EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

An adversarial environment reinforcement learning-driven intrusion detection algorithm for Internet of Things

Chahira Mahjoub[1], Monia Hamdi[2], Reem Ibrahim Alkanhel[2], Safa Mohamed[1,2] and Ridha Ejbali[1*]

*Correspondence:
ridha_ejbali@ieee.org

[1] Research Team in Intelligent Machines, National School of Engineers of Gabes, University of Gabes, B.P. W, 6072 Gabes, Tunisia
[2] Department of Information Technology, College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University, P.O. Box 84428, 11671 Riyadh, Saudi Arabia

Abstract

The increasing prevalence of Internet of Things (IoT) systems has made them attractive targets for malicious actors. To address the evolving threats and the growing complexity of detection, there is a critical need to search for and develop new algorithms that are fast and robust in detecting and classifying dangerous network traffic. In this context, deep reinforcement learning (DRL) is gaining recognition as a prospective solution in numerous fields as it enables autonomous agents to cooperate with their environment for decision-making without relying on human experts. This article presents an innovative approach to intrusion detection in IoT systems using an adversarial reinforcement learning (RL) algorithm known for its exceptional predictive capabilities. The predictive process relies on a classifier, implemented as a streamlined and highly efficient neural network. Embedded within this classifier is a policy function meticulously trained using an innovative RL model. Importantly, this model ensures that the environment's behavior is dynamically fine-tuned simultaneously with the learning process, improving the overall effectiveness of the intrusion detection approach. The efficiency of our proposal was assessed using the Bot-IoT database, consisting of a mixture of legitimate IoT network traffic and simulated attack scenarios. Our scheme shows superior performance compared to existing ones. Therefore, our approach to IoT intrusion detection can be considered a valuable alternative to existing methods, capable of significantly improving the IoT systems' security.

Keywords: Intrusion detection, Internet of Things, Reinforcement learning, Adversarial learning

1 Introduction

The rapid development of the Internet has played a prominent role in the emergence of the Internet of Things (IoT), a paradigm in which everyday objects are connected through information-sensing devices to facilitate information exchange. The IoT aims to connect diverse objects to the Internet, enabling them to interact with each other. This connectivity has given rise to various applications, such as smart homes, smart cities, and healthcare systems, where intelligent identification and management are realized.

IoT refers to a network of interrelated everyday devices fitted with processors and network cards that allow them to be controlled through web services or other

interfaces [1]. As with any widely adopted technology, IoT has captured the interest of cyber attackers who seek to take advantage of its weaknesses using sophisticated hacking techniques, such as botnets. The absence of standardization in IoT systems and the proliferation of inexpensive, lightweight, and energy-efficient devices have further exacerbated security challenges [2]. One particular way that criminals have exploited IoT networks is through the spread of botnet malware, which is capable of launching devastating Distributed Denial of Service (DDoS) attacks, reaching speeds of up to 1.1 TBps [1, 3].

The need for reliable and efficient intrusion detection has highlighted the importance of using automated systems. One such system is an Intrusion Detection System (IDS) [4], which aims to quickly and precisely examine network traffic to detect possible threats. An IDS has emerged as an important application of machine learning (ML) research on new data networks [5–7]. However, IDS faces significant challenges in terms of prediction due to the nature of the datasets it handles, which are often large, noisy, and unbalanced. An IDS algorithm is trained on a dataset encompassing captured network characteristics and their corresponding labels. This dataset is an essential component for training traditional ML models within a supervised learning framework. Nonetheless, a different approach is used when training a deep reinforcement learning (DRL) algorithm [8], where the training process revolves around the interaction between an agent and its environment. The agent interacts with the environment, where it receives actions, encounters new states, and gains rewards. [8]. The algorithm uses these inputs to improve the policy function of the agent, which plays a crucial role in the DRL framework as it guides the agent in determining the optimal action to take according to the given state and reward while adhering to the Markov property [9]. The DRL framework aims to guarantee that the agent produces actions that effectively maximize the cumulative rewards obtained from the environment during their interaction trajectory.

The application of DRL concepts to an IDS context involves establishing the following analogies: The states in DRL can be compared to network traffic samples in the IDS context. Each network traffic sample represents a particular state in the IDS system. The actions in the DRL can be interpreted as label predictions made by the IDS algorithm. The IDS algorithm selects an action (prediction) according to the given state (traffic sample). The rewards in the DRL can be associated with the value representing the quality of the prediction made by the IDS. In this case, classification accuracy can be considered the reward value. A better prediction (higher accuracy) results in a higher reward.

Given the differences between the supervised learning framework commonly used in IDS and the DRL framework, the reinforcement learning (RL) model has to be adapted to the IDS problem. This adaptation entails using a dataset that contains pre-registered samples comprising network features and corresponding intrusion labels. The IDS learns from this dataset and makes predictions using RL techniques.

Recently, a groundbreaking algorithm named Adversarial Environment using Reinforcement Learning (AE-RL) was introduced [10]. It incorporates a simulated environment that provides network traffic samples and related rewards, with an agent that acts as a classifier and aims to forecast the proper intrusion target based on samples from the network supplied by the environment. The environment generates rewards, which can be positive or negative, based on the agent's correct or incorrect predictions.

The primary objective of the AE-RL algorithm is to increase the total sum of rewards during training. A notable characteristic of the simulated environment is its random extraction of new samples from the pre-registered sample dataset. In addition, the environment itself is learned during the training process and acts adversarially opposite to the agent's policy. In other words, the environment deliberately aims to reduce the rewards provided to the agent by raising the instances of incorrect predictions made by the classifier. This strategy forces the agent to learn from extremely challenging scenarios.

AE-RL is specifically designed to integrate supervised problems based on labeled datasets into a DRL framework. A major challenge for RL algorithms, especially when working with labeled datasets, is dataset unbalance. This problem is critical for all ML algorithms but particularly important for RL algorithms because they lack prior knowledge of the correct labels needed for weight adjustment and error minimization. RL algorithms learn by iterative trial and error, exploring state transitions. Consequently, when faced with a highly unbalanced data set, the algorithm has difficulty accurately classifying under-represented samples without overfitting the majority of the data.

Driven by the significant challenges posed by unbalanced datasets and exploration limitations, AE-RL has emerged as a compelling solution. The AE-RL mainly aims to optimize the cumulative reward by achieving balanced training in the classification tasks. To this end, a smart environment is used to provide informative samples, following the exploration–exploitation principle found in RL algorithms. This innovative sampling strategy facilitates balanced learning for infrequent samples. By overcoming the exploration challenges facing RL algorithms and effectively handling unbalanced datasets, AE-RL offers a promising approach to address these issues.

To our knowledge, there are currently no reports in the literature of intrusion detection systems for the IoT being explored based on the AE-RL concept. This study presents an innovative intrusion detection system designed specifically for IoT environments.

To assess the effectiveness of our proposal, we conducted tests on the BoT-IoT dataset, which includes a variety of IoT cyber-attack labels, such as denial of service (DoS), distributed denial of service (DDoS), data gathering, and data theft.

The present work's key contributions can be summarized as follows:

- Introducing a distinctive classifier model specifically designed for network intrusion detection.
- Presenting a classifier characterized by speed, flexibility, and exceptional performance metrics for prediction.
- Pioneering the use of adversarial RL for IoT to address training bias associated with unbalanced datasets.
- Providing a comparative analysis of the results obtained from our proposal versus existing models.

2 Related works

A large body of research in the literature investigated intrusion detection using various datasets. In what follows, we highlight studies on intrusion detection methods using ML, DL, and RL techniques. To provide a comprehensive overview, we categorize these methods according to their application to both traditional and IoT networks.

2.1 Traditional network approaches

A method for detecting DDoS attacks in traditional networks using an ensemble of classifiers was introduced [11]. The dataset was divided into several subsets, each independently processed by a series of classifiers. The results of each classifier were merged using a weighted majority voting approach, where distinct weights were assigned to each algorithm. The learning algorithm used in the classifier was back-propagation. This method achieved 99.4% accuracy in classifying attacks on several publicly accessible datasets.

In [12], the authors presented a deep learning-based intrusion detection technique to counter attacks on the UNSW-NB 15 dataset [13]. The system comprises five layers, each containing ten neurons. The model was trained using a tenfold cross-validation technique. Furthermore, the Gedeon feature ranking method was used to evaluate the significance of the features and identify the most influential ones. The results showed that the proposed method reached an accuracy of about 99% when applied to the chosen dataset.

In [14], a novel intrusion detection method was introduced to counter DoS attacks using the radial basis function as the neural networks' initial layer, which enabled the mitigation of training bias and improved the weight optimization technique. The suggested method reached an accuracy of 99.69% when evaluated on UNSW-NB 15 [13] and NSL KDD [15].

Furthermore, a comparable deep network architecture was introduced in [16], incorporating deep auto-encoder stacks for unsupervised feature learning and a random forest model for instance classification. The proposed method was assessed via the NSL KDD dataset [15].

In the RL context, an intrusion detection method was developed using a multi-agent architecture based on RL [17, 18]. Unlike the above works, their framework uses a hierarchical structure to identify anomalies, and the agent configuration is designed to be non-adversarial. To apply a look-up table, they discretized the state space. The results presented in their work are specific to their simulated network and cannot be directly compared to our findings.

Moreover, an actor-critical RL model incorporating temporal difference (TD) learning was proposed [19]. The model was specifically designed for classification tasks and used a fuzzy adaptive learning control network. It was evaluated using the well-known and simple Iris dataset.

In [20], an adversarial environment is described to create a classifier that deliberately induces errors by slightly modifying the training data. Next, the author of [21] investigated channel estimation and signal detection in nonlinear Orthogonal Frequency Division Multiplexing (OFDM) systems. The authors developed a pilot-assisted channel estimator based on the theory of Bayesian posterior minimum mean square error (MMSE). They also implemented quadrature amplitude modulation (QAM) signal detection using the principles of extrinsic expectation and extrinsic variance passing. The numerical results of the study confirmed the superior performance of the proposal, demonstrating its effectiveness with relatively low computational complexity.

2.2 IoT network approaches

In [22], the authors introduced a new approach that uses a DRL algorithm in conjunction with an unsupervised learning reward technique to build a pooled monitoring system for IoT networks. This integration enhances network security and improves the predictive capabilities of the DRL agent, especially in adaptive environments.

Koroniotis et al. [24] addressed the missing realistic IoT traffic in available datasets by establishing an IoT testbed and creating the BoT-IoT dataset. This dataset included instances of legitimate and simulated IoT traffic and numerous attacks. Malicious traffic was generated through attacks, such as DoS, DDoS, data gathering, and data theft. The authors developed a set of additional features by considering the joint entropy and correlation coefficient of the fundamental features.

These newly generated features were then fed into three different algorithms: support vector machine (SVM) and two DL techniques on the basis of recurrent neural networks (RNN) to assess the detection accuracy of these algorithms. A total of 5% of the original dataset was used to assess the accuracy of the binary classifiers that distinguish between normal and attack traffic for each subcategory. The results showed good accuracy levels. However, it is worth noting that the extracted dataset showed unbalanced classes, as certain attack categories had a significantly higher number of instances compared to the normal traffic class.

A novel scheme for distributed attack detection in IoT networks incorporating a fog computing layer was introduced [25]. This approach was based on a stacked auto-encoder-based unsupervised DL methodology. In this scheme, the fog nodes played a critical role in the parallel training and updating of the models and parameters.

The stacked auto-encoder model underwent pre-training via unlabeled training data to extract meaningful features from the data. These latent features were subsequently used on the testing data for classification purposes. This method was evaluated using the NSL KDD dataset [15] consisting of 41 features. By comparing the performance of their DL model with that of a shallow learning model (one without pre-training), the authors demonstrated that the DL model achieved better detection accuracy.

An innovative method for detecting intrusions in IoT environments networks based on anomaly was introduced in [26] by developing a feed-forward deep neural network (DNN) integrating a deep belief network (DBN) whose layers were pre-trained through an unsupervised learning technique and subsequently used as hidden layers within the DNN. To prove the effectiveness of their model, the authors implemented an IoT testbed consisting of six sensors that simulated a smart home scenario.

They then collected the network traffic generated by this testbed for evaluation purposes. The results revealed that the presented model reached high precision, recall, and F1 scores for various attacks compared to an existing intrusion detection method using inverse weight clustering.

In [27], the authors designed an intrusion detection approach using a DBN and incorporated the genetic algorithm to determine the optimal architecture for the DBN. To assess the performance of their scheme, they used the DARPA dataset [28] along with the NSL KDD dataset [15], which consists of various attacks, such as denial of service, root-to-local, user-to-root, and probe attacks. It is worth noting that these datasets do not contain specific traces of IoT traffic. The authors acknowledge that IoT devices

may be vulnerable to these attacks. Nevertheless, the absence of legitimate IoT traffic in the evaluation datasets leaves the effectiveness of their approach uncertain in the IoT context.

The method developed in [29] aims to detect DoS attacks on smart IoT sensors using average dependency estimators. A custom IoT testbed employing the Message Queuing Telemetry Transport (MQTT) protocol was implemented and used to introduce DoS attack scenarios. The dataset was generated by considering various packet-level features. The authors adopted two estimators, namely the averaged one-dependency estimator (A1DE) and the averaged two-dependency estimator (A2DE), and evaluated their performance against various classical ML algorithms. The results proved that the classifier outperformed other classification algorithms regarding accuracy in tests conducted on both the custom dataset and the BoT-IoT testbed [24].

Recently, a lightweight IDS that incorporates a new technique for data preprocessing and uses both ML and DL classifiers has been proposed [30]. The study focused on developing lightweight IDSs specifically tailored to sense and alleviate DDoS attacks in IoT networks. Two datasets were used to conduct experiments that aim to evaluate the proposed approach: the BOT-IoT dataset and the TON-IoT network dataset. Both datasets included several DDoS attacks. Several experiments were performed on each dataset, including the binary attack labels classification: One experiment covered all attack types, while the other targeted DDoS attacks.

Although numerous research efforts have been dedicated to intrusion detection in IoT networks, mainly based on machine learning approaches, DRL has emerged as a promising alternative. Unlike traditional methods, DRL allows autonomous agents to interact with their environment and make decisions without human intervention. In our study, we present an innovative approach to intrusion detection in IoT systems using an adversarial RL algorithm known for its superior predictive capabilities. We evaluate the performance of our proposed method using the Bot-IoT database.

3 Proposed model for IoT intrusion detection

Figure 1 depicts the proposed approach for intrusion detection. The process comprises several essential steps, including data acquisition, data preprocessing, data modeling, and anomaly detection. A detailed description of each of these steps is provided below.

3.1 Dataset description

The proposed intrusion detection model was evaluated using the BoT-IoT dataset presented in [24]. This dataset was chosen for several reasons:

- Real testbed deployment: The authors used a physical testbed consisting of virtual machines (VMs), simulated IoT sensors, and networking and security devices. This real-world setup adds authenticity to the dataset, making it more representative of real-world IoT environments.
- Simulated IoT traffic: The dataset includes legitimate traffic generated by simulated IoT devices. This traffic is designed to mimic the behavior of real IoT devices, providing a realistic representation of the communication patterns and data flows in a smart home scenario.

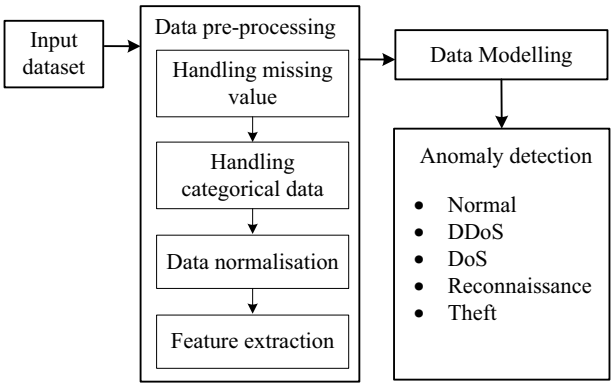


Fig. 1 Proposed approach for IoT intrusion detection. The figure illustrates the approach that we proposed

- Realistic attack traffic: Along with legitimate traffic, the dataset contains various types of malicious traffic representing different attack scenarios. These attacks were specifically designed and executed within the testbed to ensure they capture the characteristics of real-world IoT attacks.
- Labeled data: The dataset provides labeled data, indicating which instances correspond to legitimate traffic and which ones represent malicious activities. This labeling enables supervised ML techniques to train and evaluate intrusion detection models.

The testbed includes five types of IoT devices: lights, thermostats, refrigerators, garage doors, and a weather station. These devices are connected to an IoT hub in the cloud by communicating with an MQTT broker that uses the MQTT protocol to publish data.

Four categories with ten subcategories of malicious traffic are generated in the dataset. These categories include DoS/DDoS attacks over TCP/UDP/HTTP, probing attacks for data gathering (such as OS fingerprinting and port scanning), and data theft (including data exfiltration and keylogging). For more information on the testbed setup and detailed statistics of the attacks, one could refer to [24].

Note that only 5% of the original dataset was extracted and used [24] due to the cumbersome nature of the generated dataset (Bot-IoT). The extracted 5% was divided into training and test sets, and the data were saved as Cascading Style Sheets (CSS) files, which are publicly available and contain around 3 million records. Table 1 shows the statistics of the Bot-IoT dataset. As can be seen in Table 1, there are fewer normal instances compared to attack instances. Figure 2 visually illustrates this unbalanced nature by showing intrusion classes' frequency distribution (Normal, DDoS, DoS, Reconnaissance, and Theft).

3.2 Data preprocessing

3.2.1 Handling missing values

Dealing with missing values is among the first and most crucial stages in any data preprocessing method. In this particular dataset, some protocols, such as address resolution protocol (ARP), resulted in missing (not applicable) values for the source and destination

Table 1 Statistics of the BOT-IOT dataset

Class	Dataset		
	Training	**Testing**	**Total**
Normal	370	107	477
DDoS	1,541,315	385,309	1,926,624
DoS	1,320,148	330,112	1,650,260
Reconnaissance	72,919	18,163	91,082
Theft	65	14	79
Total	2,934,817	733,705	3,668,522

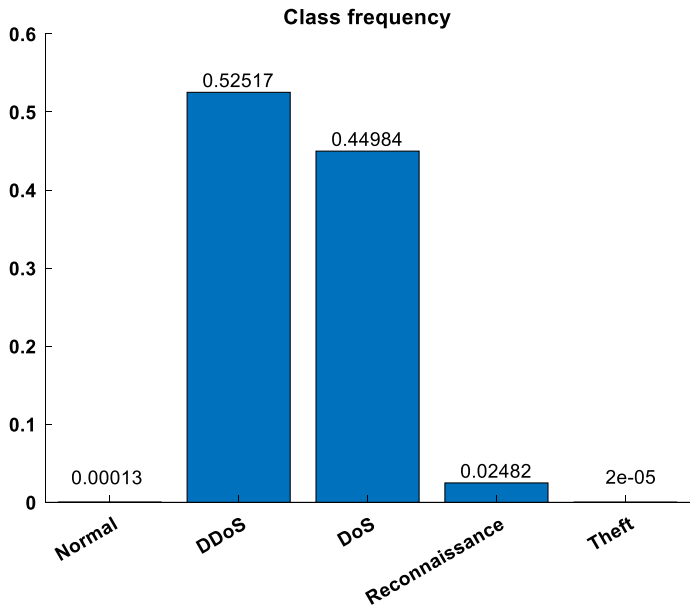


Fig. 2 Distribution of intrusion classes based on occurrence frequency. The figure illustrates the distribution rate of intrusions according to their frequency of appearance

port numbers. To address this issue, these missing values were intentionally set to (-1), an improper port number, to evaluate the dataset.

3.2.2 Handling categorical data

The categorical feature values were converted to sequential numeric values to facilitate the application of statistical techniques. For example, the "state" attribute contains categorical values such as "RST", "CON", and "REQ", which have been transformed into "1", "2", and "3", respectively.

This numerical representation allows for easier processing and analysis by ML algorithms.

3.2.3 Data normalization

Normalization was performed to scale the data within a certain range, such as [0, 1] while preserving the original data distribution. This crucial step facilitates the convergence of statistical models and DL methods to attain their goals by overcoming

challenges posed by local optima. The min–max transformation was used to perform the normalization process using the following formula:

$$x_N = (x - x_{\min}).\frac{(b-a)}{(x_{\max} - x_{\min})} + a \tag{1}$$

with x is the original value, $x_{\min}$ and $x_{\max}$ are, respectively, the minimum and maximum values from the original dataset. The variables a and b are the new minimum and maximum values, which define the range of interest. This transformation normalizes the data to the specified range. The relative relationships between the data points are preserved.

3.2.4 Feature extraction

To ensure the effectiveness of the proposed model and avoid overfitting problems, it is recommended to include a feature selection step. In this regard, both correlation coefficient and entropy have been used [24] to choose the most relevant features from the generated set. This feature selection process helps identify the features that contribute the most to the model's predictive power while reducing the potential for overfitting. According to [24], an ideal feature for the data set is characterized by a high entropy value and a low correlation value, suggesting that the feature lacks any redundant information shared with other features and is as independent as possible from other features. Based on these criteria and their analysis [24], the ten best features with the highest combined average correlation coefficient and joint entropy are listed in Table 2.

3.3 Model description

In what follows, we describe an innovative approach called AE-RL, which uses a classifier inspired by RL principles [9]: an agent receives information about the current state of the environment and takes action within that environment. This action ultimately results in a modification of the environment state. Then, the agent receives a reward from the environment, indicating the quality of the action with respect to a given goal.

Following [10], we used a simulation environment that generates states corresponding to random samples extracted from a labeled network intrusions dataset.

Table 2 Selected features and descriptions

Number	Feature	Description
1	seq	Argus sequence number
2	stddev	Standard deviation of aggregated records
3	NIC-SrcIP	Number of inbound connections per source IP
4	min	Minimum duration of aggregated records
5	state_number	Numerical representation of feature state
6	mean	Average duration of aggregated records
7	NIC_DstIP	Number of inbound connections per destination IP
8	drate	Destination-to-source packets per second
9	state	Source-to-destination packets per second
10	max	Maximum duration of aggregated records

The agent aims to accurately classify these states by predicting the intrusion label based on the information obtained from the simulated environment. The environment assigns rewards based on the agent's correct or incorrect predictions.

The previous modifications [10] result in a framework that allows the application of the known RL algorithm (Q-learning algorithm) [9] in classifying intrusions based on a dataset of prerecorded intrusion data.

The Q-learning algorithm operates by identifying the optimal Q-function for the agent. In this case, the agent represents the classifier. The Q-function calculates a value for each combination of state and action, representing the accumulated rewards for a given state when a given action is taken, considering the current policy. It may be computed iteratively, as follows [9]:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_A Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \right] \tag{2}$$

where S_t is the present state, A_t represents the present action, A_{t+1} is the succeeding action, R_{t+1} is the succeeding reward value, α denotes the learning rate, and γ signifies the discount factor. The discount factor has a value near zero as the states do not correlate, and the algorithm need not retain information about previous states.

A fully connected neural network (NN), as described in [31], is used to approximate the Q-function. The present state, consisting of the extracted features from the labeled dataset, serves as the input to this NN. The NN output denotes the Q-function for the available action set. In [31], the Deep Q-Network (DQN) algorithm is described in detail, followed by the AE-RL algorithm. Instead of using the expression of Eq. (2) to update the Q-function, the DQN model uses a related expression using a quadratic loss function: $\left(R_{t+1} + \gamma \max_A Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \right)^2$.

However, to address the problem of dataset unbalance, the framework has been modified [10]. These modifications are designed to imbue the environment with an intelligent action behavior that transcends the randomness associated with dataset sampling. Consequently, a novel model is introduced, which conducts a dynamic and intelligent re-sampling of the dataset throughout the training process. This updated approach aims to address the challenges posed by the unbalanced nature of the dataset.

In the novel approach described as AE-RL [10], an RL system incorporates a new agent into the existing environment. Deep Q-networks (DQN) are used to train the environment and the classifier agents concurrently.

The classifier agent functions as a classifier, while the environment agent functions like an attack selector for the subsequent training round. These two agents operate within an adversarial framework. The rewards originally given to the primary agent are transferred to the environmental agent. Consequently, the latter tries to influence the sample creation process to increase classification errors (reduces rewards) for the classifier agent, imposing the classifier to concentrate on handling more difficult illustrations.

The positive rewards for the main agent are changed to negative for the environment. This approach allows the environment agent to identify the classes where the main agent frequently experiences failure and increase the occurrence of samples from those classes with a certain probability. Two different Q-functions are used to

facilitate this process: $Q_c(s,a)$ optimizes the classifier, while $Q_e(s,a)$ optimizes the environment. The principle of evaluating the effectiveness of an action in a given state is the same for the two functions.

During the training process, a decreasing epsilon-greedy policy is used, where the exploration factor starts high and gradually decreases over episodes. Each episode, which can also be referred to as an epoch, includes the entire data set. Both the agent and the environment make action decisions based on their respective policies, with different lower bounds of epsilon for each case. To optimize detection performance, the classifier policy's epsilon lower bound is set low. Conversely, the environment policy bounds are set as hyperparameters.

The training process [10] follows a specific sequence that is outlined below and specified in the pseudo-code of Table 3:

Table 3 AE-RL algorithm

-Initiate $Q_c(s,a_{c_t})$ randomly

-Initiate $Q_e(s,a_{e_t})$ randomly

-Iterate (for every episode):

-Set the initial state value: s_0 =random sampling (dataset)

-Select the initial action a_{e_t} for the environment based on epsilon-greedy derived from $Q_e(s_t,a_{e_t})$.

-Substitute s_t with s_t =random sampling $\left(S\left(a_{e_t}\right)\right)$, where $S\left(a_{e_t}\right)$ represent all samples whose label is a_{e_t}

-Iterate over each time step, t , from 0 to N :

-Select the action a_{c_t} for the agent based on the policy derived from $Q_c(s_t,a_{c_t})$

-Take RL step obtaining $\left(r_{c_t},r_{e_t},s_{t+1}\right)$:

-Get rewards for both the agent and the environment: r_{c_t} , r_{e_t}

-Get next state :

-Select the next action $a_{e_{t+1}}$ for the environment based on epsilon greedy derived from $Q_e(s_t,a_{e_t})$

-Substitute s_{t+1} with s_{t+1} =random sampling $\left(S\left(a_{e_{t+1}}\right)\right)$, where $S\left(a_{e_{t+1}}\right)$ represent all samples whose label is $a_{e_{t+1}}$

-Q function update :

-Perform gradient descent on the loss function defined as:

$$\left(r_{e_t}+\gamma\max_{a_{e_{t+1}}}Q_e\left(s_{t+1},a_{e_{t+1}}\right)-Q_e\left(s_t,a_{e_t}\right)\right)^2$$

-Perform gradient descent on the loss function defined as:

$$\left(r_{c_t}+\gamma\max_{a_{c_{t+1}}}Q_c\left(s_{t+1},a_{c_{t+1}}\right)-Q_c\left(s_t,a_{c_t}\right)\right)^2$$

- The Q-functions of the agents (environment and classifier) are randomly reset. In addition, an initial state s_0 is chosen randomly from the dataset, and action values are obtained for this state through the Q-function.
- The environment chooses an action a_{e_t} according to its policy and the current state.
- The environment randomly chooses the current state (s_t) from the dataset. The action is the same as the one selected by the environment referred to as $S(a_{e_t})$ in Table 3, resulting in a feature-label pair (s_t, a_{e_t}).
- The agent attempts to classify the state according to its policy and associate an action (a_{c_t}), given the state chosen by the environment. This step is similar to a standard DQN algorithm.
- The action (a_{c_t}) representing the intrusion label is compared to the ground-truth label. If they match, the agent gets a positive reward; otherwise, the positive reward is assigned to the environment.
- The environment provides the new state (s_{t+1}) based on its action-value function and policy, as in a typical DQN algorithm, which results in the next feature and label pair $(s_{t+1}, a_{e_{t+1}})$.
- The classifier and environment agents' policy functions are adjusted following the DQN update rule using the obtained reward values and inferred next states.

These steps are repeated iteratively during the training process. The reward function used in this approach is a straightforward 1/0 reward scheme. A positive reward is associated with a value of $+1$, while a negative one is associated with 0.

Table 3 shows the algorithmic steps that deviate from a typical DQN algorithm, highlighted with bold lines. These additional steps are designed to facilitate the training of the environmental agent using an adversarial strategy.

The AE-RL algorithm in its final implementation [10] incorporates three additional refinements than the simplified version presented in Table 3:

- Double Deep Q-Network (DDQN) variant: The AE-RL's final algorithm is based on DDQN [32], an alternative of DQN that uses two networks for action selection (Classifier) and evaluation (Attacker). This modification enhances the decision-making process.
- Huber Loss: The agents are trained using the Huber loss as their selected loss function. The Huber loss resembles a quadratic loss up to a specific threshold and behaves linearly above that point. The Huber loss is used to mitigate the risk of gradients exhibiting explosive behavior and to promote more stable training.
- Dual Epsilon-Greedy Strategies: Since the implementation involves two agents, each represented by a different neural network, two epsilon-greedy strategies are used, ensuring that both agents have their own exploration–exploitation trade-off approach to action selection.

In this work, we followed an approach similar to [10]: We initialized the two networks with a high epsilon value and gradually decreased it over the training process.

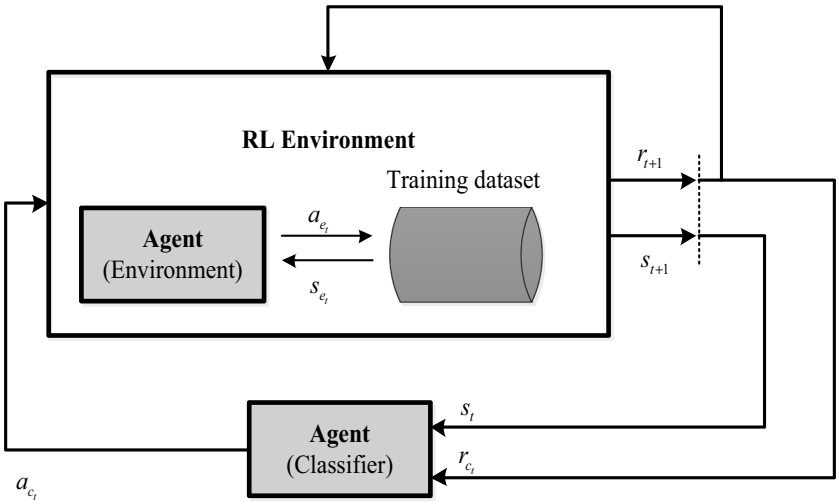


Fig. 3 Reinforcement Learning: Agent-Environment Interaction in an Intelligent System. The figure models the interaction of an agent with its environment according to the reinforcement learning approach

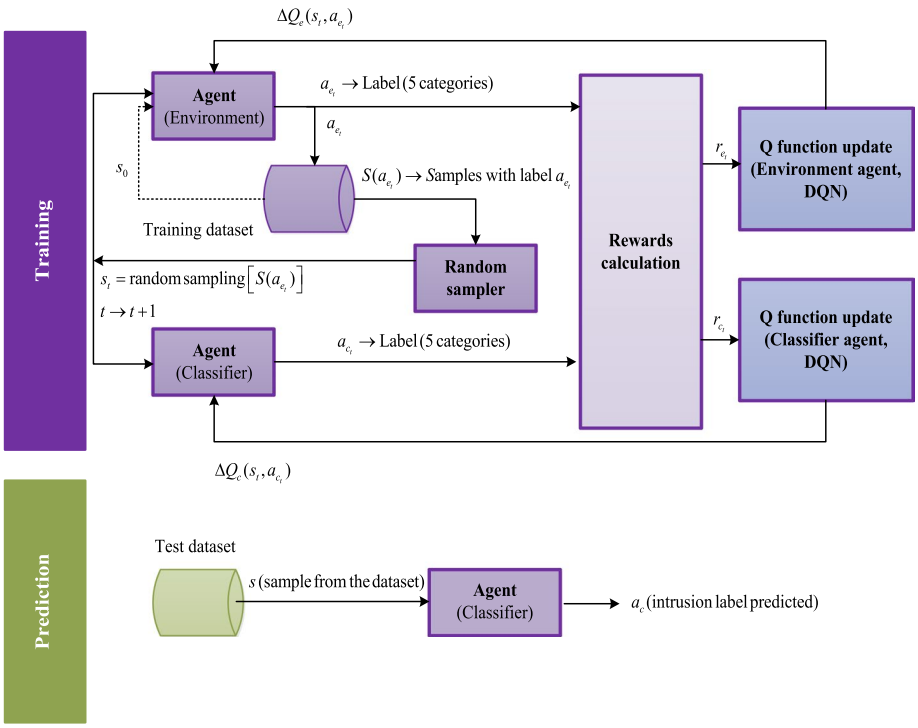


Fig. 4 Detailed AE-RL algorithm overview: Training and Prediction Phases. The figure shows an overview of our learning and prediction algorithm

Specifically, the epsilon value of the environmental network was reduced to 0.99 and that of the classifier network to 0.01.

Figure 3 illustrates the components of the algorithm, highlighting the role of the environment, which acts as a pseudo-agent in an opposing manner to the classifier. The algorithm consists of two phases: training and prediction (Fig. 4). The diagram

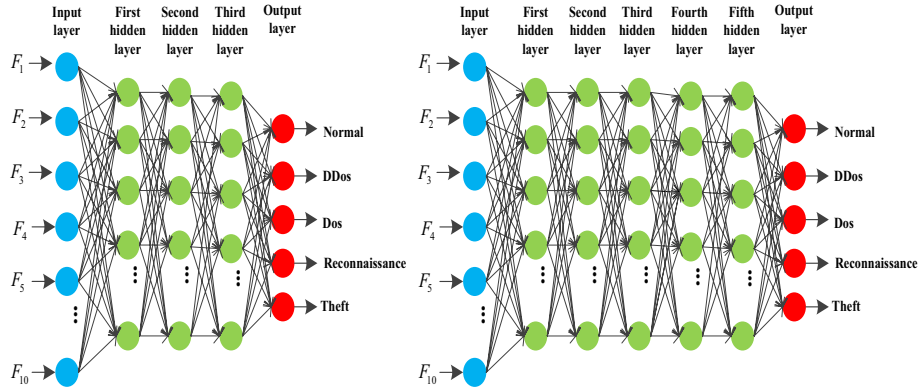


Fig. 5 Neural Network Architecture: Environment (Attacking Agent) on the Left Side, Classifier (Defending Agent) on the Right Side. The diagram provides an insight into the neural network architecture, showing the attacking agent's environment on the left side and the defending agent's classifier on the right side

Table 4 Parameter configuration used in our RL-AE model

Parameters	Value	
	Classifier agent (defender)	Environment agent (attacker)
Epsilon (exploration)	1	1
Min epsilon	0.01	0.99
Learning rate α	0.01	0.2
Discount factor γ	0.001	0.001
Hidden Layers	5	3
Hidden size	100	100

shows the ongoing process, where s_0 denotes the initial circumstances. Moreover, the primary state is selected arbitrarily from the dataset. Only the trained classifier agent is used during the prediction stage. The environment and the classifier agents are constructed via neural networks, as illustrated in Fig. 5.

Regarding the neural network architecture used by the environment agent (attacker) and the classifier agent (defender), we conducted experiments with different layer configurations: based on the results, we finalized the architecture with a relatively shallow neural network. Specifically, the attacker agent used a 3-layer neural network, while the defender agent used a 5-layer neural network (Fig. 5).

The layers of the two networks consisted of 100 units each. This simplified architecture reached efficient response times, while RL training effectively tuned the weights and biases for optimal performance. The parameter configurations used in our model are depicted in Table 4.

3.4 Machine and deep learning analysis techniques

We used machine and deep learning models to evaluate the performance of Bot-IoT in training a classifier. The models used included support vector machine (SVM),

Naïve Bayes (NB), logistic regression, artificial neural network (ANN), recurrent neural network (RNN), and long-short-term memory (LSTM).

SVM This model assumes that each data instance can be understood as a coordinate in an N-dimensional space, where N is the number of features. During training, it searches for a hyperplane that optimally divides the data into distinct groups (classes) and maximizes the margin. In our study, we used an SVM classifier with a linear kernel [33].

Logistic Regression Logistic regression, expressed mathematically, is a model that predicts $P(Y = 1)$ as a function of X. Typically associated with binary logistic regression involving data points of 0 or 1, which determine the outcome into two groups, it can also extend its application to handle multiple classifications for more than two desired outcomes [34]. Within the Logistic Regression (LR) model, characterized by the parameter θ, we can derive the likelihood and log-likelihood of the data X, y.

$$h\theta(x) = \sigma(\theta^T X) \tag{3}$$

h represents the hypothesis, x denotes the input feature vector, θ stands for Logistic Regression parameters, σ is the sigmoid function or threshold function, and r defines the threshold sigmoid function.

$$\sigma(r) = \frac{1}{1 + e^{-r}} \tag{4}$$

here, r corresponds to the term $\theta^T X$ from the previous equation (Eq. 3), and the resulting value falls within the range of 0 to 1 [26].

Naïve Bayes Naïve Bayes classification is a Bayesian classification technique using the Bayes theorem. Under the premise that the data is uniformly distributed, NB calculates the conditional probability of a class label given a data set. The Bayes theorem also presents a logical method for calculating this conditional probability under the assumption of feature independence, where each specific set is considered an independent feature of all other features in the data set. The Bayes theorem was formulated in [35] as:

$$P(u/V) = \frac{P(V/u)P(u)}{P(V)} \tag{5}$$

$P(u/V)$ denotes the posterior probability of h if h is given. $P(V/u)$ defines the probability of V given u as the maximum likelihood. $P(u)$ denotes the prior probability of hypothesis *'u'*. $P(V)$ defines the probability of the training data used in the data frame (evidence). The Naive Bayes method used as a classifier in the above statement is considered a strategy consisting of a family of algorithms with the common principle that each pair of features to classify is autonomous.

Artificial Neural Networks (ANNs) Many trial-and-error iterations are required before a satisfactory model is obtained. Using the Dense class in Keras, the goal is to construct a Fully Connected Network, a structured forward neural network that transmits weight values from each neuron as output to the following layer after processing inputs from neurons in the previous layer [36]. The first argument for the dense layer is the neuron. The activation argument facilitates the specification of the activation function. After defining the model, compilation becomes mandatory. TensorFlow was used in

the experiments for the compilation of the general-purpose Artificial Neural Network (ANN) model.

RNN Integrates a memory component into its architecture [37]. The output of an RNN at each iteration is both dependent on the current input and the output of the hidden state from the previous iteration. What distinguishes RNNs from other neural networks is their dependence on both the current input and past outputs, making them well-suited for handling temporal data, a characteristic present in our dataset. Typical applications of RNNs include tasks, such as machine translation, speech recognition, and image description.

LSTM is a special type of recurrent neural network. It uses internal cells designed to maintain a form of memory, making LSTMs well-suited for identifying associations over significant temporal distances [38]. LSTMs address the shortcomings of traditional RNNs, such as the vanishing gradient and exploding gradient problems, by introducing a memory cell responsible for managing updates to the model's memory. This enhancement positions LSTMs as effective tools for capturing long-term dependencies in data.

4 Results and discussion

This section compares the proposed (AE-RL) algorithm with several ML models on the BOT-IoT dataset. To ensure a comprehensive evaluation, we selected commonly used ML and DL models, including SVM, Naive Bayes, Logistic Regression, RNN, ANN, and LSTM. To evaluate the predictive performance of these models, we considered the metrics precision, accuracy, recall, and F1 score.

When dealing with multi-class classification problems, it is important to consider two approaches to presenting results: "aggregated" and "one vs. the rest". The latter approach centers on individual classes separately, grouping the rest of the classes into a single substitute category, thus simplifying the task by converting the problem into a binary classification for each class. However, the aggregated results summarize of the overall performance across all classes. Several methods are available for aggregation, including micro, macro, sampling, and weighted averaging. These methods differ in how the averaging process is performed. Here, to compute the aggregated F1 score, precision, and recall, we relied on the weighted averaging performed by Scikit-Learn.

Let us start with the multi-classification results. First, we aimed to examine how our proposal addressed the intrusion detection problem. Table 5 shows the performance results of an AE-RL algorithm using the One vs. the Rest approach for multi-classification tasks. It is worth noting that the algorithm achieves high accuracy for most classes, especially for the Normal and Theft (0.9998 and 0.9992, respectively) classes.

Table 5 Multi-classification performance scores of the AE-RL algorithm: One versus The rest metrics

Class	Accuracy	F1-score	Precision	Recall
Normal	0.9998	0.5961	0.4246	1.0
DDoS	0.9459	0.9476	0.9643	0.9314
DoS	0.9473	0.9421	0.9305	0.9540
Reconnaissance	0.9931	0.8750	0.7938	0.9747
Theft	0.9992	0.0491	0.0251	1.0

This finding indicates the successful classification of instances from these classes, which are minority classes in the dataset. "Reconnaissance" also shows a high accuracy of 0.9931. However, the "DDoS" and "DoS" classes have slightly lower accuracies of 0.9459 and 0.9473, respectively. The F1 scores provide valuable insight into the algorithm's efficiency regarding precision and recall for each class. The Normal class has a moderate F1 score of 0.5961, indicating a reasonable balance between precision and recall.

This result suggests that the algorithm achieved satisfactory accuracy in correctly classifying instances from the "Normal" class, taking into account both false positives and false negatives.

Moreover, the "Reconnaissance" class has a relatively high F1 score of 0.8750, indicating a good balance between precision and recall. The algorithm achieves accurate predictions for instances belonging to this class, with a strong compromise between false positives and false negatives. In contrast, the "DDoS" and "DoS" classes have higher F1 scores of 0.9476 and 0.9421, respectively. These scores suggest a better balance between precision and recall for these classes, indicating the algorithm's success in accurately classifying instances while minimizing both false positives and false negatives. However, the "Theft" class has a very low F1 score of 0.0491, indicating a significant imbalance between precision and recall. Further analysis and improvements may be needed to enhance the algorithm's performance in the "Theft" class in particular.

In the following analysis, we compare the performance of our proposal with the multilayer perceptron (MLP) algorithm. Figure 6 provides the results obtained in terms of accuracy, F1-score, precision, recall, correctness, and false alarm probability.

As can be seen in Fig. 6, the proposed approach outperforms the MLP approach with higher accuracy (0.9427 vs. 0.8164), F1 score (0.9433 vs. 0.8050), precision (0.9448 vs. 0.8250), recall (0.9427 vs. 0.8164), correctness (0.94 vs. 0.798), and lower false alarm

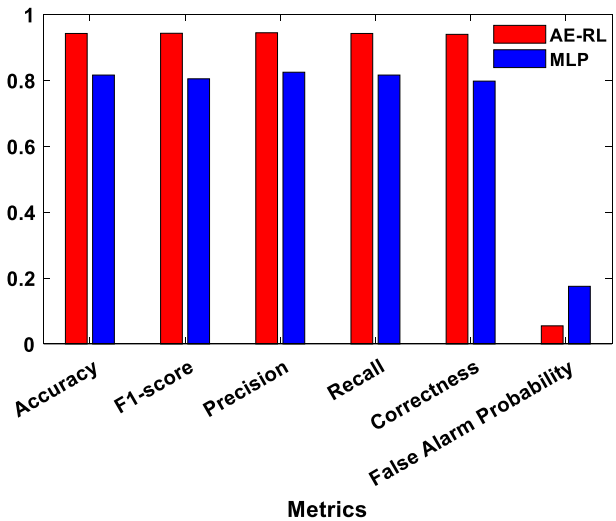
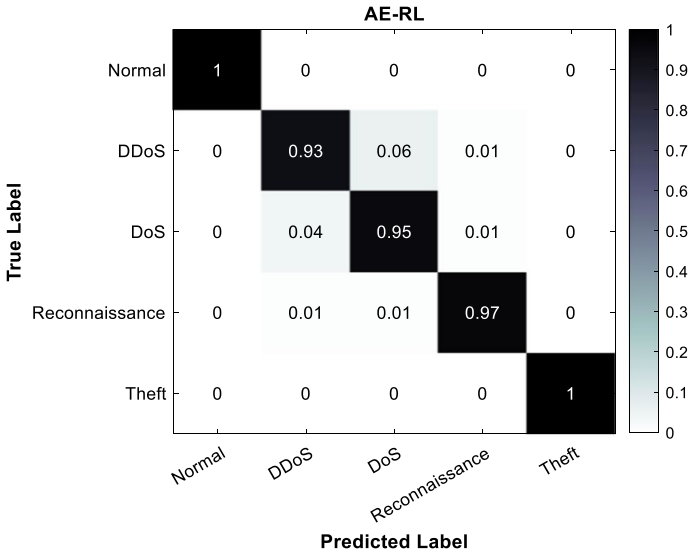


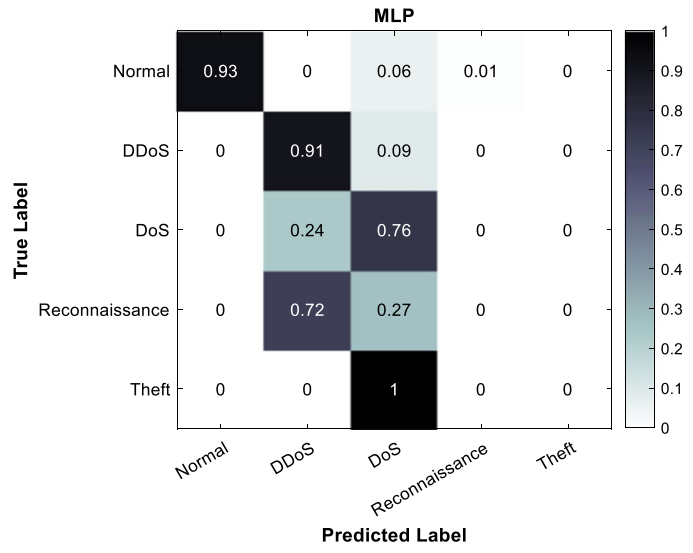
Fig. 6 Comparison of performance metrics between the proposed and MLP models: The figure presents a comprehensive comparative analysis of performance metrics, including precision, accuracy, recall, F1-score, correctness, and false alarm probability, between the proposed model and the MLP (Multi-Layer Perceptron) model

probability (0.055 vs. 0.175). These findings highlight the effectiveness of the proposed approach in accurately classifying instances and maintaining a good balance between precision and recall. Overall, the proposed model outperforms the MLP model in all evaluated metrics, indicating its superiority in classification performance.

The AE-RL model confusion matrix is depicted in Fig. 7a for the Bot-IoT dataset and the MLP algorithm (Fig. 7b) with a similar structure to the policy network in AE-RL. The results show that AE-RL achieves the lowest number of false negatives (0 for the "Normal" and "Theft" attacks). However, the MLP algorithm shows 100%



(a)



(b)

Fig. 7 Confusion matrix of AE-RL and MLP algorithms on the BoT-IoT dataset: **a** AE-RL, **b** ML. The figure illustrates the confusion matrix compares the AE-RL and MLP algorithms on the BoT-IoT and AE-RL and b ML bases

false negatives for the "Theft" attack, representing an undesirable characteristic for a detection system. In this context, false negatives indicate an intrusion that went undetected or was classified as a different type of attack. In particular, the MLP misclassifies all instances of "Theft" as "DoS" attacks.

In contrast, AE-RL focuses on improving the classification of minority classes (less frequent classes). This improvement can be seen in Fig. 7, where AE-RL successfully reduces the number of false negatives for these less frequent classes. However, it is accompanied by a slightly higher false positive rate for the "DoS" class, with a value of 0.01 compared to 0 for the MLP algorithm.

To prove the effectiveness of our proposal in binary classification, we compared our results with two recent studies on IoT intrusion detection based on the same dataset [24] and [30], focusing on two cases: Normal vs. All Attacks and Normal vs. DDoS Attacks. Table 6 shows the performance results of binary classification on the Bot-IoT dataset for the two specific cases.

As can be seen in Table 6, in the case of Normal vs. All Attacks, [24] achieves accuracies of 88.37% with SVM and 99.74% with RNN using 10-best features. Similarly, [30] achieves accuracies ranging from 88 to 98% using different classifiers and 20-best features.

However, our work outperforms these studies in both cases. Using the 10-best features and the AE-RL algorithm, we achieved an accuracy of 99.98% for Normal vs. All Attacks, surpassing all previous approaches.

In contrast, while SVM, ANN, and LSTMs achieved higher accuracy (99, 95, and 95%, respectively) compared to our AE-RL approach (94.59%) in IoT intrusion detection, it is essential to highlight the distinguishing characteristics of our approach. The autonomous interactions with the environment and real-time application capabilities of our approach are crucial for effective IoT intrusion detection. Although our scheme's accuracy may be slightly lower, these inherent strengths make our approach well-suited to address the dynamic nature of IoT systems and respond to potential intrusions promptly.

In short, the results reached by AE-RL can be summarized as follows:

Table 6 Binary classification performance results on the BOT-IOT dataset in two cases: Normal-All attacks and Normal-DDOS attacks

Ref and years	Features	Classifiers	Accuracy	
			Case 1: normal-all attacks	**Case 2: normal-DDoS attack**
[24], 2020	10-best features	SVM	88.37	–
		RNN	99.74	–
[30], 2023	20-best features	Linear SVM	88	99
		Naïve Bayes	88	88
		Logistic Regression	98	88
		ANN	98	95
		LSTM	98	95
This work	10-best features	AE-RL	**99.98**	94.59

- AE-RL achieves classification results equal to or better than state-of-the-art classifiers.
- AE-RL is particularly effective in handling unbalanced datasets, as it is explicitly developed to diminish under-represented classes' classification errors.
- AE-RL is especially valuable when the impact of false negatives can be substantial. False negatives are often found in classes that are less frequent because algorithms tend to prioritize achieving the uppermost overall prediction, often involving the increased prediction of the most frequent classes, thus more false negatives in the less frequent ones. This finding is particularly relevant in the context of intrusion detection issues, where some types of attacks may match less frequent classes.

5 Conclusion

In the field of IoT security, the detection and mitigation of network attacks are crucial to monitor and control unwanted traffic flows. Numerous researchers have proposed ML models to spot and mitigate attack traffic in IoT networks. However, the unique characteristics of intrusion detection in IoT networks necessitate exploring novel models capable of overcoming the challenges that IDS poses to ML algorithms. These challenges include handling unbalanced, noisy, and complex datasets subject to varying conditions.

This study introduced an innovative approach by integrating the RL framework into the IDS domain. RL algorithms have shown remarkable success in various fields, such as robotics, finance, video games, and business operations. This research aimed to extend the application of RL algorithms to IDS. The projected model, known as AE-RL, incorporates the advantages of both supervised and reinforcement learning frameworks to establish a simulation environment that conforms to RL principles.

The AE-RL model can interact with a dataset of network feature samples and their corresponding intrusion labels. By employing an optimized policy, it aimed to achieve superior classification results. Notably, an adversarial strategy was used as the underlying learning mechanism in this environment. This research represents a pioneering application of adversarial RL for IoT intrusion detection.

This paper presents several notable contributions. First, it introduces a novel architecture that combines supervised and adversarial RL models, offering a promising approach to address the challenges of prediction problems in demanding IoT networks. Our model proves exceptional predictive capabilities compared to the MLP algorithm, demonstrating its potential as an effective solution for intrusion detection. In addition, a comprehensive comparison with various ML models was conducted, providing valuable insights into their strengths and weaknesses in the context of IoT network security. Furthermore, the adaptability of the proposed model to handle highly unbalanced datasets is highlighted, addressing a common problem in intrusion detection.

Several interesting avenues can be explored in future research. One potential avenue is implementing a prioritized experience replay mechanism in the environment. This approach, inspired by prioritized experience replay techniques, would involve optimizing the prioritization of samples using an additional agent trained with RL algorithms. By prioritizing some samples based on their importance or relevance, the learning process could be further enhanced, leading to improved performance of the proposed model. Another important future step is to compare the proposed model

with several algorithms. This comparative analysis would allow a deeper understanding of the strengths and weaknesses of the model as compared to existing approaches. In addition, applying the proposed model to other IoT datasets would help evaluate its performance in different contexts and scenarios, further validating its effectiveness and generalizability.

6 Methods/experimental

In this paper, we introduce an innovative architecture that blends supervised and adversarial RL models, offering a compelling solution for prediction challenges in demanding IoT networks. Our model outperforms the MLP algorithm, showcasing its potential as an efficient intrusion detection tool. A thorough comparison with various ML models provides valuable insights into their strengths and weaknesses in IoT network security. Furthermore, we highlight the proposed model's adaptability to handle highly unbalanced datasets, addressing a common issue in intrusion detection. This research presents a comprehensive exploration of a novel approach with potential implications for strengthening IoT network security.

Abbreviations

IoT	Internet of Things
DRL	Deep reinforcement learning
RL	Reinforcement learning
DDoS	Distributed denial of service
IDS	Intrusion detection system
ML	Machine learning
AE-RL	Adversarial environment using reinforcement learning
DoS	Denial of service
SVM	Support vector machine
RNN	Recurrent neural networks
DNN	Forward deep neural network
DBN	Deep belief network
MQTT	Message queuing telemetry transport
A1DE	The averaged one-dependency estimator
A2DE	The averaged two-dependency estimator
VMs	Virtual machines
ARP	Address resolution protocol
NN	Neural network
DQN	Deep Q-network
DDQN	Double deep Q-network
MLP	Multi-layer perceptron

Author contributions

This paper introduces a hybrid architecture, merging supervised and adversarial RL models for robust IoT network prediction. Outperforming MLP in intrusion detection, it undergoes comprehensive ML model comparisons, revealing insights into IoT security. Notably, its adaptability to handle unbalanced datasets addresses a common intrusion detection challenge.

Funding

This research was funded by the Deanship of Scientific Research at Princess Nourah bint Abdulrahman University, through the Research Funding Program, Grant No. (FRP-1444–29).

Availability of data and materials

The datasets used and analyzed during the current study are publicly available.

Declarations

Competing interests

The authors declare that they have no competing interests.

Received: 11 October 2023 Accepted: 5 April 2024
Published online: 04 May 2024

References

1. N. Moustafa, B. Turnbull, K. K. R. Choo, Towards Automation of Vulnerability and Exploitation Identification in IIoT Networks. Proc. - 2018 IEEE Int. Conf. Ind. Internet, ICII 2018, no. Icii, pp. 139–145, 2018, doi: https://doi.org/10.1109/ICII.2018.00023
2. N. Moustafa, B. Turnbull, K.K.R. Choo, An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of Internet of Things. IEEE Internet Things J. **6**(3), 4815–4830 (2019). https://doi.org/10.1109/JIOT.2018.2871719
3. C. Kolias, G. Kambourakis, A. Stavrou, J. Voas, DDoS in the IoT: Mirai and Other Botnets. Computer **50**(7), 80–84 (2017)
4. M.H. Bhuyan, D.K. Bhattacharyya, J.K. Kalita, Network anomaly detection: methods, systems and tools. IEEE Commun. Surv. Tutorials **16**(1), 303–336 (2014). https://doi.org/10.1109/SURV.2013.052213.00046
5. J.J.P.C. Rodrigues et al., Enabling technologies for the internet of health things. IEEE Access **6**, 13129–13141 (2018). https://doi.org/10.1109/ACCESS.2017.2789329
6. K.A.P. da Costa, J.P. Papa, C.O. Lisboa, R. Munoz, V.H.C. de Albuquerque, Internet of Things: a survey on machine learning-based intrusion detection approaches. Comput. Netw. **151**, 147–157 (2019). https://doi.org/10.1016/j.comnet.2019.01.023
7. M.M.E. Mahmoud et al., Enabling technologies on cloud of things for smart healthcare. IEEE Access **6**, 31950–31967 (2018). https://doi.org/10.1109/ACCESS.2018.2845399
8. K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath, Deep reinforcement learning: a brief survey. IEEE Signal Process. Mag. **34**(6), 26–38 (2017)
9. R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction. Robotica **17**(2), 229–235 (1999)
10. G. Caminero, M. Lopez-Martin, B. Carro, Adversarial environment reinforcement learning algorithm for intrusion detection. Comput. Networks **159**, 96–109 (2019). https://doi.org/10.1016/j.comnet.2019.05.013
11. P.A. Raj Kumar, S. Selvakumar, Distributed denial of service attack detection using an ensemble of neural classifier. Comput. Commun. **34**(11), 1328–1341 (2011). https://doi.org/10.1016/j.comcom.2011.01.012
12. M. Al-Zewairi, S. Almajali, A. Awajan. Experimental evaluation of a multi-layer feed-forward artificial neural network classifier for network intrusion detection system. Proc. - 2017 Int. Conf. New Trends Comput. Sci. ICTCS 2017, vol. 2018-January, pp. 167–172, 2017, doi: https://doi.org/10.1109/ICTCS.2017.29
13. N. Moustafa, J. Slay, The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Inf. Secur. J. **25**(1–3), 18–31 (2016). https://doi.org/10.1080/19393555.2015.1125974
14. B.A. Bhuvaneswari, S. Selvakumar, Deep radial intelligence with cumulative incarnation approach for detecting denial of service attacks. Neurocomputing **340**, 294–308 (2019). https://doi.org/10.1016/j.neucom.2019.02.047
15. M. Tavallaee, E. Bagheri, W. Lu, A. A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. in IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 1–6
16. N. Shone, T.N. Ngoc, V.D. Phai, Q. Shi, A deep learning approach to network intrusion detection. IEEE Trans. Emerg. Top. Comput. Intell. **2**(1), 41–50 (2018). https://doi.org/10.1109/TETCI.2017.2772792
17. A. Servin, D. Kudenko, *Multi-agent reinforcement learning for intrusion detection* (University of York, 2009)
18. K. Malialis, *Distributed Reinforcement Learning for Network Intrusion Response* (University of York, 2014)
19. X. H. Quah, C. Q. G. Leedham, Pattern classification using fuzzy adaptive learning control network and reinforcement learning. in Proceedings of the 9th International Conference on Neural Information Processing, vol. 3, pp. 1439–1443
20. S. Huang, N. Papernot, I. Goodfellow, Y. Duan, P. Abbeel, Adversarial attacks on neural network policies. in 5th International Conference on Learning Representations, ICLR 2017, (2017)
21. T. Liu, H. Jiang, Z. Chen, H. Zhang, X. Yang, F. Shu, Nonlinear channel estimation and signal detection for quantized OFDM System. IEEE Commun. Lett. **27**(10), 2772–2776 (2023). https://doi.org/10.1109/LCOMM.2023.330231
22. M. S. Frikha, S. M. Gammar, A. Lahmadi, Multi-attribute monitoring for anomaly detection: a reinforcement learning approach based on unsupervised reward. in 2021 10th IFIP International Conference on Performance Evaluation and Modeling in Wireless and Wired Networks (PEMWN), Ottawa, ON, Canada, 2021, pp. 1–6, doi: https://doi.org/10.23919/PEMWN53042.2021.9664667
23. J. Chen, Y.-C. Liang, H.V. Cheng, W. Yu, Channel estimation for reconfigurable intelligent surface aided multi-user mmWave MIMO systems. IEEE Trans. Wireless Commun. **22**(10), 6853–6869 (2023). https://doi.org/10.1109/TWC.2023.3246264
24. N. Koroniotis, N. Moustafa, E. Sitnikova, B. Turnbull, Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. Futur. Gener. Comput. Syst. **100**, 779–796 (2019). https://doi.org/10.1016/j.future.2019.05.041
25. A. Abeshu, N. Chilamkurti, Deep learning: the frontier for distributed attack detection in fog-to-things computing. IEEE Commun. Mag. **56**(2), 169–175 (2018). https://doi.org/10.1109/MCOM.2018.1700332
26. G. Thamilarasu, S. Chawla, Towards deep-learning-driven intrusion detection for the Internet of Things. Sensors (2019). https://doi.org/10.3390/s19091977
27. Y. Zhang, P. Li, X. Wang, Intrusion detection for IoT based on improved genetic algorithm and deep belief network. IEEE Access **7**, 31711–31722 (2019). https://doi.org/10.1109/ACCESS.2019.2903723
28. J. Mchugh, Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory. ACM Trans. Inf. Syst. Secur. **3**(4), 262–294 (2000). https://doi.org/10.1145/382912.382923
29. Z.A. Baig, S. Sanguanpong, S.N. Firdous, N. Van Vo, T.G. Nguyen, C. So-In, Averaged dependence estimators for DoS attack detection in IoT networks. Futur. Gener. Comput. Syst. **102**, 198–209 (2020)

30. S.A. Khanday, H. Fatima, N. Rakesh, Implementation of intrusion detection model for DDoS attacks in lightweight IoT networks. Expert Syst. Appl. **215**, 119330 (2023). https://doi.org/10.1016/j.eswa.2022.119330
31. V. Mnih et al., Playing Atari with deep reinforcement learning. arXiv:1312.5602v1 [cs.LG], pp. 1–9, 2013, [Online]. Available: http://arxiv.org/abs/1312.5602
32. H. Van Hasselt, A. Guez, D. Silver. Deep reinforcement learning with double Q-Learning. arXiv:1509.06461v3 [cs.LG], pp. 2094–2100, 2016, doi: https://doi.org/10.1609/aaai.v30i1.10295.
33. D. Meyer, F. Wien, Support vector machines. R News **1**(3), 23–26 (2001)
34. J. R. Brzezinski, G. J. Knafl, Logistic regression modeling for context-based classification. in Proceedings of the Tenth International Workshop on Database and Expert Systems Applications (DEXA 99), pp. 755–759, 1999
35. M. J. Islam, Q. M. Jonathan Wu, M. Ahmadi, M. A. Sid-Ahmed, Investigating the performance of Naive-Bayes classifiers and K-nearest neighbor classifiers. in 2007 International Conference on Convergence Information Technology (ICCIT 2007), pp. 1541–1546, 2007
36. M.M. Saritas, A. Yasar, Performance analysis of ANN and Naive Bayes classification algorithm for data classification. Int. J. Intell. Syst. Appl. Eng. **7**(2), 88–91 (2019). https://doi.org/10.18201/ijisae.2019252786
37. S. Grossberg, Recurrent neural networks. Scholarpedia **8**(2), 1888 (2013)
38. K. Greff, R.K. Srivastava, J. Koutník, B.R. Steunebrink, J. Schmidhuber, LSTM: a search space odyssey. IEEE Trans. Neural Netw. Learn. Syst. **28**(10), 2222–2232 (2017)

Publisher's Note