, A_t) \leftarrow (1-\alpha)Q^+(S_t, A_t) + \\ \alpha(R_t + \gamma max Q^+(S_t^+, A_t^+)) \end{aligned} \tag{11}$$

Equation (12) formulates the estimated function $V^\pi(S)$ which represents the estimation of the future reward $R$ that will be received in the initial state $S$ [41]. In the equation, $\pi(S, A)$ denotes the likelihood of action $A$ in state $S$, $P_{SS^+}(A)$ stands for the state transition probability between states $S$ and $S^+$ with action $A$. $R(S, S^+, A)$ is the reward issued after transitioning from state $S$ to $S^+$ at action $A$, $r$ is the discount factor weight from future rewards to current rewards [41]. The value iteration used method is shown in Equation (13) below, where $V^\pi(S)$ refers to the value estimation of $R$ that at initial state $S$, $\pi(S, A)$ is the probability of $A$ in $S$, $P_{SS^+}(A)$ refers to the transitional probability from state $S$ to $S^+$ at $A$, $R(S, S^+, A)$ is the reward returned from transition from state $S$ to $S^+$ at $A$, $r$ is the discount factor weight from future rewards to current rewards, $V_I^\pi(S^+)$ is the value estimation of $R$ at state $S^+$ at the initial iteration $I$, and $V_{I+1}^\pi(S)$ is the value estimation of $R$ at state $S$ at the updated iteration $I + 1$.

$$V^\pi(S) = \sum_A \pi(S, A) \sum_{S^+} P_{SS^+}(A) R(S, S^+, A) + rV^\pi(S^+) \tag{12}$$

$$V_{I+1}^\pi(S) = max_A \sum_{S^+} P_{SS^+}(A) R(S, S^+, A) + rV_I^\pi(S^+). \tag{13}$$

It is worth noting that the reason behind adopting Q-learning as one of our reinforcement learning methods is the model-free nature of Q-learning. Furthermore, by applying Q-learning, it is also possible to address stochastic rewards in a non-adaptive manner. In addition, Q-learning has the ability to learn without necessarily following the current policy [42].

*5.3.2 State-Action-Reward-State-Action Learning (SARSA).* SARSA is an MDP-based reinforcement learning algorithm which is considered as a Modified Connectionist Q-Learning (MCQ-L) algorithm. SARSA updates the Q-values depending on current state $S$, the current action $A$ for $S$, the returned reward $R$ of action $A$, the new state $S$, and the next action $A$ for the new state, which can be represented in the quintuple $(S_t, A_t, R_t, S_{t+1}, A_{t+1})$.

SARSA in an on-policy learning algorithm where the agent interrelates with environment and updates the policy considering the taken actions. In SARSA, the Q-value function represents the received reward in the next time step for taking action $A$ in state $S$, and the reward received from the next state and action [43]. The previous Q-value function (Equation (11)) can be updated as in Equation (14).

The equations Equation (11) and Equation (14) look almost the same except that in Q-learning the action with highest estimation of all possible next actions will be considered while the actual next action is considered in SARSA. Looking for the maximum reward in Q-learning can make it

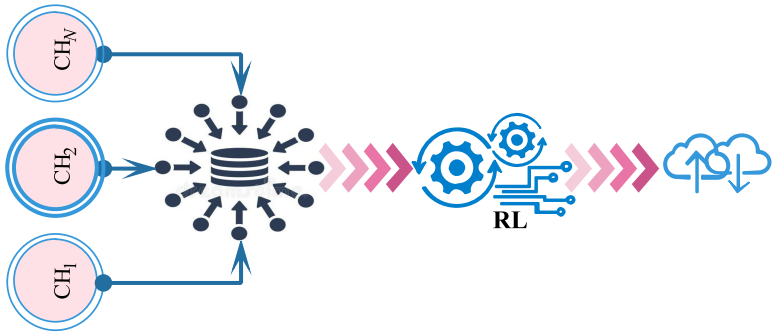


Fig. 4. Reinforcement- learning-based IDS representation.

more costly compared to SARSA technique [14]:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_t + \gamma Q^+(S_{t+1}, A_{t+1}) - Q(S_t, A_t)). \tag{14}$$

*5.3.3 Temporal Difference Learning (TD).* This is a model-free reinforcement learning technique which can learn by estimating the expected value function by considering approximating distribution from the current value. The TD technique estimates the state value function under a policy $\pi$ as shown in Equation (15) and Equation (16) beow below.

$$V^{\pi}(S_t) = E_{\pi}\left\{\sum_{t=0}^{\infty} \gamma^t R_t\right\} \tag{15}$$

$$V^{\pi}(S_t) = E_{\pi}\{R_0 + \gamma V^{\pi}(S_1)\}, \tag{16}$$

where $V^{\pi}(S_t)$ refers to the state value function with state $S_t$, $R$ refers to the reward, and $\gamma$ is the discount rate under the policy $\pi$. In Equation (16), $R_0 + \gamma V^{\pi}(S_1)$ represents the unbiased estimate for $V^{\pi}(S_t)$. TD is a technique used to learn how to estimate a value that depends on future values, which make it useful to learn both the V- function and the Q- function, whereas, Q-learning is a specific technique to learn only the Q-function.

**Reinforcement learning-based Intrusion Detection System (Q-IDS)** for WSNs is illustrated in Figure 4. The IDS consists of hierarchically connected clusters with the aggregator and one central agent that is the Agent in the representation of Reinforcement learning Box which tries to model the state of the monitored network. Following a series of iterations, the central agent knows the action $A$ that needs to be executed in response to each state $S$ in order to obtain a positive reward $R^+$.

It is worth mentioning that the value iteration works by generating consecutive estimates of the optimal value function in which each of these iterations can be accomplished in $O(|A||S|^2)$. In reinforcement learning, the required number of iterations can grow exponentially.

## 6 EXPERIMENTAL RESULTS AND ANALYSIS

### 6.1 Description of the KDD Dataset

We simulate 20 devices in a network that communicate through H-DSR protocol which is a Dynamic Source Routing protocol specialised for the Hierarchical networks [44]. The tested devices are deployed in four clusters in an area of 100m x 100m. The resulting figures present the mean of 10 tests (runs) for each scenario with confidence level of 95%. Table 2 presents the simulation inputs.

Table 2. Testing Settings

| Simulation Inputs | Input Value |
|---|---|
| Visible elements (inputs) | 41 |
| Hidden layers | 3 |
| Number of sensors | 20 |
| Operational area | 100m x 100m |
| Number of clusters | 4 |
| Simulation time | 600s |
| Attack Types | DoS,Probe,U2R,R2L |
| Communication range | 100m |
| $\alpha$ (The weight of $TP/FP$) | 0.7 |
| $INIT$ | 0.5 |
| Routing protocol | H-DSR |
| Packet size | 250 bytes |
| Trust range | [0,1] |

Table 3. KDD Dataset Description [46]

| Attack | Attacks in Testset | Attacks in training set |
|---|---|---|
| Dos | processtable, mail-bomb | neptune, teardrop |
| R2L | named, xsnoop | spy, multihop |
| U2R | ps, xterm | bufferoverflow, perl |
| Probe | saint, mscan | portsweep, ipsweep |

The MIT Lincoln laboratory has collected the datasets for computer network IDS evaluation under the support of Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL). The Knowledge Discovery in Data mining CUP 1999 (KDDCup99) dataset is a subset of the DARPA dataset [45]. The KDDCup99 is used to test the IDSs efficiency on the simulated WSN, with each connection record containing 41 features and which are classified as normal or attack behaviors. The KDDCup99 contains around 311,029 records as test dataset and 494,020 records as training dataset. The numerous attack types in these datasets are gathered into attack groups that assign similar attack types within a single group, resulting in an advancement of the detection rate [45]. KDDCup99 attacks fall under four major groups, specifically, DoS, Probe, R2L and, U2R, which refer to Denial of Service, probe, Remote to Local and, User to Root attacks, respectively. Table 3 presents examples of the attacks in the KDDCup99 test dataset and KDDCup99 training dataset.

### 6.2 Performance Evaluation

IDSs are evaluated based on the following criteria:

(1) *True Positive (TP)* are the anomalous cases that were correctly classified as abnormal.
(2) *False Positive (FP)* are the normal cases that were incorrectly classified as anomalous.
(3) *True Negative (TN)* are the normal cases that were classified correctly.
(4) *False Negative (FN)* are the anomalous cases that were incorrectly classified as normal.

*6.2.1 Preprocessing Phase.* Since the models' performance evaluations are tested using the KDD'99 dataset and some of the features are represented by string values (i.e., protocols' names), a

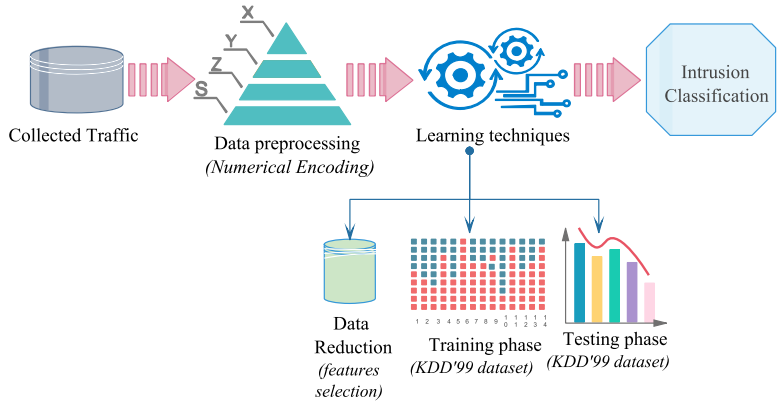


Fig. 5. The adopted learning phases.

*Numerical encoding* process has been adopted for this matter. The TCP, ICMP, and UDP protocols' names were encoded as 001, 010, and 011, respectively.

*6.2.2 Training and Testing Phase.* The KDD'99 dataset that is used consists of training and testing datasets. Each data line of the dataset has 41 features, consisting of 38 numerical features and 3 non-numerical features. In NS3, we start with training the different models (e.g., RBC-IDS training started with training the first layer, gathering the data generated from the trained layer, and using these data as the training dataset for the second layer, and so on).

Figure 5 represents the adopted common phases among all the proposed learning mechanisms (e.g., machine learning, deep learning, and reinforcement learning).

*6.2.3 Evaluated Metrics.* The following metrics have been considered for evaluating the various IDSs.

*Accuracy Rate (AR).* AR denotes the ratio of the truly classified incidences that return to True Positive (TP) and True Negative (TN) incidences. AR is represented in Equation (17) [47].

$$AR = \frac{TP + TN}{TP + TN + FP + FN} \tag{17}$$

AR has been presented to trace ASCH-IDS, RBC-IDS, TD-IDS, SARSA-IDS, and Q-IDS ARs for different scenarios. Figure 6 shows the AR for the ASCH-IDS, RBC-IDS, TD-IDS, SARSA-IDS, and Q-IDS. The proposed Q-IDS achieves the highest AR of 100% followed by SARSA-IDS with an *AR* of 99.97% and TD-IDS with an AR of 99.94%.

Q-IDS performs with the highest AR for two reasons: First, Q-Learning is based on the exemplars from the training datasets and suited for decision making while the system is running. The agent interacts with the environment and aims to optimise the cumulative reward ($R^+$) by learning the best actions through feedback. On the other hand, RBC-IDS makes predictions about the data and learns from the training datasets to build a classifying model.

*Detection Rate (DR).* DR denotes the behaviour that is accurately recognised as intrusive. It signifies the (TP) as displayed in Equation (18) [47]. DRs for ASCH-IDS, RBC-IDS, TD-IDS, SARSA-IDS, and Q-IDS are shown in Figure 7.

$$DR = \frac{TP}{TP + FP} \tag{18}$$

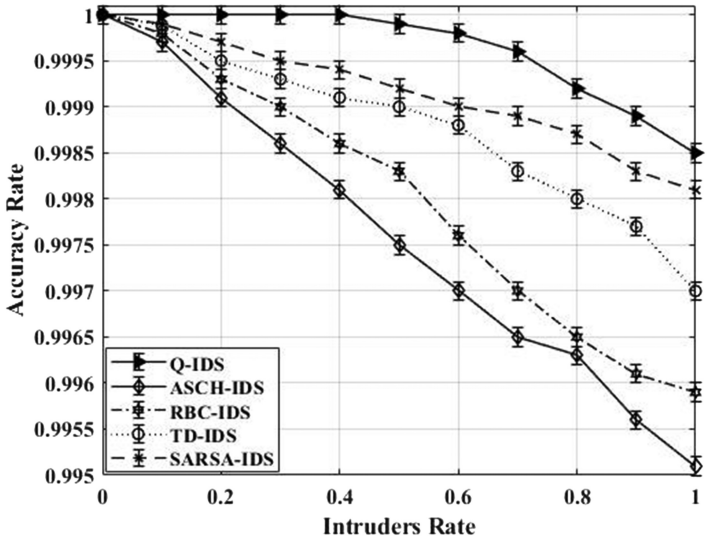


Fig. 6. Accuracy rate comparison among Q-IDS, ASCH-IDS, RBC-IDS, TD-IDS, and SARSA-IDS.

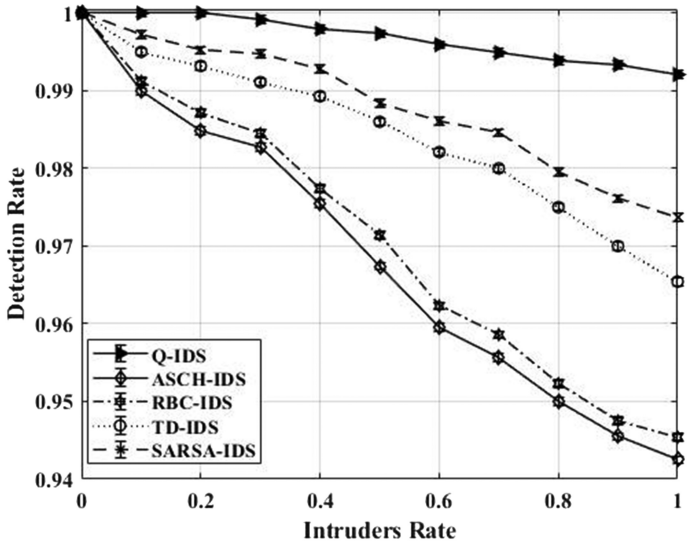


Fig. 7. Detection rates comparison among Q-IDS, ASCH-IDS, RBC-IDS, TD-IDS, and SARSA-IDS.

Figure 7 illustrates the DRs for ASCH-IDS, RBC-IDS, TD-IDS, SARSA-IDS, and Q-IDS. As shown in the figure, the proposed Q-IDS achieves the highest *DR* followed by SARSA-IDS and TD-IDS.

*False Negative Rate (FNR).* FNR is defined as the ratio of malicious behaviours that have been designated as non-intrusive [6], as shown in Equation (19) [47].

$$FNR = \frac{FN}{TP + FN + FP + TN} \tag{19}$$

FNR of ASCH-IDS, RBC-IDS, SARSA-IDS, TD-IDS compared to Q-IDS is shown in Figure 8.

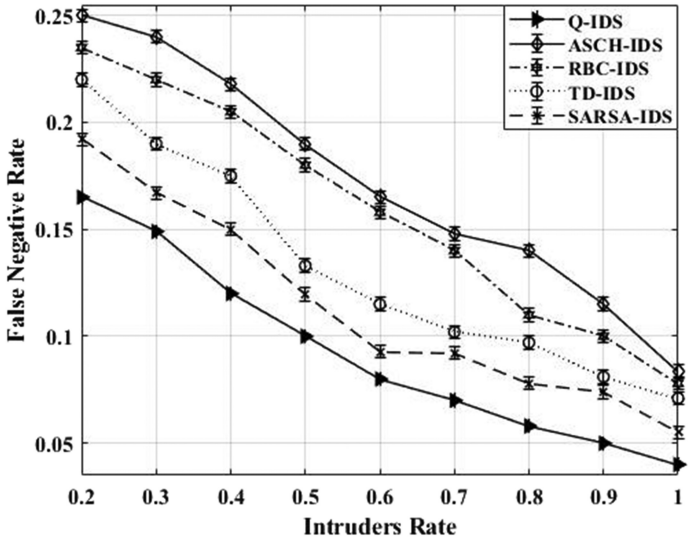


Fig. 8. FNR comparison between Q-IDS, ASCH-IDS, RBC-IDS, TD-IDS, and SARSA-IDS.

In reinforcement-learning-based Q-IDS, the overall FNR has been mitigated when compared to the case under the other reinforcement learning techniques (SARSA-IDS and TD-IDS), deep-learning-based RBC-IDS and the ASCH-IDS reacting to the increase of TP which is the reason behind the DR and AR enhancement.

This can be interpreted as follows: While QL uses function approximation that represents the value function to react to all actions and target the positive rewards, RBM is competent at feature reduction which helps in the elimination of redundant features and reduction of the FN cases.

*ROC Curve*. The ROC curve refers to the ratio between Sensitivity and the $(1 - Specificity)$, which are TP and FP, respectively.

The *Sensitivity-Specificity Ratio* is represented by the area under the curve where the larger area reflects the best performance. We plot the ROC curves for ASCH-IDS, Q-IDS, RBC-IDS SARSA-IDS, and TD-IDS in order to assess the system performance as shown in Figure 9. It is clear that the Q-IDS performs better with the largest area under the curve followed by SARSA-IDS and TD-IDS. Since Q-IDS achieves the highest TP, the Q-IDS achieves the best performance in terms of ROC.

$F_1$ *Score*. $F_1$ score studies the *precision-recall* of the test in order to calculate its $F_1$ score. The *precision* is the number of true positive incidences divided by all positive incidences; it is expressed as $TP/(TP + FP)$. The *recall* is formulated as $TP/(TP + FN)$, representing the number of true positive incidences divided by all *actually* positive instances as shown in Figure 10. High system performance can be achieved by the high recall and high precision. Thus, the near precision-recall to 1 results in the best performance [48]. Q-IDS achieves the highest precision-recall ratio compared to ASCH-IDS and RBC-IDS as shown in Figure 10. Precision and recall primarily depend on TP performance. Q-IDS performs with the highest *precision-recall* since Q-learning is based on the exemplars from the training datasets and suited for on-the-fly decision making while the agent interacts with the environment tested environment and uses the feedback to choose the actions to optimise the cumulative reward ($R^+$).

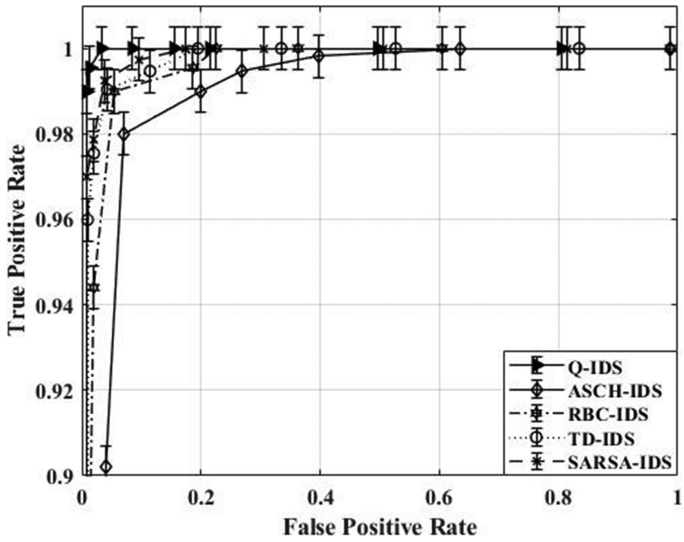


Fig. 9. ROC curves representation comparison between Q-IDS, ASCH-IDS, RBC-IDS, SARSA-IDS, and TD-IDS.

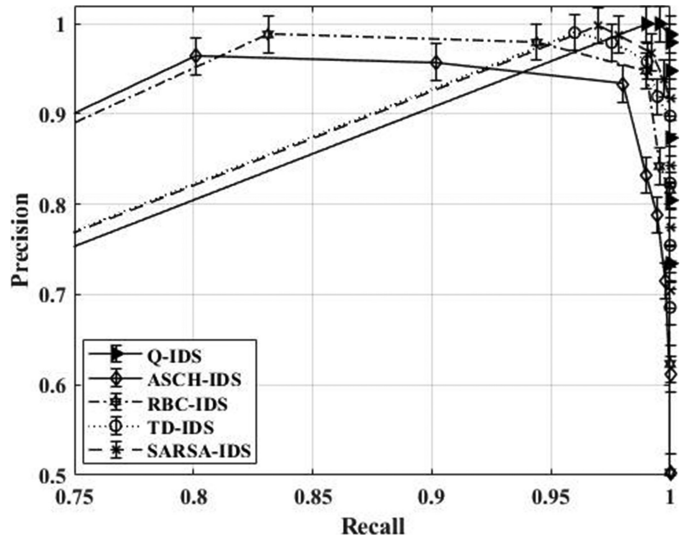


Fig. 10. Precision-Recall representation of Q-IDS, ASCH-IDS, RBC-IDS, SARSA-IDS and TD-IDS.

$F_1$ denotes the harmonic average of precision and recall; $F_1$-score is formulated as in Equation (20); and $F_1$-score performance of the IDS solutions for the WSNs under investigation are plotted in Figure 11 [49].

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{20}$$

The $F_1$ score measures test accuracy [49] by studying the precision-recall of the test in order to calculate its $F_1$ score. $F_1$ score is a reliable metric to use if the test cases seek balance between

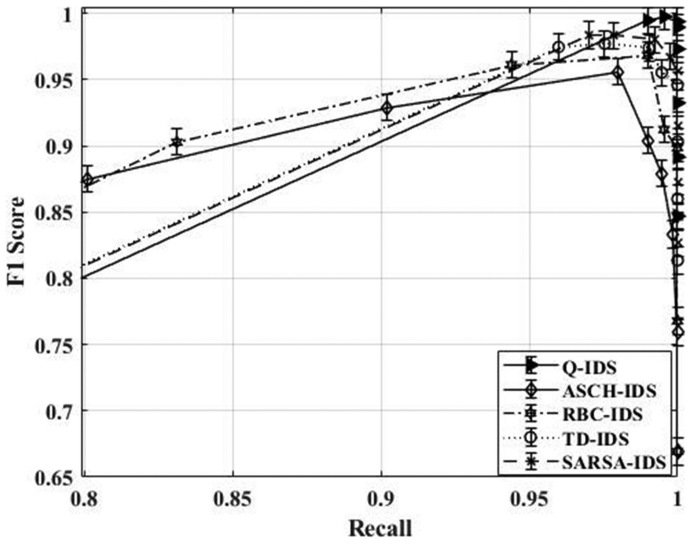


Fig. 11. $F_1$ Score representation of Q-IDS, ASCH-IDS, RBC-IDS, SARSA-IDS and TD-IDS.

precision and recall. Thus, the better the precision-recall, the better the $F_1$ score. Figure 11 shows that Q-IDS introduces an improved $F_1$ score compared to SARSA-IDS, TD-IDS, ASCH-IDS, and RBC-IDS.

## 7 CONCLUSION

Volunteer computing uses Internet-connected devices, which makes it easy for participants to share their resources, especially for the critical infrastructures. Critical infrastructure security is considered as one of the vital issues in smart cities. Monitoring the network's components and detecting malicious behaviors are fundamental functions to ensure security of the monitoring operation. Since Wireless Networks are deployed in open and uncontrolled environments, transmitting data through them leads to huge vulnerabilities. Therefore, the robustness of Intrusion Detection Systems (IDSs) in Wireless networks is a must. In this study, we present a comparative study between machine learning-based IDS systems with deep-learning-based IDS for critical infrastructure monitoring WSNs. We specifically investigate our previously proposed machine learning-based Adaptively Supervised and Clustered Hybrid IDS (ASCH-IDS) with a Restricted Boltzmann Machine-based and Clustered IDS (RBC-IDS), Q-Learning-based IDS (Q-IDS), SARSA-IDS and TD-IDS, which are reinforcement learning solutions. Through simulations, we have verified that QL-IDS works with ≈100% detection rate and ≈100% accuracy rate in the existence of intrusive behavior in the tested WSN. From the performance analysis, we have shown that the adaptive machine learning-based solution performs in same rates as the deep learning-based solution whereas adopting a machine learning-based IDS framework leads to approximately half the detection time of the deep learning-based RBM-IDS framework. We have also shown that the reinforcement learning-based solutions perform with the best precision-recall, $F_1$ score with ≈1 which represents the best performance and with the largest area under curve (ROC).

As a future work, we are planning to extend the presented IDS to larger networks that consist of a large number of nodes and clusters. Moreover, we are testing the impact of heterogeneous cluster sizes on our presented solution's overall performance.

## REFERENCES


[1] I. Al-Ridhawi, S. Otoum, M. Aloqaily, Y. Jararweh, and Th. Baker. 2020. Providing secure and reliable communication for next generation networks in smart cities. *Sustainable Cities and Society* 56 (2020), 102080. http://dx.doi.org/10.1016/j.scs.2020.102080

[2] L. Buttyan, D. Gessner, A. Hessler, and P. Langendoerfer. 2010. Application of wireless sensor networks in critical infrastructure protection: Challenges and design options [Security and Privacy in Emerging Wireless Networks]. *IEEE Wireless Communications* 17, 5 (October 2010), 44–49. http://dx.doi.org/10.1109/MWC.2010.5601957

[3] Ismaeel Al Ridhawi, Yehia Kotb, Moayad Aloqaily, Yaser Jararweh, and Thar Baker. 2019. A profitable and energy-efficient cooperative fog solution for IoT services. *IEEE Transactions on Industrial Informatics* 16, 5 (2019), 3578–3586.

[4] Safa Otoum, Burak Kantraci, and Hussein T. Mouftah. 2017. Hierarchical trust-based black-hole detection in WSN-based smart grid monitoring. *IEEE International Conference on Communications (ICC)* (2017). http://dx.doi.org/icc.2017

[5] M. Al-Khafajiy, S. Otoum, Th. Baker, M. Asim, Z. Maamar, M. Aloqaily, M. J. Taylor, and M. Randles. 2020. Intelligent control and security of fog resources in healthcare systems via a cognitive fog model. *ACM Transactions on Internet Technology* (2020).

[6] Safa Otoum, Burak Kantarci, and Hussein T. Mouftah. 2017. Detection of known and unknown intrusive sensor behavior in critical applications. *IEEE Sensors Letters* 1, 5 (Oct 2017), 1–4. http://dx.doi.org/10.1109/LSENS.2017.2752719

[7] Safa Otoum, Burak Kantraci, and Hussein T. Mouftah. 2018. Adaptively supervised and intrusion-aware data aggregation for wireless sensor clusters in critical infrastructures. In *IEEE International Conference on Communications (ICC)*. 1–6.

[8] Safa Otoum, Burak Kantraci, and H. T. Mouftah. 2017. Mitigating false negative intruder decisions in WSN-based smart grid monitoring. In *13th International Wireless Communications and Mobile Computing Conference (IWCMC)*. 153–158. http://dx.doi.org/10.1109/IWCMC.2017.7986278

[9] R. Jain and H. Shah. 2016. An anomaly detection in smart cities modeled as wireless sensor network. In *International Conference on Signal and Information Processing (IConSIP)*. 1–5. http://dx.doi.org/10.1109/ICONSIP.2016.7857445

[10] C. Ioannou, V. Vassiliou, and C. Sergiou. 2017. An intrusion detection system for wireless sensor networks. In *24th International Conference on Telecommunications (ICT)*. 1–5. http://dx.doi.org/10.1109/ICT.2017.7998271

[11] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. 2016. A deep learning approach for network intrusion detection system. In *9th EAI International Conference on Bio-inspired Information and Communications Technologies (Formerly BIONETICS)*. 21–26. http://dx.doi.org/10.4108/eai.3-12-2015.2262516

[12] C. Yin, Y. Zhu, J. Fei, and X. He. 2017. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access* 5 (2017), 21954–21961. http://dx.doi.org/10.1109/ACCESS.2017.2762418

[13] L. Dali, A. Bentajer, E. Abdelmajid, K. Abouelmehdi, H. Elsayed, E. Fatiha, and B. Abderahim. 2015. A survey of intrusion detection system. In *2nd World Symposium on Web Applications and Networking (WSWAN)*. 1–6. http://dx.doi.org/10.1109/WSWAN.2015.7210351

[14] Stefano Zanero and Sergio M. Savaresi. 2004. Unsupervised learning techniques for an intrusion detection system. In *ACM Symposium on Applied Computing (SAC'04)*. ACM, New York, 412–419. http://dx.doi.org/10.1145/967900.967988

[15] Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld. 2009. Active learning for network intrusion detection. In *2nd ACM Workshop on Security and Artificial Intelligence (AISec'09)*. ACM, New York, 47–54. http://dx.doi.org/10.1145/1654988.1655002

[16] J. Straub. 2017. Testing automation for an intrusion detection system. In *IEEE Autotestcon*. 1–6. http://dx.doi.org/10.1109/AUTEST.2017.8080473

[17] Andrew Honig, Andrew Howard, Eleazar Eskin, and Sal Stolfo. 2002. *Adaptive Model Generation: An Architecture for Deployment of Data Mining-Based Intrusion Detection Systems*. Kluwer Academic Publishers, 153–194.

[18] Mostafa A. Salama, Heba F. Eid, Rabie A. Ramadan, Ashraf Darwish, and Aboul Ella Hassanien. 2011. Hybrid intelligent intrusion detection scheme. In *Soft Computing in Industrial Applications*, António Gaspar-Cunha, Ricardo Takahashi, Gerald Schaefer, and Lino Costa (Eds.). Springer Berlin, Berlin, 293–303.

[19] Arnaldo Gouveia and Miguel Correia. 2017. *A Systematic Approach for the Application of Restricted Boltzmann Machines in Network Intrusion Detection*. Vol. 10305. 432–446.

[20] Yazan Otoum, Dandan Liu, and Amiya Nayak. DL-IDS: A deep learning–based intrusion detection framework for securing IoT. *Transactions on Emerging Telecommunications Technologies* n/a, n/a ([n.d.]), e3803. http://dx.doi.org/10.1002/ett.3803 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/ett.3803. e3803 ett.3803.

[21] M. Z. Alom, V. Bontupalli, and T. M. Taha. 2015. Intrusion detection using deep belief networks. In *National Aerospace and Electronics Conference (NAECON)*. 339–344. http://dx.doi.org/10.1109/NAECON.2015.7443094

[22] Ugo Fiore, Francesco Palmieri, Aniello Castiglione, and Alfredo De Santis. 2013. Network anomaly detection with the restricted Boltzmann machine. *Neurocomputing* 122 (2013), 13–23. http://dx.doi.org/10.1016/j.neucom.2012.11.050

[23] Yuancheng Li, Rong Ma, and Runhai Jiao. 2015. A hybrid malicious code detection method based on deep learning. *International Journal of Security and Its Applications* 9 (05 2015), 205–216.
[24] A. Abeshu and N. Chilamkurti. 2018. Deep learning: The frontier for distributed attack detection in fog-to-things computing. *IEEE Communications Magazine* 56, 2 (Feb 2018), 169–175. http://dx.doi.org/10.1109/MCOM.2018.1700332
[25] Rafal Kozik, Michal Choras, Massimo Ficco, and Francesco Palmieri. 2018. A scalable distributed machine learning approach for attack detection in edge computing environments. *J. Parallel and Distrib. Comput.* 119 (2018), 18–26. http://dx.doi.org/10.1016/j.jpdc.2018.03.006
[26] Arturo Servin and Daniel Kudenko. 2008. Multi-agent reinforcement learning for intrusion detection. In *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*. Springer Berlin, Berlin, 211–223.
[27] Xin Xu and Tao Xie. 2005. A reinforcement learning approach for host-based intrusion detection using sequences of system calls. In *Advances in Intelligent Computing*. Springer Berlin, Berlin, 995–1003.
[28] Indah Tiyas, Ali Barakbah, Tri Harsono, and Amang Sudarsono. 2014. Reinforced intrusion detection using pursuit reinforcement competitive learning. *EMITTER International Journal of Engineering Technology* 2, 1 (2014), 39–49. http://dx.doi.org/10.24003/emitter.v2i1.16
[29] James Cannady Georgia. 2000. Next generation intrusion detection: Autonomous reinforcement learning of network attacks. In *23rd National Information Systems Secuity Conference*. 1–12.
[30] Arturo Servin. 2007. Towards traffic anomaly detection via reinforcement learning and data flow. Department of Computer Science, University of York, United Kingdom.
[31] Fatma Belabed and Ridha Bouallegue. 2016. An optimized weight-based clustering algorithm in wireless sensor networks. In *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)* (2016). http://dx.doi.org/10.1109/iwcmc.2016.7577152
[32] Wei Zhang, Sajal Das, and Yonghe Liu. 2006. A trust based framework for secure data aggregation in wireless sensor networks. In *IEEE Communications Society on Sensor and Ad Hoc Communications and Networks* (2006). http://dx.doi.org/10.1109/sahcn.2006.288409
[33] S. Seo, S. Park, and J. Kim. 2016. Improvement of network intrusion detection accuracy by using restricted Boltzmann machine. In *8th International Conference on Computational Intelligence and Communication Networks (CICN)*. 413–417. http://dx.doi.org/10.1109/CICN.2016.87
[34] Daoying Ma and Aidong Zhang. 2004. An adaptive density-based clustering algorithm for spatial database with noise. In *IEEE International Conference on Data Mining (ICDM'04)* (2004). http://dx.doi.org/10.1109/icdm.2004.10036
[35] A. Ram, A. Sharma, A. S. Jalal, A. Agrawal, and R. Singh. 2009. An enhanced density-based spatial clustering of applications with noise. In *IEEE International Advance Computing Conference*. 1475–1478. http://dx.doi.org/10.1109/IADCC.2009.4809235
[36] Leo Breiman and Adele Cutler. [n.d.]. Random Forests. http://www.stat.berkeley.edu/~breiman/RandomForests/.
[37] Jiong Zhang, M. Zulkernine, and A. Haque. 2008. Random-forests-based network intrusion detection systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C* 38/5 (2008), 649–659. http://dx.doi.org/10.1109/tsmcc.2008.923876
[38] M. F. Jiang, S. S. Tseng, and C. M. Su. 2001. Two-phase clustering process for outliers detection. *Pattern Recognition Letters* 22/6-7 (2001), 691–700. http://dx.doi.org/10.1016/s0167-8655(00)00131-8
[39] S. Doltsinis, P. Ferreira, and N. Lohse. 2014. An MDP model-based reinforcement learning approach for production station ramp-up optimization: Q-learning analysis. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 44, 9 (Sept 2014), 1125–1138. http://dx.doi.org/10.1109/TSMC.2013.2294155
[40] Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8, 3 (01 May 1992), 279–292. http://dx.doi.org/10.1007/BF00992698
[41] Xin Du and Jinjian Zhai. 2016. Algorithm trading using Q-learning and recurrent reinforcement learning. *Positions Journal*.
[42] Chris Gaskett, David Wettergreen, and Alexander Zelinsky. 1999. Q-learning in continuous state and action spaces. In *Advanced Topics in Artificial Intelligence*, Norman Foo (Ed.). Springer Berlin, Berlin, 417–428.
[43] D. Kumar, N. Logganathan, and V. P. Kafle. 2018. Double SARSA based machine learning to improve quality of video streaming over HTTP through wireless networks. In *2018 ITU Kaleidoscope: Machine Learning for a 5G Future (ITU K)*. 1–8. http://dx.doi.org/10.23919/ITU-WT.2018.8597682
[44] M. Tarique, K. E. Tepe, and M. Naserian. 2005. Hierarchical dynamic source routing: Passive forwarding node selection for wireless ad hoc networks. *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications* 3 (Aug 2005), 73–78 Vol. 3. http://dx.doi.org/10.1109/WIMOB.2005.1512887
[45] The UCI KDD Archive, University of California, Irvine. *KDD Cup 1999 Data*. Available at http://www.kdd.ics.uci.edu/databases/kddcup99/kddcup99/html/, Last Visit: April.10.2018.
[46] P. Natesan and P. Balasubramanie. 2012. Multi stage filter using enhanced adaboost for network intrusion detection. *International Journal of Network Security & Its Applications* 4 (05 2012), 121–135.

[47] B. M. Beigh and M. A. Peer. 2014. Performance evaluation of different intrusion detection system: An empirical approach. In *International Conference on Computer Communication and Informatics*. 1–7. http://dx.doi.org/10.1109/ICCCI.2014.6921740

[48] Hesham Elmahdy, M. Elhamahmy, and Imane A. Saroit. 2010. A new approach for evaluating intrusion detection system. In *Artificial Intelligent Systems and Machine Learning*, Vol. 2. 290–298.

[49] M. Elhamahmy, N. Hesham, and A. Imane. 2010. A new approach for evaluating intrusion detection system. In *Artificial Intelligent Systems and Machine Learning*, Vol. 2. 290–298.

Received March 2020; revised May 2020; accepted June 2020