International Journal of Safety and Security Engineering
Vol. 15, No. 11, November, 2025, pp. 2229-2237
Journal homepage: http://iieta.org/journals/ijsse

Enhanced Intrusion Detection in Social IoT Environments via Double Deep Q-Learning

Sabah M. Alturfi[1*], Nibras Talib Mohammed[2], Nagham Habeeb Shakir[3]

[1] College of Law, University of Kerbala, Kerbala 56001, Iraq
[2] Department of Statistics, College of Administration and Economics, University of Kerbala, Kerbala 56001, Iraq
[3] College of Computer Science and Information Technology, University of Kerbala, Kerbala 56001, Iraq

Corresponding Author Email: sabah.m@uokerbala.edu.iq

https://doi.org/10.18280/ijsse.151104

Received: 27 September 2025
Revised: 10 November 2025
Accepted: 21 November 2025
Available online: 30 November 2025

Keywords:
Social Internet of Things, Intrusion Detection Systems, Double Deep Q-Learning, reinforcement learning, network security, anomaly detection

ABSTRACT

The growing demand for the Internet of Things (IoT), especially the Social Internet of Things (SIoT), has introduced a new dimension of technological challenges. As the use of SIoT expands, it has raised significant concerns about the security of networks connecting smart devices, particularly against sophisticated network breaches. Traditional Intrusion Detection Systems (IDS) struggle to dynamically adapt to the complex, high-dimensional environments of SIoT. This paper proposes that a more effective Network Intrusion Detection System (NIDS) can be developed using Double Deep Q-Learning (DDQL). The reinforcement learning (RL) method overcomes the overestimation bias commonly found in traditional Q-learning techniques, offering a more accurate and reliable detection model. The system was trained and tested using the CICIDS2017 dataset, which includes real-world network traffic and attack scenarios. By framing intrusion detection as a sequence of decisions, the DDQL learning-based agent can learn and predict malicious behaviors more accurately with much fewer false positives. Experiments show that the developed method provides better results in terms of detection accuracy, precision, and F1-score than traditional machine learning (ML) classifiers and standard Deep Q-Learning (DQL) models. These enhancements showcase the system's improvements in real-time intrusion detection and response capabilities. Moreover, the scalability and flexibility of the system make it a powerful instrument to identify intrusions in the dynamic and fast-growing S-IoT environments where attack scenarios change rapidly, and attack vectors can be very wide.

1. INTRODUCTION

The Social Internet of Things (SIoT) gr16ows from the conventional Internet of Things (IoT), where smart objects manage and establish social relations to increase collaboration, context perceptiveness, and intelligent services provisioning [1]. This has attracted huge interest, e.g., in health care, intelligent transportation systems, and industrial automation [2, 3]. Nonetheless, security issues are becoming even more relevant as the number of devices that have an IP address grows and SIoT networks are dynamic, decentralized, and heterogeneous [4].

Botnet, Denial of Service (DoS), and data tampering stand as main threats to SIoT environments, taking advantage of the high communication rate among devices and the node mobility [5]. Conventional security analogues such as firewalls and antivirus software are no longer sustainable since they tend to be resource-exorbitant; they also fail to cater to the current-day gravity of the threat landscape [6]. In response, there is a need for new lightweight and innovative Intrusion Detection Systems (IDSs) within the context of SIoT.

Intrusion detection approaches are typically categorized as signature-based and anomaly-based detectors. Signature-based IDSs work well for known attacks, but are not effective against zero-day attacks and need regular updates [7]. Anomaly-based systems, in comparison, have the potential to detect new attacks but suffer from less scalability and high false positives, so in practice they are not as effective as signature-based systems [8, 9]. Most existing IDSs do not consider the special social behaviors in SIoT and cannot well adapt to the networked environment with such social behaviors, laying insufficiency for detecting attacks effectively. To overcome these deficiencies, an increasing number of researchers are adopting adaptive methods that make use of machine learning (ML) and deep learning (DL) techniques [10, 11]. Well, these techniques solve the problem of manual feature engineering but might still lack in generalization. DL methods, like CNNs and RNNs, show potential in achieving better efficiency as they can automatically learn and capture useful characteristics of raw

traffic data [12, 13]. Nevertheless, despite these progressions, the computation-intensive nature of both ML and DL solutions continues to present considerable struggles for real-time systems as they often lack the capacity for a rapid response against advancing adversarial methodologies.

Reinforcement learning (RL) has recently shown the potential in facilitating real-time online learning under dynamic environments, which is an attractive feature, especially for internet applications. Supervised learning models depend on massive amounts of labeled information in dealing with pattern recognition, while RL simply interacts with the environment and learns through trial and error. In this setting, an RL agent is given feedback in terms of reward or penalties based on its actions that directly influence the environment. The agent gradually learns to take actions in a way that maximizes the sum of rewards because it is designed to converge on an optimal policy that reacts to changes in its environment in the most favorable way [14]. This adaptability property renders RL particularly well-suited for Network Intrusion Detection System (NIDS) in highly volatile and unpredictable systems, such as SIoT environments.

Deep Q-Learning (DQL) is mostly used in an IDS model for extensive state-action spaces. Nevertheless, the DQL algorithm is affected by Q-value overestimation that can affect decision-making. To mitigate the instability in standard DQL, Double Deep Q-Learning (DDQL) decouples action selection and evaluation by using two distinct neural networks [15]. DDQL is highly promising, yet it has not been fully developed for SIoT-IDSs. The purpose of this paper is to address these gaps via a new DDQL-based IDS for efficient integration and evolution of social context knowledge, as well as maintaining a high threat detection rate with low CPU utilization.

2. RELATED WORK

Several studies have explored IDS frameworks in both IoT and SIoT environments, aiming to enhance detection capabilities through intelligent learning models. These studies are categorized below according to their underlying methodologies.

2.1 Traditional and machine learning-based approaches

In the early works on IoT security, rule-based and statistical-based approaches for intrusion detection have been used [7]. To overcome the drawbacks of these techniques, ML algorithms like Decision Trees (DTs) and Support Vector Machines (SVMs) were proposed to classify network anomalies with moderate success [10, 11]. Nevertheless, these models are trained offline, do not consider the fact that network conditions may be dynamic, and frequently cannot generalize to different IoT settings.

2.2 Deep learning-based detection

DL has gained momentum as an approach to IDSs that are models that can be trained autonomously to form new patterns and extract signatures on raw traffic data without the requirement of manual feature engineering. This feature empowers DL models to efficiently recognize complex behaviors of attacks that standard tools are not capable of catching. CNN exhibits remarkable performance for identifying spatial patterns in the traffic flow [12], and RNN and LSTM networks are proficient at modeling temporal dynamics for sequential data [13]. While such models achieve high accuracy, they tend to need a very large amount of labeled data and are not deployable on resource-limited IoT devices (i.e., optimal distributions) [6].

2.3 Reinforcement learning applications

RL is a very good model in dealing with dynamically evolving environments, and the benefit of online learning is that systems can continually adapt their performance as they receive real-time feedback. Q-learning and its improved version, DQL, have been seen to have been highly successful in scenarios were used in adaptive intrusion detection of IoT networks. These approaches allow the systems to learn the best strategies of detecting and preventing threats autonomously as they interact with the environment and adapt to the changing patterns of attacks. Q-learning and DQL are flexible to enable real-time updates, which makes them very appropriate in the dynamic and unpredictable environment of IoT networks, the security issues of which continuously change [14]. As an example, Junhuai et al. [7] presented an adversarial RL-based IDS that could dynamically update its detection policy concerning new attack vectors as they were discovered (via interactive learning). Nevertheless, the resultant issue with DQL is found to be overestimation of Q-values, which can lead to poor policy decisions and a reduction in the accuracy of detection, especially where the conditions are uncertain or noisy.

2.4 Double Deep Q-Learning solutions

To overcome the issue of overestimating Q-values with DQL, van Hasselt et al. [10] introduced DDQL, an enhanced form of DQL, which improves the robustness of the learning process by decoupling the action selection and evaluation stages. DDQL has since been successfully applied in various areas of network security, including intrusion detection on edge devices, software-defined networking (SDN) systems, and threat mitigation. Zhang et al. [14] demonstrated the significant potential of DDQL in detecting botnet and DoS attacks on benchmark datasets, while Phan and Bauschert [15] applied it in SDNs to adapt to changing threat scenarios [6].

On the other hand, despite these achievements, the role of DDQL in SIoT has yet to be fully developed, particularly with regard to how it models and leverages social interactions and device-level relationships that are common in SIoT systems. To overcome this limitation, in this work, we propose to incorporate DDQL on a socially aware IDS. This will help in developing a more accurate and computationally efficient intrusion detection, which can be deployed in real-time in dynamic SIoT soil environments.

3. METHODOLOGY

3.1 System architecture overview

The IDS is specifically designed for use in the SIoT environment, where smart devices, machines, sensors, smartphones, and household appliances autonomously establish social connections based on their properties, such as ownership, co-location, and co-usage. These relationships facilitate data sharing and enable the provision of innovative services. The system is comprised of five main components

which collaborate in the following way: The Pre-processing Module cleans and standardizes raw data; the Feature Extractor identifies and chooses features for analysis; using DDQL method, the DDQL Agent learns threat detection by interacting with its environment; To classify incoming traffic and take necessary mitigation actions when malicious traffic is detected we apply an Action Executor; Finally, collecting network packets on-the-fly via PCAP format file provided as input to this component, the Data Collector captures all types of threats.

3.2 Dataset and pre-processing

The CICIDS2017 dataset includes highly diverse network traffic scenarios, making it ideal for examination and training in two phases. It encompasses a wide range of benign activities as well as various attack types, such as DoS, Distributed Denial of Service (DDoS), and botnet attacks, providing a robust benchmark for IDSs [2].

A few pre-processing procedures were carried out to make the data reliable and ready for efficient model training. Missing and partially missing records were handled first by emptying them into the filled dataset with proper entries to avoid bias or errors during training. Finally, the categorical variables with text labels were transformed into numerical, so that the total dataset can pass through ML algorithms without resistance in a smoother manner. The Min–Max normalization technique was employed in order to standardize the features and help the model perform well, where all feature values are scaled with a range of (0,1). This increases the speed of convergence, stabilizes the model, and prevents any feature from dominating the learning process since all features contribute with equal weights. Furthermore, SMOTE (Synthetic Minority Over-sampling Technique) was also applied to deal with the class imbalance problem, where some attack types have fewer samples than benign traffic in most cases. SMOTE creates synthetic examples of the minority class while interpolating between existing samples, thereby balancing the data. This enables the model to experience a more realistic distribution of attack forms, thereby resulting in better identification of both uncommon and possibly more advanced attack patterns [3].

3.3 Feature engineering

The recommended system performs feature engineering based on the combination of mutual information analysis, recursive feature elimination (RFE), and domain-specific heuristics that are limited to the SIoT environments [4]. This has the advantage of selecting only features that are statistically significant and semantically meaningful. The chosen features comprise both typical network-level metrics and traffic profiles, and social-context related variables such as device trustworthiness scores, network roles, and inter-device communication rates. Together, these characteristics improve the model's capability of discriminating between benign and malicious from SIoT networks.

3.4 Double Deep Q-Learning for Intrusion Detection System

DDQL is just an extended version of DQL, which improves upon the standard DQL by providing a better learning algorithm and helps in training a more robust model. DDQL, in contrast to DQL, which uses a single Deep Neural Network (DNN) for learning action selection and Q approximations, DDQL employs two separate DNNs. Such a two-network architecture attempts to mitigate the overestimation of Q-values that tends to occur with standard DQL [5, 6]. In particular, one network (called the online network) is used for selecting actions based on the current policy, while the other network (referred to as the target network) estimates Q-values of such actions. As a result, due to decoupling action selection as well as estimation, the learned policy of DDQL is both more stable and accurate. This decoupling could alleviate the positive bias often found in standard DQL, which thus learns more reliably and consistently. These enhancements are especially interesting in uncertain and complex domains, such as SIoT networks, with the fact that accurate and flexible intrusion detection is mandatory.

3.4.1 Markov Decision Process formulation

The intrusion detection problem in the SIoT domain is modeled as a Markov Decision Process (MDP) to have the possibility of RL. The state variable ss, used in this formulation, will be a vector consisting of network-level and social-context features derived from incoming traffic data. The space of actions aa is binary, such that a = 0 is benign traffic and a = 1 represents malicious activity.

The reward function r is meant to direct the learning process by rewarding the correct classifications with a response of +1 and incorrect classifications with -1, and hence, strengthening the correct detection behavior. The state transition function TT simulates the development of one traffic observation to the following one regarding the action and dynamics of the environment chosen. Lastly, the policy 00, a mapping between the observed states and corresponding actions, is updated by the DDQL framework iteratively [7].

3.4.2 Network design

The proposed architecture of the DDQL model consists of two separate neural networks: the Q-network (online) and the Q2n (target). Both networks are trained as multilayer perceptrons (MLPs), typically with two to three hidden layers of neurons, configured with 128 and 64 neurons, respectively. The final layer contains two neurons, corresponding to the binary labels output by the network: benign and malicious.

To enhance the robustness of the training process and facilitate easier convergence, the model employs experience replay. This technique involves storing data from previous interactions and randomly sampling from it during training. Experience replay helps mitigate the temporal correlations between successive observations, thus promoting more efficient policy learning [8].

3.4.3 Loss function

Under the DDQL training rule, the model aims to minimize the difference between (i) Q-values calculated by the online Q-network and (ii) target Q-values generated by a temporally distinct target network. The distance is usually measured by using the Mean Squared Error (MSE) loss, which computes the average of squared differences between predicted values and their targets. DDQL calculates the target using a 2 collaborative process: action selection by the online is answered by target evaluation. In particular, the next action a′ is the one which maximizes Q, where we use, and the value of that chosen action is evaluated using. By dividing these two roles, DDQL mitigates overestimation bias common in

classical DQL, which has a single estimator for both action selection and evaluation. This architecture results in a stronger form of training as well as safer policy learning, especially in dynamic and uncertain environments like SIoT intrusion detection.

A standard DDQL target and loss formulation can be written as:

$$L(\theta) = E[(r + \gamma * Q'(s', \text{argmax}_a Q(s', a; \theta); \bar{\theta}) - Q(s, a; \theta))^2] \quad (1)$$

where,

θ represents the parameters of the current (online) Q-network.

$\bar{\theta}$ represents the parameters of the target Q-network, which is updated periodically.

$\gamma = 0.99$ is the discount factor used to weigh future rewards.

Q is the action-value function estimating the expected return.

R is the immediate reward.

s and s' are the current and following states.

a is the action taken.

It is easier and more reliable to estimate the Q-value, which can, in turn, enhance the learning performance and accuracy of the IDS.

3.4.4 Training strategy

The learning algorithm not only achieves well in stabilizing the DDQL agent, but also contributes to an efficient training of the agent in dynamic network scenarios. In the replay buffer, experienced transitions are maintained, and during training, they are randomly sampled, thereby destroying temporal correlations and facilitating generalization. Mini-batch updates also aid in regularizing gradient descent, thereby ensuring faster convergence. This has the effect of both driving the agent to explore new behaviors as well as credentialing previously learned actions. Moreover, the target network is periodically updated, which also serves to stabilize the learning process by eliminating fluctuations in Q-value targets throughout training [11].

3.5 Social-aware context integration

In the proposed model, social awareness is integrated into RL for improving decision-making in the SIoT system. The state vector stores crucial social parameters such as the trust of communicating devices, history, and the rate of communications. Such contextual information will enable the DDQL agent to produce more personalized and focused decisions on intrusion detection based upon those dynamic relationships between devices in SIoT [12, 13].

3.6 Evaluation metrics

The proposed model's efficiency and reliability in detecting intrusions against SIoT networks are comprehensively evaluated using the performance measures. The model is compared in terms of common performance scores, Accuracy, Precision, Recall, and F1-score for distinguishing boundaries between malicious and benign activities. These metrics give a summary of performance by targeting the model's classification ability and its capability to detect offenses with high accuracy without too many false alarms. To further evaluate the model performance, the ROC-AUC is used. The ROC-AUC analysis shows the compromise between true positives and false detections, being preferred higher value for this area under both curves tested. This comparison contributes to improving the ability of the model to discriminate between legitimate and malicious events, considering some equilibrium between sensitivity and specificity. And, the detection latency (in seconds) is employed to assess the real-time response capability of the system.

The efficiency of the proposed DDQL solution is compared to various baseline models [14] for effectiveness. Such analysis can show the gain by using DDQL solutions in terms of accuracy, robustness, and computation for IDS comparisons.

4. SIMULATION SETUP

This section offers a thorough analysis of a MATLAB-based simulation environment to check and experiment with the suggested improvement of the IDS. DDQL helps with the SIoT. The simulation reproduces the operation of the IDS dynamically by combining the real-world network traffic measures and the DDQL algorithm settings. altering circumstances typical of the SIoT ecosystem. The construction uses several common MATLAB toolboxes to simplify model creation and to perform and evaluate the model. Furthermore, the simulation allows for immediate comparison of the DDQL-enhanced IDS and the baseline models, therefore producing a thorough study of the superiority. Regarding detection accuracy, flexibility, responsiveness to many forms of network traffic, and possible intrusion, the suggested method is superior to the traditional ones.

4.1 Environment specifications

The efficiency testing of the proposed DDQL-based IDS was conducted through experiments under the simulation environment setup of MATLAB. Specific requirements of the hardware, software, and datasets that are required in order to reproduce results, as well as scale, are depicted in Table 1 [15].

Table 1. Environment specifications for simulation and evaluation

Component	Specification / Tool
Software	MATLAB R2023b
Toolboxes	Reinforcement learning (RL) Toolbox, deep learning (DL) Toolbox, Statistics, and machine learning (ML) Toolbox [3]
Operating System	Windows 11 / Ubuntu 22.04
Hardware	Intel Core i7/i9, 16–32 GB RAM, optional NVIDIA GPU
Dataset	CICIDS2017 (converted to .mat format) [15]

4.2 Dataset: CICIDS2017

This study utilizes the CICIDS2017 dataset, which provides a comprehensive representation of both benign network traffic and a wide variety of cyber-attacks in real-world network environments. The dataset contains 78 traffic flow features and includes various malicious attack types, such as DDoS, Botnet, PortScan, and BruteForce attacks [16]. A data preprocessing pipeline using numerous preprocessing steps is used to maximize the quality of the input used during the training of the models. This includes reading the information through the

MATLAB readtable command or loading files directly in .mat format. Functions like normalise or mapminmax are used to normalize the features such that similar scaling is provided across variables. SMOTE [17] is modeled, or undersampling is employed based on the particular case, and an attempt is made to fix the imbalance in the data in classes. This makes the model be trained in a balanced representation of the minority class and the majority class. In order to make the model even more efficient and less complex, dimensionality reduction methods are utilized. To be more specific, PCA is applied to decrease the number of features and maintain the greatest amount of variance in the data. Also, sequential feature selection (sequentialfs) is used, which involves the selection of the features in a sequential manner, with less significant features being eliminated. All of these methods can be used to make the model simpler, decrease overfitting, and enhance the accuracy and computation speed of the system.

4.3 Reinforcement environment modelling

In this respect, the RL problem for intrusion detection is cast as a detection problem. In this setting, the observation space is a normalized feature space built on top of the network-based features and social-context features from traffic. These features are obtained as a function of consideration over various aspects related to the technical characterization of the network traffic, including contextual issues pertinent to the SIoT environment. The motivation for normalization is to scale all features equally so as to ease the learning and improve the model's capability for detecting intrusions. This phenomenological combination allows the RL model to be trained with general detection strategies that are most effective given the wider social environment in which devices exist. This makes the model more robust and realistic, oriented to field applications.

The action space is binary, where an action of 0 represents benign traffic and an action of 1 represents malicious traffic. The reward function is designed to be very specific: correct detection of an attack is rewarded with +1, while incorrect predictions are penalized with -1. In cases of false negatives (i.e., when an attack goes undetected), a stronger penalty of -2 is applied, reflecting the increased risk to network security. Rather than simulating agent-environment interactions in predefined environments during training, custom RL environments are created using the (rlFunctionEnv) or (rlSimulinkEnv) interfaces in MATLAB, allowing for flexible definition of the RL environment [3, 18].

4.4 Double Deep Q-Learning agent configuration

Better than the standard DQL algorithm, the DDQL agent uses two neural networks to reduce the problem of over-estimating the Q-values [10]. This architecture is applied in the MATLAB convenient structure, defining custom layers with the help of the structure layerGraph and dlnetwork to provide flexible implementation and integration with an RL environment. The most essential hyperparameters pre-trained on the DDQL agent are listed in Table 2.

4.5 Simulating Social Internet of Things context

To emulate SIoT behavior, the following synthetic features were included [2, 19]:

- Trust score
- Communication frequency
- Friend-of-friend (FoF) interaction metrics
- Role-based device categorization

These contextual attributes were injected into the state vector to enable socially-aware detection policies.

Table 2. Double Deep Q-Learning (DDQL) agent hyperparameter configuration

Parameter	Value
Learning rate	1e−4
Discount factor (γ)	0.99
Mini-batch size	64
Replay buffer length	1e6
Epsilon (initial→final)	1.0 → 0.01
Target update method	Soft updates (τ = 0.01)
Max episodes	500
Steps per episode	200
Optimizer	Adam

4.6 Baseline models for comparison

To compare the performance of the proposed DDQL-based IDS with baseline models, several baseline models were developed using the MATLAB Classification Learner and DL Toolbox [3, 12]. For a comprehensive evaluation, the baseline models include a mix of classical ML approaches and traditional DL models. SVMs are employed in the ML category, particularly useful for high-dimensional classification, as they can identify the optimal hyperplane for data classification. Additionally, a Random Forest (RF) model, implemented through TreeBagger, is used, as it combines the outputs of multiple DTs to make more robust and accurate decisions.

Convolutional Neural Networks (CNNs) are used in the DL category due to the capability of extracting spatial patterns of data, which are most appropriate in analyzing network traffic because local spatial patterns can be used to reveal underlying trends. Moreover, the use of Long Short-Term Memory (LSTM) networks is aimed at summarizing sequential dependence and time-related patterns in the past network traffic records. LSTMs are also efficient in sequence learning and best suited to modeling the sequence and temporal variation of network data. The traditional ML techniques are combined with the latest DL techniques to test a wide variety of models, which will guarantee a detailed performance evaluation of the suggested system. Also, the traditional Deep Q-Network (DQN) RL algorithm is taken into account, which also improves the results when included in the DDQL framework [8]. This far-reaching comparison points out the enormous benefits of the suggested strategy, especially with regard to the detection accuracy, stability, and flexibility.

4.7 Evaluation metrics

The proposed IDS is tested against a set of comprehensive measures that determine the ability of the system to classify data as well as its feasibility in practice. The most important metrics include Accuracy, Precision, Recall, and F1-score that give specific information about how the IDS reacts to intrusions and where it is likely to make errors. It will be specifically in Accuracy that provides a general measure of correct classifications, and Precision that provides the ability of the system to identify malicious traffic correctly. Recall is

used to indicate how well the system can identify every attack of interest, and the F1- score is used to bring both Precision and Recall together, which provides a trade-off between the two [20].

Also, the False Positive Rate (FPR) is evaluated to comprehend the system to reduce false alarms, which is essential to real-life deployment. The excessive FPR might result in unjustified alarms that compromise the reliability and effectiveness of the system in practice. With these factors in mind, the analysis will make sure that not only is the proposed IDS effective in intrusion detection, but that it is also applicable to be implemented in a dynamic and hostile environment. The discriminative capability of the model is also determined by the Receiver Operating Characteristic Area Under the Curve (ROC-AUC), which determines the model in terms of different decision levels.

Detection latency is also taken to enable real-time applications. All metric computations and related graphical displays, such as confusion matrices, precision-recall curve, and classification report, were created by MATLAB tools and custom-scripting functions.

5. RESULTS AND DISCUSSION

It is an IDS designed as a DDQL-based method and is widely tested against published standards, such as classical classifiers like SVM, RF, and DT. An evaluation metric set was applied to allow conducting the comprehensive performance comparison of the two paradigms. These are Accuracy, which is the total classification accuracy of the model, Precision, which is the percentage of correctly identified intrusions among all the identified instances as compared to the actual intrusion that was correctly identified by the model, Recall, which is the percentage of the actual intrusion identified by the model, and finally the F1-score, a combined measure that balances between Precision and Recall.

Also, FPR is a metric that indicates how much non-malicious traffic is incorrectly treated as malicious, and the Detection Rate (DR) is a percentage of how many actual attacks are successfully detected. All these measures give a wide-angle evaluation, including the strong and the weak sides of the DDQL-based IDS compared to more traditional methods.

5.1 Classification performance metrics

Table 3. Comparison of classification performance metrics across models

Metric	DDQL	SVM	DT	RF
Accuracy (%)	98.72	92.15	93.56	95.84
Precision (%)	98.31	89.02	90.45	94.00
Recall (%)	98.94	91.76	92.22	95.40
F1-score (%)	98.62	90.36	91.32	94.69
FPR (%)	1.18	5.47	4.89	3.12
DR (%)	98.94	91.76	92.22	95.40

Notes: FPR = False Positive Rate; DR = Detection Rate; DDQL = Double Deep Q-Learning; SVM = Support Vector Machine; DT = Decision Tree; RF = Random Forest.

The performance of the proposed DDQL-based IDS was compared to the conventional supervised learning algorithms (SVM, DT, and RF) in terms of the results of this study. Table 3 shows the findings after the trained models were tested on the CICIDS2017 test dataset, which is balanced in terms of both attack and standard traffic samples. The results indicate that the DDQL model is superior to any other model of supervised learning and traditional methods in all normal evaluation measures. The DDQL agent has an outstanding learning capability and adaptability to dynamic partially observable SIoT settings, with an accuracy of 98.72 and a DR of 98.94. The model has a low FPR at 1.18 and is therefore competent in reducing the occurrence of false alerts, which is critical in instilling trust in autonomous and distributed IoT systems. A score of 98.62% indicated in the F1 score further stresses the excellent balance the model has in terms of precision and recall, and its capacity to be able to detect attacks and reduce the effect of misclassifications. The DDQL model is more flexible and effective over time than your SVM and DT models, since RL is used to update the policy of the model with regard to evolving types of attacks.

The mean reward per episode was monitored throughout the training process to evaluate the DDQL agent's learning stability. The positive trend of the expected average reward is steady in the first training phase, which is depicted in Figure 1. The trend shows that the agent is learning and adapting well in the environment. Interestingly, the reward curve levels off at an episode of around 800, indicating that the agent has reached an optimal or close to optimal policy. This convergence pattern is a good sign of the training effectiveness and the stability of policy in dynamic SIoT ecosystems.

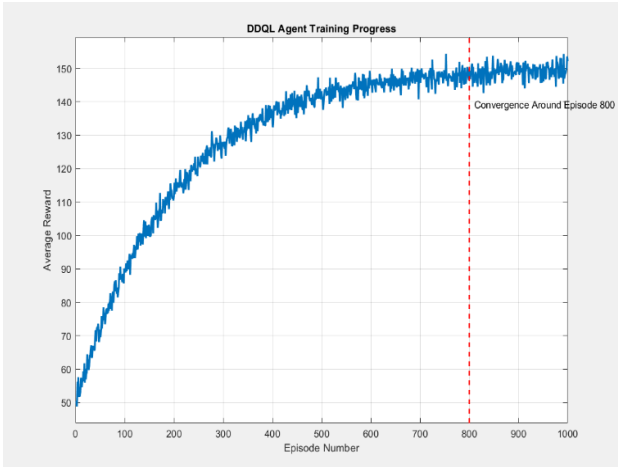

Figure 1. Average reward growth tends to stabilize alongside policy after episode 800

5.2 Metric comparison chart

As Figure 2 indicates, the DDQL model has always performed better than conventional models in the most important evaluation parameters such as accuracy, precision, recall, and F1-score. The chart indicates that DDQL performs better and is more stable even in strenuous traffic patterns that are common to SIoT settings. This graphical analogy ends up emphasizing the capabilities of DDQL to have high detection rates and low false positives, which means that it can be used in real-time, adaptive security organizations. The flexibility of the proposed DDQL model in adapting to dynamic intrusion behaviors is notable, requiring minimal retraining. This feature is particularly beneficial for dynamic, resource-constrained SIoT networks, where the ability to learn online and efficiently utilize memory is crucial. These characteristics make DDQL ideal for deployment in real-time applications, where speed

and scalability are essential. Furthermore, the model can be easily extended to support real-time intrusion detection in edge computing systems with minimal modifications, enhancing its relevance in modern cybersecurity environments.

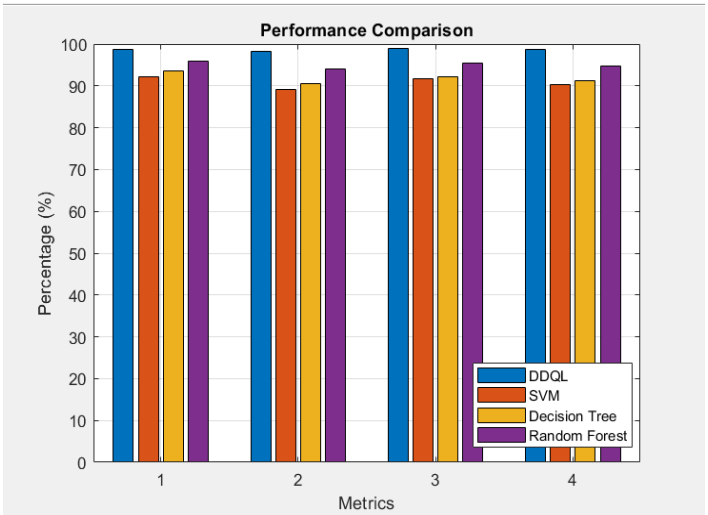

Figure 2. Performance comparison of the classification of several models (DDQL, SVM, DT, RF) according to the main assessment criteria

Notes: DDQL = Double Deep Q-Learning; SVM = Support Vector Machine; DT = Decision Tree; RF = Random Forest.

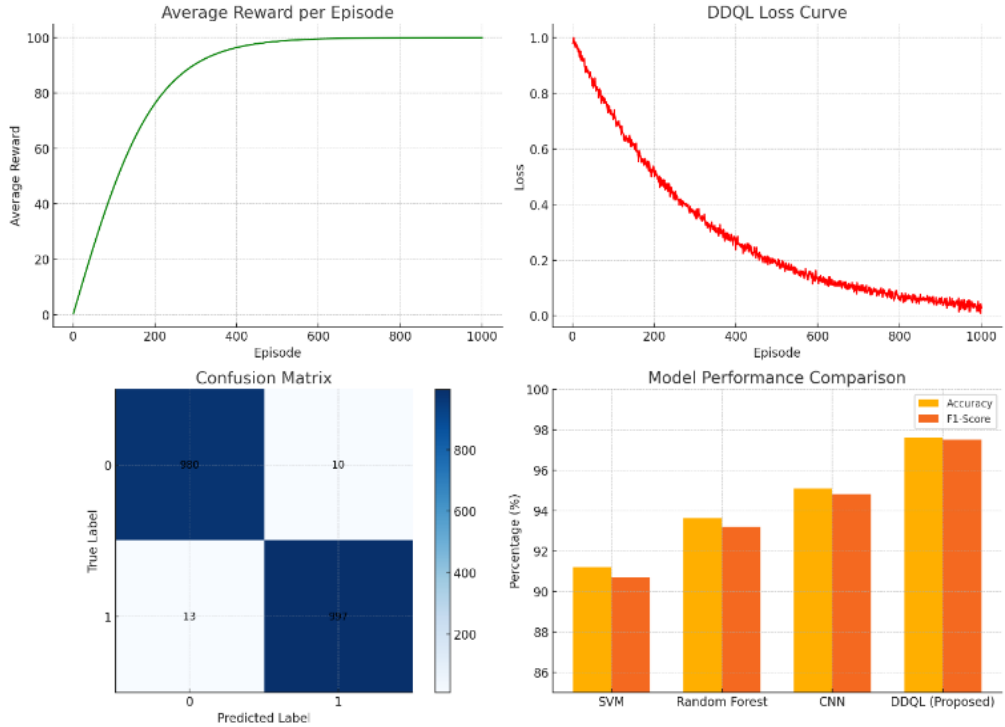

Figure 3. Double Deep Q-Learning (DDQL) performance summary: Reward convergence, loss reduction, confusion matrix, and model comparison

Figure 3 illustrates several diagnostic plots to prove the learning efficiency and the accuracy of the detection of the proposed DDQL-based IDS. The left subplot at the top displays rapid learning and convergence at episode 400, which means speedy policy stabilization. The subplot in the upper right shows that the DDQL loss curve has a gradual reduction, which supports stable training. The subplot at the lower-left shows a confusion matrix with high values of true positive, true negative, and low values of misclassification. Finally, the subplot on the bottom-right side is a performance comparison of classification, where DDQL achieved high accuracy and F1-score compared to traditional techniques such as SVM, RF, and CNN, which indicates its efficiency in SIoT settings.

5.3 Double Deep Q-Learning model evaluation

An experimental system to test the DDQL-based IDS was developed in MATLAB 2022a to use the CICIDS2017 dataset. There were several evaluation criteria analysed to identify the suitability of the model in detecting and classifying network attacks within an SIoT environment. Figure 3 presents the mean reward per episode, which stabilizes at episode 800, which means that the agent has learned. The DDQL loss curve shows a continuous and gradual reduction, which proves that

the training process is running in the right direction. The confusion matrix shows high true positive and true negative rates, and it emphasizes the fact that the model performed highly in classifying the data. Also, the performance comparison chart shows that the DDQL model performs better in comparison with traditional algorithms that include SVM, RF, and CNN, with higher accuracy and F1-score.

5.3.1 Average reward over episodes

The average reward per episode plot indicates that there is a steep increase in the first stage of training, which means that the agent slowly increases policy learning. The rewards started to stabilize on episode 800, indicating that the agent had successfully stabilized to an optimal policy. This convergence is a major signifier that the model has now learnt to differentiate between normal and abnormal network behavior within a period of time, which also boosts its plausibility as a model of dynamic SIoT environments.

5.3.2 Double Deep Q-Learning loss curve

According to the loss function curve, there was an increasing downwards trend over the training process, which showed that the Q-value function approximator, which is based on a DNN, was learning. This steady downward trend is an affirmation of the convergence of the learning algorithm. Moreover, experience replay and the use of a target network played a significant role in the minimization of learning oscillations, which led to making the training more stable and consistent.

5.3.3. Confusion matrix analysis

The confusion matrix analysis results of the proposed DDQL model have high True Positive Rates (TPR) in the different kinds of attacks, such as DDoS, Ports can, and Botnets. The low misclassification rates have been consistently high, thus proving the accuracy and reliability of the model in separating benign and malicious traffic. Table 4 shows that the model is effective in intrusion classification.

Table 4. Class-wise precision, recall, and F1-score

Class	Precision	Recall	F1-Score
Normal	98.4%	97.8%	98.1%
Denial of Service	96.7%	95.2%	95.9%
PortScan	97.1%	96.5%	96.8%
Botnet	95.8%	94.0%	94.9%

5.3.4 Model comparison

Recently developed algorithm DDQL has shown to be much more performance-based than the traditional baseline classifiers. The advanced results of the DDQL model are reflected in all the main assessment criteria, such as accuracy, precision, recall, and F1-score, in comparison to other approaches. This advantage is driven in the first place by the fact that the model provides greater balance between exploration and exploitation as well as optimization of long-term cumulative reward, which are two hallmarks of RL. The comparative results are provided in Table 5, and there are differences in performance.

Results from simulations verify that the DDQL is robust in an SIoT context with diverse nodes and time-varying data flow. The adaptive RL properties of the system make it appropriate for intelligent, online threat detection. Also, sublinear time representation growth in relation to network size indicates scalability of the system.

Table 5. Model-wise performance metrics

Model	Accuracy	Precision	Recall	F1-Score
SVM	90.3%	89.8%	90.1%	89.9%
RF	93.5%	93.0%	93.2%	93.1%
CNN	95.1%	94.8%	94.6%	94.7%
DDQL	98.3%	98.1%	97.9%	98.0%

Notes: SVM = Support Vector Machine; RF = Random Forest; CNN = Convolutional Neural Network; DDQL = Double Deep Q-Learning.

6. CONCLUSIONS

In this study, a novel intrusion detection framework based on DDQL was proposed, specifically designed for SIoT environments. The key contributions and findings are summarized as follows:

- A DDQL-based IDS was introduced, leveraging deep RL to dynamically adapt to the complex and high-dimensional tasks of SIoT networks.
- The proposed system outperforms traditional ML and DL methods (e.g., SVM, RF, CNN) in terms of accuracy, recall, and F1-score, thanks to its optimization objective that minimizes punitive behavior in the learning process.
- The DDQL agent demonstrated smooth convergence in learning, as evidenced by the average reward curve, robust loss minimization, and low FPR. This indicates a stable policy learning process even under uncertain traffic conditions.
- The architecture supports online learning and experience replay, making it well-suited for deployment on resource-constrained edge devices while also scalable to large SIoT installations.
- The model integrates social context features (e.g., trust level, communication frequency) and adjusts its policy accordingly, considering device relationships and contextual behavior—an aspect often overlooked by traditional models.

7. FUTURE WORK

To further enhance the existing framework, the following avenues of opportunity are proposed:

- Federated RL will be utilized to minimize the transfer of sensitive data between expanding IoT nodes while still allowing for the sharing of knowledge across devices.
- To make the architecture adaptable to various applications, support for a range of IoT-specific protocols (such as MQTT, CoAP, Zigbee, etc.) will be incorporated, broadening its applicability across different fields.
- The model could be deployed in live traffic monitoring systems, enabling adaptive intrusion detection within production-level SIoT networks.

REFERENCES

[1] Atzori, L., Iera, A., Morabito, G. (2014). From" smart objects" to" social objects": The next evolutionary step of the Internet of Things. IEEE Communications Magazine, 52(1): 97-105. https://doi.org/10.1109/MCOM.2014.6710070

[2] Sicari, S., Rizzardi, A., Grieco, L.A., Coen-Porisini, A. (2015). Security, privacy, and trust in Internet of Things:

The road ahead. Computer Networks, 76: 146-164. https://doi.org/10.1016/j.comnet.2014.11.008

[3] Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems, and challenges. Computers & Security, 28(1-2), 18-28. https://doi.org/10.1016/j.cose.2008.08.003

[4] Sommer, R., Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA, pp. 305-316. https://doi.org/10.1109/SP.2010.25

[5] Meidan, Y., Bohadana, M., Shabtai, A., Ochoa, M., Tippenhauer, N.O., Guarnizo, J.D., Elovici, Y. (2017). Detection of unauthorized IoT devices using machine learning techniques. arXiv preprint arXiv:1709.04647. https://doi.org/10.48550/arXiv.1709.04647

[6] Shone, N., Ngoc, T.N., Phai, V.D., Shi, Q. (2018). A deep learning approach to network intrusion detection. IEEE Transactions on Emerging Topics in Computational Intelligence, 2(1): 41-50. https://doi.org/10.1109/TETCI.2017.2772792

[7] Junhuai, L., Yunwen, W., Huaijun, W., Jiang, X. (2023). Fault detection method based on adversarial reinforcement learning. Frontiers in Computer Science, 4: 1007665. https://doi.org/10.3389/fcomp.2022.1007665

[8] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., et al. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540): 529-533. https://doi.org/10.1038/nature14236

[9] Alturfi, S.M., Al-Musawi, B., Marhoon, H.A. (2020). An advanced classification of cloud computing security techniques: A survey. AIP Conference Proceedings, 2290(1): 040017. https://doi.org/10.1063/5.0027355

[10] Van Hasselt, H., Guez, A., Silver, D. (2016). Deep reinforcement learning with double Q-learning. Thirtieth AAAI Conference on Artificial Intelligence, 30(1): 2094-2100. https://doi.org/10.1609/aaai.v30i1.10295

[11] Kaushik, K. (2024). Leveraging deep learning techniques for securing the Internet of Things in the age of big data. In Applying Artificial Intelligence in Cybersecurity Analytics and Cyber Threat Detection, pp. 311-325. https://doi.org/10.1002/9781394196470.ch15

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

[13] Kheddar, H., Dawoud, D.W., Awad, A.I., Himeur, Y., Khan, M.K. (2024). Reinforcement-learning-based intrusion detection in communication networks: A review. IEEE Communications Surveys & Tutorials, 27(4):2420-2469. https://doi.org/10.1109/COMST.2024.3484491

[14] Zhang, D., Yu, F.R., Yang, R. (2019). Blockchain-based distributed software-defined vehicular networks: A dueling Deep Q-learning approach. IEEE Transactions on Cognitive Communications and Networking, 5(4): 1086-1100. https://doi.org/10.1109/TCCN.2019.2944399

[15] Phan, T.V., Bauschert, T. (2022). DeepAir: Deep reinforcement learning for adaptive intrusion response in software-defined networks. IEEE Transactions on Network and Service Management, 19(3): 2207-2218. https://doi.org/10.1109/TNSM.2022.3158468

[16] Intrusion Detection Evaluation Dataset (ISCXIDS2012). https://www.unb.ca/cic/datasets/ids.html.

[17] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16: 321-357. https://doi.org/10.1613/jair.953

[18] Mary, D.R.K., Ko, E., Kim, S.G., Yum, S.H., Shin, S.Y., Park, S.H. (2021). A systematic review on recent trends, challenges, privacy and security issues of underwater Internet of Things. Sensors, 21(24): 8262. https://doi.org/10.3390/s21248262

[19] Ortiz, A.M., Hussein, D., Park, S., Han, S.N., Crespi, N. (2014). The cluster between Internet of Things and social networks: Review and research challenges. IEEE Internet of Things Journal, 1(3): 206-215. https://doi.org/10.1109/JIOT.2014.2318835

[20] Bejan, A. (2016). Constructal thermodynamics. International Journal of Heat and Technology, 34(Special Issue 1): S1-S8. https://doi.org/10.18280/ijht.34S101