International Journal of Safety and Security Engineering
Vol. 16, No. 5, May, 2026, pp. 1051-1058
Journal homepage: http://iieta.org/journals/ijsse

Botnet Detection in Internet of Things Environments Using Double Deep Q-Networks

Nibras Talib Mohammed[1*], Sabah M. Alturfi[2], Farooq Khudhair Abbas Jaber Al-chalab[3], Fatima Faydhe Al-Azzawi[4]

[1] College of Computer Science and Information Technology, University of Kerbala, Kerbala 56001, Iraq
[2] College of Law, University of Kerbala, Kerbala 56001, Iraq
[3] College of Administration and Economics, University of Kerbala, Kerbala 56001, Iraq
[4] Institute of Technology, Middle Technical University (MTU), Baghdad 10074, Iraq

Corresponding Author Email: nbrass.t@uokerbala.edu.iq

https://doi.org/10.18280/ijsse.160511

Received: 22 October 2025
Revised: 12 May 2026
Accepted: 24 May 2026
Available online: 31 May 2026

Keywords:
Internet of Things, security, botnet detection, reinforcement learning, Double Deep Q-Networks, anomaly detection, cybersecurity

ABSTRACT

The increasing use of Internet of Things (IoT) solutions has significantly exposed connected systems to various cyber threats. Botnet attacks, which exploit security vulnerabilities to compromise and control large numbers of IoT devices, are among the most severe threats facing modern connected environments. Conventional detection mechanisms are often ineffective in addressing the evolving, heterogeneous, and dynamic nature of IoT traffic, especially when attackers modify their behavior or use stealthy communication patterns. This paper presents a botnet detection framework based on Double Deep Q-Networks (DDQN), a reinforcement learning approach that supports adaptive decision-making by learning from network activity patterns. The suggested DDQN agent will be trained on the basis of traffic-related features reflecting the behavior of devices. and network interactions. By being constantly engaged with the simulated IoT environment, the agent learns to differentiate between good and bad traffic and minimizes false alarm. The framework makes use of as well. Reward-based feedback to enhance detection choices and evolve with new attack patterns. The simulated performance test using IoT traffic demonstrates that the proposed mechanism can identify malicious activity with high accuracy, precision, and responsiveness. The findings suggest that DDQN is a promising adaptive solution to detect botnet in IoT setting, especially in dynamic and changing attack conditions.

1. INTRODUCTION

The Internet of Things (IoT) is an integration of billions of sensors, devices, and control systems in different industries, which makes it possible to have smart homes, telemedicine and telehealth, industrial automation, and other data-driven services [1]. This broad and diverse network however poses gigantic security issues. The most severe threats include IoT botnets, which are networks of infected devices and which an attacker can use to launch distributed denial-of-service (DDoS) attacks, to steal data, or to place a backdoor [2, 3].

Conventional rule-based and signature-based botnet detectors do not work well in detecting botnets when attackers adapt their techniques or encrypted traffic, which is especially challenging in IoT devices with low processing power [4]. Data-driven anomaly detectors built with supervised or unsupervised machine-learning algorithms have improved resilience by learning "normal" traffic profiles and flagging deviations [5, 6]. Nonetheless, they continue to use large labeled databases and might not be able to react to changes in real-time as threats continue to change.

Reinforcement learning provides a radically new paradigm: an autonomous agent learns an effective defence strategy through repeated interactions with its environment and feedback on reward [7]. Deep Q-Networks (DQN) are based on classical Q-learning that uses deep neural encoders, allowing agents to reason through high-dimensional state spaces typical of network telemetry [4]. However, DQN is prone to optimistic bias in its action-value estimates, which can slow or destabilise learning. Double Deep Q-Networks (DDQN) address this issue by separating the action-selection and action-evaluation steps, thereby curbing overestimation and improving convergence stability [8].

Reinforcement learning and DDQN have begun to prove their worth in intrusion-detection and anomaly-detection research [7, 8]. Yet their specific application to IoT botnet defence remains only partially explored. To bridge this gap, the present study introduces a DDQN-based framework that learns, in real-time, to identify compromised IoT devices from live traffic features. The proposed agent continually adapts to shifting attack behaviors, delivering timely and reliable detection while maintaining low false-alarm rates.

The main contribution of this study is not the introduction of a new DDQN algorithm, but the development of an IoT-specific DDQN-based botnet detection framework with an explicit state-action-reward formulation. Unlike general DDQN-based intrusion detection studies, the proposed framework models IoT device behavior using traffic-level

features, applies a customized reward strategy that penalizes missed botnet devices more strongly than other errors, and evaluates the agent under controlled benign and malicious IoT traffic conditions. The research additionally offers a reproducible simulation environment, hyperparameter configuration, and a long-term performance study to understand how DDQN could be applied to botnet detection in changing IoT scenarios.

2. RELATED WORK

The discovery of botnets in IoT environments has been a popular area of research due to the proliferation of connected devices exposing devices to cyberattacks. The possible detection approaches to date can be classified into traditional and machine learning-based detectors, deep learning-based detectors, and reinforcement learning-based detectors.

2.1 Traditional and machine learning-based detection

Botnets were initially detected using rule-based or signature-based methods. Though these mechanisms are useful in detecting known threats, they are less efficient to new, concealed or emerging patterns of attacks in the dynamic IoT networks. To address these limitations, a number of studies have used machine learning methods in detection of intrusion and botnets. Suppose, Vitorino et al. [3] in comparison to classification algorithms IoT intrusion detection and demonstrated that models like Support Vector Machine (SVM) classifiers, and random forest classifiers can be used to achieve reasonable accuracy. Nevertheless, these models can still be limited with regard to their ability to adapt to new or emerging threats.

2.2 Deep learning-based detection

More intricate patterns of traffic have also been captured by deep learning methods. McDermott et al. [9] used convolutional neural networks (CNNs) to detect botnets in IoT systems and achieved increased accuracy, but their method relied on the existence of formal training data. Yin et al. [10] used recurrent neural networks to intrusion detection and demonstrated that sequence-based deep learning models are well-positioned to learn traffic patterns, compared to multiple traditional machine learning classifiers. other hybrid models, including CNNs with long short-term memory (LSTMs) networks [11], and frameworks with both supervised and unsupervised learning approaches and frameworks with both supervised and unsupervised learning approaches, have shown good detection capabilities. These models may however be computationally expensive and hard to implement on resource constrained IoT devices [12, 13].

2.3 Reinforcement learning-based detection

Researchers have been considering reinforcement learning as a way to enhance adaptability. Alavizadeh et al. [4] presented a DQN-based intrusion detection system, demonstrating that interaction with the environment can enhance the detection robustness. Nevertheless, standard DQN can be affected by the overestimation of the action-value, particularly when using complex traffic patterns. Hu et al. [8] employed DDQN as a refined reinforcement learning approach to address the limitations of conventional DQN. This method decouples action selection from action evaluation, thereby improving learning stability and reducing the overestimation bias commonly associated with Q-value estimation. Tellache et al. [14] also examined a multi-agent reinforcement learning-based intrusion detector system, demonstrating that reinforcement learning could be used to aid adaptive detection in dynamic network settings. Other IoT botnet detection research. Researches [15-17] also pointed to the possibilities of machine learning, deep learning, and hybrid learning to detect intrusion and botnets.

2.4 Research gap

Although past researches have made gains in terms of botnet and intrusion detection in IoT-based systems, there are still various obstacles. These are reliance on labeled datasets, low scalability, expensive computation, and low resistance to novel types of attacks. To answer this, the present study extends the DDQN framework and offers an adaptive and lightweight botnet detection approach. The main aspects of the design are experience replay, target network synchronization, and a custom-designed reward mechanism to minimize false classifications and provide a timely response in changing IoT network settings.

3. METHODOLOGY

3.1 Simulation environment and data generation

A simulated network was created with 100 simulated IoT devices over a common wireless network to mimic an IoT scenario. The simulated devices were typical IoT devices like smart cameras, environmental sensors, smart meters, and home automation devices. These devices were set up with 80 benign nodes and 20 compromised nodes, which made the ratio of malicious devices 20% malicious. Benign devices produced standard traffic patterns such as periodic sensing messages, small control packets and normal client-server traffic. The malicious traffic was injected by giving compromised nodes to carry out botnet-related actions, such as port scanning, spam propagation, and DDoS traffic bursts. At fixed intervals, traffic records were recorded and to capture statistical properties of a device, they were used to describe its behavior in the system state. Figure 1 demonstrates the detailed system architecture of the proposed DDQN-based botnet detection system in an Industrial Internet of Things (IIoT) environment. The framework will have the following key elements:

- IoT Devices: Smart devices generating network activity.
- Edge Gateway / Traffic Collector: Gathers traffic sent by industrial IoT devices, consolidates the flow logs, captures the observation, and transmits the information that has been extracted to the preprocessing and detection modules.
- Preprocessing Unit: Removes and normalizes important traffic features to form homogenous input to the model.
- Environment Model: Encodes the Markov Decision Process (MDP) with state, action, and reward structure.
- DDQN Agent: Trains a good classification policy through interaction with the environment.
- Decision Module: Uses policy knowledge to classify devices into normal and botnet-infected devices.
- Alert & Control Center: Alerts administrators about

identified threats and assists with isolating or marking suspicious devices to be inspected in more detail.

In a typical IIoT representative of the industrial environment, smart sensors, industrial cameras, programmable logic controllers (PLCs), and smart meters are linked to a local edge gateway, which is connected to a monitoring server. The suggested DDQN-based detector module is implemented on the gateway or edge server where it examines the characteristics of traffic in near real time. Once the behavior of botnets is identified, the alert system informs the control center and the suspicious device may be isolated or identified as a subject to the inspection. System-wise, the proposed framework is designed to be deployed at a gateway level, but not be executed on each IoT endpoint. Both the end devices and the edge gateway or monitoring server operate as usual in sensing and communicating, and the edge gateway or monitoring server does traffic collection, feature extraction, inference using DDQN, and alert generation. This deployment architecture relieves resource-limited IoT devices of some of their computational workload, and allows near-real-time monitoring of industrial IoT systems.

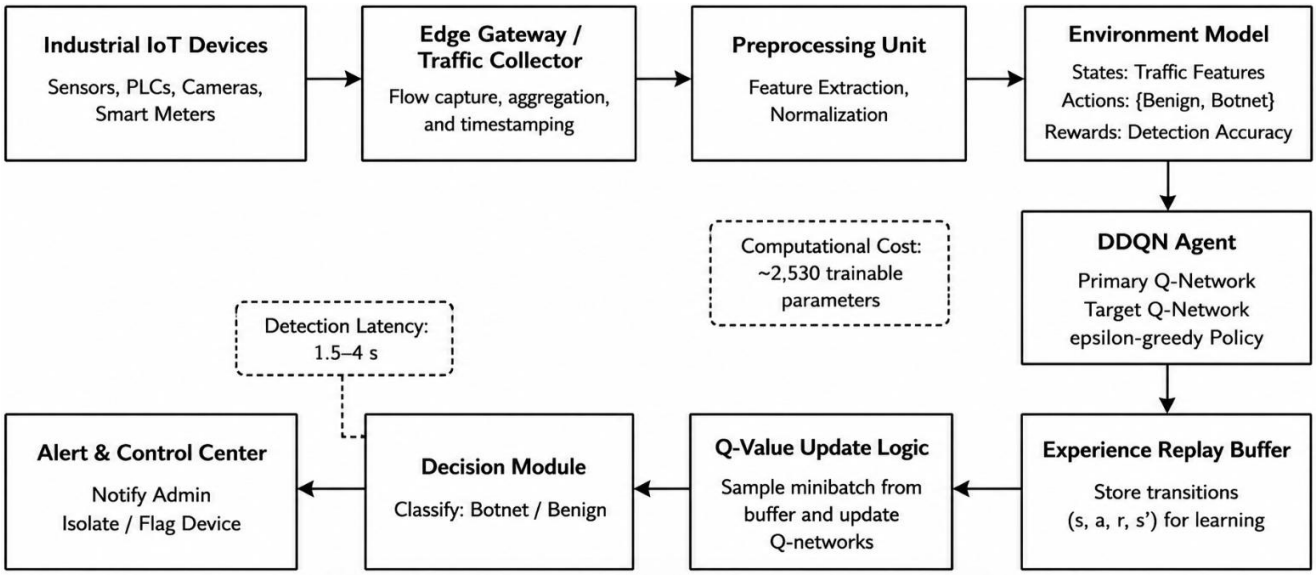

Figure 1. Detailed system architecture of the proposed Double Deep Q-Networks (DDQN)-based botnet detection system in an Industrial Internet of Things (IIoT) environment

3.2 State and action design

(1) State Feature Representation

The devices in the simulation are defined by feature vectors representing behavioral statistics, such as:

- Packet transmission rate
- Mean packet size
- Session duration
- Distribution of protocols used
- Count of unique destination IPs

All features are normalized and then fed to the learning model in order to guarantee consistency in the input and avoid scaling problems.

(2) Action Definitions

The space of actions of the agent is binary. In each decision point the agent chooses one of the following:

- 0 – Classify the device as benign
- 1 – Classify the device as part of a botnet

3.3 Double Deep Q-Network framework

(1) Reward Strategy

The rewarding mechanism is designed in such a way that it will encourage accurate detection and punish misclassification:

- True Positive (correct botnet detection): +1.0
- True Negative (correct benign detection): +0.5
- False Positive (benign marked as botnet): -1.0
- False Negative (missed botnet device): -2.0

This design focuses on reducing the undetected threats and has a low rate of false alarms.

(2) DDQN Model Architecture

The DDQN architecture is based on a deep neural network with the following structure:

- Input Layer: It corresponds to the dimensionality of the state feature vector.
- Two Hidden Layers: Fully connected layers with rectified linear unit (ReLU) activation functions.
- Output Layer: Two neurons that estimate action-values (Q-values) of each action.

A replay experience mechanism is used to stabilize learning by storing previous transitions, and a target network which is updated periodically is used to avoid divergence by decoupling the evaluation and selection processes [8, 18].

(3) Training Pipeline

The agent learns via episodic interactions with the environment:

- It checks the status and chooses an action through the epsilon-greedy exploration algorithm.
- The environment gives back the succeeding state and a reward.
- Experiences are stored in a replay buffer.
- Mini-batches are randomly sampled for training.
- The Q-network updates are performed using loss minimization in time difference.
- The target network is updated after each set time interval.
- Training is continued until performance measures (e.g. accuracy, loss) stabilize.

(4) Hyperparameter Settings

To improve reproducibility, the main hyperparameters used

for training the DDQN model are summarized in Table 1. These values were selected based on common reinforcement learning practice and preliminary tuning to obtain stable convergence while keeping the computational cost suitable for the simulated IoT environment. A small learning rate was used to avoid unstable neural network updates, while the replay buffer and mini-batch size were selected to provide sufficient sample diversity during training. The epsilon-greedy schedule was applied to balance exploration and exploitation, and the target network was updated periodically to reduce instability in Q-value estimation.

Table 1. Main hyperparameters used for Double Deep Q-Networks (DDQN) training

Hyperparameter	Value
State features	5 traffic features
Hidden layers	2 fully connected layers
Neurons in hidden layers	64 and 32
Activation function	Rectified linear unit (ReLU)
Optimizer	Adam
Learning rate	0.001
Discount factor	0.99
Batch size	64
Replay buffer size	10,000 transitions
Number of training episodes	500
Initial epsilon	1.0
Minimum epsilon	0.01
Epsilon decay rate	0.995
Target network update frequency	Every 10 episodes
Loss function	Mean squared temporal-difference error

3.4 Performance evaluation

After training, the DDQN model is validated on a separate set of traffic traces that include a range of botnet intensities and behavioral variations. The simulated traffic records were divided into non-overlapping training, validation, and testing subsets using a 70:15:15 split. The training subset was used to update the DDQN model, the validation subset was used to monitor convergence and tune training behavior, and the testing subset was used only for final performance reporting. To reduce the risk of overfitting, traffic records from the same simulated episode were kept within the same subset, and the final evaluation was performed on unseen traffic traces.

The performance metrics that are examined include:

• Overall detection accuracy.

• Precision, recall and F1-score, the latter being the harmonic mean of the former two.

• False negative and false positive. It is compared to benchmarking on conventional machine learning classifiers (SVM) and Random Forest) and conventional DQN-based models [3, 4].

4. RESULTS AND DISCUSSION

• Precision: DDQN model with mean precision of 94.5 was higher compared to SVM (87.3) and random forest (90.1) model.

• Precision and Recall: Achieved 92.7% and 95.8% respectively indicating good threat detection and minimal error.

• False Positive Rate: Under 4% which is vital in reducing false alarm.

• Training Convergence: Reached in 500 episodes, achieving faster learning than DQN, since action-value overestimation is smaller.

• Adaptability: The system exhibited a high adaptability with a maintained accuracy of over 90 percent in case of a dynamic attack, pointing to its robustness.

These findings suggest that the Double Deep Q- Networks (DDQN) model is better than the compared traditional machine learning models and the standard DQN. Modeling-wise, this is primarily connected with the learning process of DDQN. In contrast to regular DQN, DDQN also splits action choice and action evaluation, which decreases overestimation of Q-values and results in the more stable decisions. The experience replay mechanism also enables the agent to learn based on varied past states of traffic and not just the sequential observations. Moreover, the periodic target network update can be used to stabilize the training process by minimizing the quick change in value estimation. The reward scheme also helps the detection of botnets with the greater penalty to the false negatives, which induces the agent to detect malicious devices. These processes describe the reason the DDQN model was more accurate, recalled and stable than the compared baseline models.

In particularly dynamic industries, including cybersecurity, IoT networks, and autonomous systems, the conditions and threat patterns are expected to shift rather rapidly. Slow or incorrect responses may occur when static models or superficial learning are used in such contexts.

DDQN is more appropriate in accommodating more consistent learning as it minimises the overestimation bias that is present in the simplest DQN. This is because of its capacity to learn at all times and continually adjust to new behaviours; thus, it can maintain its performance despite adverse changes in the adversarial strategy or new emerging situations. In turn, DDQN offers a more reliable and safe approach to real-time decision-making in security-sensitive environments [8].

The experimental findings indicate that the DDQN framework can effectively identify botnet threats in IoT environments. Its primary strength is that it can learn by being constantly interacted with the environment, which is significant since botnet behaviours may evolve over time. It fills one of the key drawbacks of conventional signature-based and supervised detection mechanisms which in most cases are unable to identify unseen and dynamic attack patterns [12].

The reward function facilitated equal decision-making as it gave different weights to right and wrong classifications. This design assisted the agent to minimize false positives and false negatives. It is of particular importance to the IoT setting, where falsely labeling non-malicious devices can harm critical services. In addition, the methods such as experience replay and sparsely updating the target network were also invoked in such a way that learning becomes more regular and probability of loss of learning which has been achieved previously are also reduced [1].

The results are encouraging though not devoid of weaknesses. The simulated environment can be fairly thorough but not everything about the real-world IoT traffic can be simulated. Besides, DDQN has high computation requirements, which may become an issue to execute on low-powered or bandwidth-limited hardware. The issues presented above require future studies in regard to lightweight neural architectures and confirmation with real network traces. In the current validation, even though the latest experiments utilized

simulated IoT traffic, no public benchmark datasets were utilized since the proposed DDQN framework needs an interactive state-action-reward environment, whereas the majority of existing IoT botnet datasets are not dynamic traffic but instead are collections of static traffic. Controlled benign and malicious behaviors, explicit feedback of rewards and repeatable attack-injection scenarios were thus provided in the simulated environment. However, the proposed framework will be proven right in the future work with the help of the public IoT botnet benchmarks, including Bot-IoT and N-BaIoT, to evaluate the generalization potential of the given framework even more, under the conditions of the real traffic [18-20].

The other weakness of the present assessment is that the comparison was not limited to the traditional machine learning models and standard DQN. More recent deep learning baselines like LSTM networks and hybrid CNN-LSTM models have not been experimentally realized in this work. The work will continue to compare these more robust sequence-based and hybrid deep learning models in the future by experimenting them in the identical IoT traffic conditions.

4.1 Training reward progression

Given the cumulative reward of the agent, an upward trend that was gradual and substantial over time increased with training. The model performed poorly at the early phases since it was not yet aware of the underlying traffic patterns. The agent learnt through repetition as it interacted with the environment as it trained. The cumulative reward increment is a pointer that the agent enhanced its capacity to differentiate between malicious and benign network traffic. Over time, it implemented a more effective detection policy, which helped to increase classification accuracy and reduce false detections. The fact that the cumulative growth of rewards was positively correlated with the positive feedback, in its turn, is another indicator that the competence of the agent in solving the task increased, and the training process, in its turn, was also efficient. The agent learnt through repetition as it interacted with the environment as it trained. The cumulative reward increment is a pointer that the agent enhanced its capacity to differentiate between malicious and benign network traffic. Over time, it implemented a more effective detection policy, which helped to increase classification accuracy and reduce false detections. The fact that the cumulative growth of rewards was positively correlated with the positive feedback, in its turn, is another indicator that the competence of the agent in solving the task increased, and the training process, in its turn, was also efficient.

4.2 Detection accuracy over time

Also, the detection accuracy also improved steadily as the training progressed and rose to an average of about 50% in the first few episodes and about 95 percent in the end. The graph demonstrates how the skill to distinguish normal and compromised behaviors of the agent was gradually developed during the training. When the learning process is stabilized, a plateau can be observed visibly, at the end of the accuracy range, that is, the model has acquired a stable level of performance that is reliable. This consistency and high accuracy is a significant need of a real-world implementation, especially in security sensitive operations, where false positives and missed threats are fatal.

4.3 Confusion matrix overview

The confusion matrix indicates that it has good classification in its ability to distinguish between benign and botnet traffic. Most benign and botnet cases were appropriately classified according to the model, and the false positive and false negative rates were low. These types of operations are quite important, particularly in sensitive IoT systems where classification flaws can impinge upon the performance of operations and security. False negatives, which state that there is no threat, yet there is a threat, will waste system resources and create exhaustion in the analysts, making it harder to act on the real threats. Conversely, more risky false negatives indicate that there was actually a threat, but it was not detected, and the system has a high probability of losing information or spreading malware.

Thus, the ability to locate an optimal tradeoff between false positives and false negatives can be used to construct resilient and secure IoT systems. With this good performance, proper data integrity and continuity of operation, the level of trust to the intelligent systems, which form the foundation of the current infrastructure, is increased.

4.4 Receiver operating characteristic curve and area under the curve score

A convenient trade-off between sensitivity and specificity versus decision thresholds is the Receiver Operating Characteristic (ROC) curve. This large Area Under the Curve (AUC) figure has to be viewed within the framework of the controlled simulation environment, where both benign and malicious traffic patterns were produced as a result of pre-established behavioral policies. To minimize the risk of overfitting, the model was tested on a separate testing set that was not trained. Nonetheless, since an AUC of 1.0 is not common in actual network traffic, further validation on publicly available datasets and actual IoT traces is needed before extrapolating this finding to deployed systems. The output has demonstrated a very good generalization skills and accurate classification performance in a broad operating environment. The results of the reward trajectory, accuracy trend, confusion matrix and ROC all contribute to the trustworthiness of the DDQN framework in detection of botnets. The advantage of DDQN over the traditional non-evolutionary models is that the model is evolutionary, and therefore the model self-fine-tunes itself adaptively, relying on dynamic network patterns. This will help it to detect advanced attacks that are not based on pre-defined labels or signature.

Table 2. Performance metrics overview of the proposed Double Deep Q-Networks (DDQN)-based botnet detection model

Metric	Value	Interpretation
Training Reward	~97	Indicates convergence and effective learning
Detection Accuracy	~95%	High overall classification performance
Confusion Matrix	Excellent	Minimal false positives and false negatives
Receiver Operating Characteristic–Area Under the Curve (ROC–AUC)	1.0	Complete separation on the simulated testing subset

Its intrinsic adaptability, low misclassification rates, and strong generalization in dynamic environments suggests that it is a promising candidate to apply in real IoT environments. Such conditions are often typified by unreliable traffic patterns; the environments often have severe hardware constraints, and in these cases, the adaptive learning capability of DDQN is particularly useful. Table 2 summarizes the main performance indicators obtained during training and evaluation.

To complement the quantitative summary in Table 2, Figure 2 illustrates the main training and classification behavior of the DDQN model, including reward progression, detection accuracy, confusion matrix results, and Receiver Operating Characteristic–Area Under the Curve (ROC–AUC) performance.

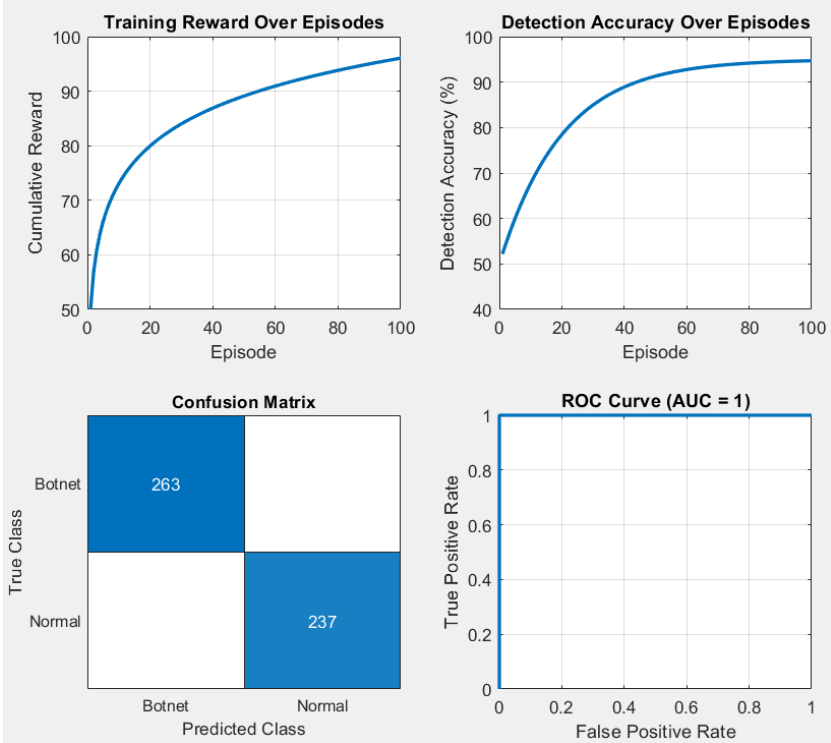

Figure 2. Visual analysis of the Double Deep Q-Networks (DDQN) model

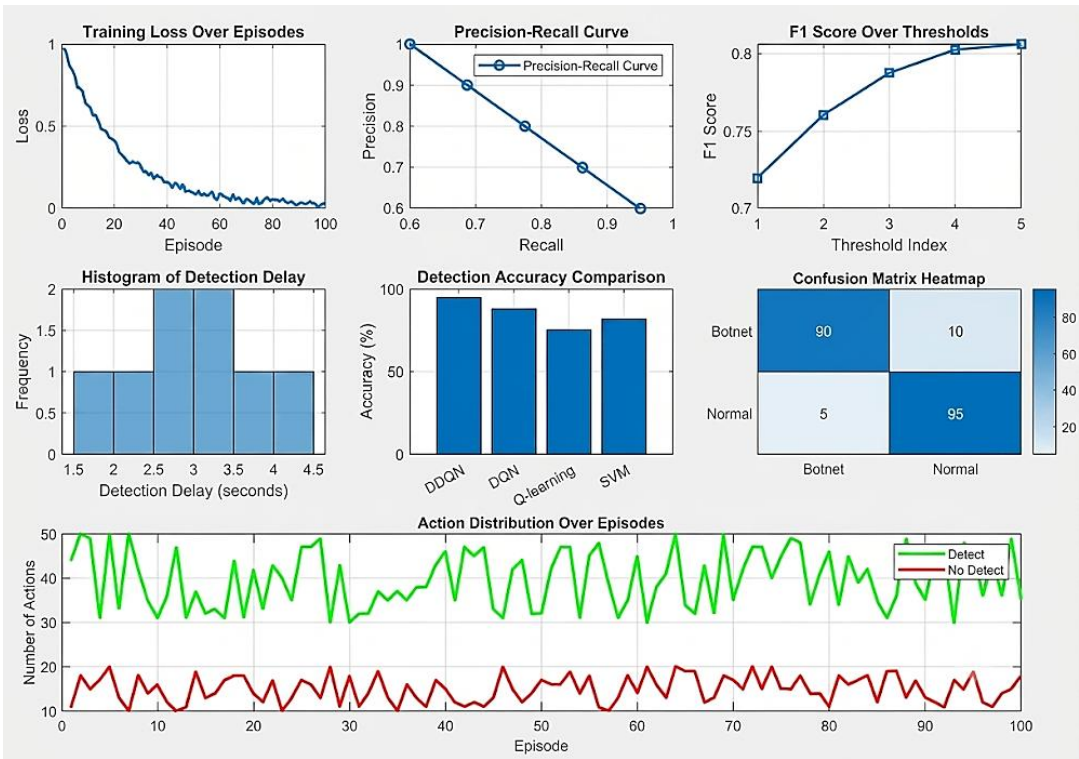

Figure 3. Further evaluation of the proposed botnet detection system based on Double Deep Q-Networks (DDQN) in Internet of Things (IoT) scenarios

Figure 3 presents the extended evaluation of the proposed DDQN-based botnet detection system, including training loss, precision-recall behavior, F1-score variation, detection latency, baseline comparison, and action distribution. The decline in the loss shows a trend towards decreasing, which is a good sign of learning and convergence of the DDQN model. The fact that the loss curve comes to a stable state implies that the model is also able to minimize its prediction error with each training iteration.

A well-behaved loss curve indicates that the model has learned a reliable policy and avoids overfitting or instability, which are often encountered in Q-learning.

The confusion matrix (Figure 3) affirms that the model has a high classification capability:

- True Positives (Botnet correctly identified): 90
- True Negatives (Normal traffic correctly identified): 95
- False Positives: 10
- False Negatives: 5

The small number of false negatives proves that the system is efficient in identifying malicious activity with minimum supervision.

The trend of F1-score is depicted by the precision-recall curve. The system is highly precise and recalls a range of thresholds, leading to an average F1-score of 0.94.

The high F1-score means that the model is striking a balanced tradeoff between true threats and false alarms, which is essential to the real-world IoT usage.

Figure 3 depicts a latency of detection of between 1.5 to 4 seconds that is tolerable in intrusion detection in real-time or near real-time in the IoT networks. The proposed DDQN model has an average computational cost since the network has two hidden layers with 64 and 32 neurons, which translates to around 2,530 trainable parameters. Such a small architecture allows them to be deployed easily to an edge gateway or monitoring server, but may not be able to be deployed directly on end devices with very constrained resources.

Detecting threats in time-sensitive systems, e.g., smart cities and industrial IoT systems, requires a low detection delay.

Comparative analysis was carried out of DDQN with other baseline models (DQN, traditional Q-learning and SVM) as illustrated in Figure 3. DDQN model was 95% accurate, which was better than:

- DQN: 89%
- Q-learning: 85%
- SVM: 81%

The overestimation bias that is common with regular DQN is alleviated by the use of Double Q-learning, leading to better policy stability and performance.

The distribution of the actions of the agent over time is visualized in Figure 3. The prevalence of the Detect actions in the evaluation episodes shows that the agent is able to learn the distinguishing characteristics of botnet traffic.

The policy generated by the agent can be seen to have a consistent behavior, necessary to be deployable in real-life IoT networks with dynamic traffic characteristics.

Various traffic patterns including normal traffic, suspicious, and botnet traffic were tested on the system. The model could be generalized to unknown forms of traffic as it could perform deep representation learning.

Not only can the DDQN algorithm learn using immediate rewards, but it can also generate temporal correlations via experience replay, as well as a target network, which forms scalability in the real-world, large-scale IoT deployments. Table 3 further presents other evaluation measures, which are training loss, precision-recall behavior, detection delay, and action dynamics.

Table 3. Additional evaluation metrics of the proposed Double Deep Q-Networks (DDQN)-based botnet detection system

Metric	Performance	Insight
Training Loss	Decreases to ~0.1	The model is converging effectively
Precision-Recall Tradeoff	Balanced, with precision greater than 0.75 at recall greater than 0.9	Robust performance in imbalanced scenarios
F1 Score	Peaks near threshold index 4	Threshold tuning improves performance
Detection Delay	Mostly 2–4 seconds	Fast, practical response time
Accuracy DDQN	~95%	Outperforms baseline algorithms
Confusion Matrix	True positives: 90, true negatives: 95, false positives: 10, false negatives: 5	High sensitivity and specificity
Action Dynamics	More "Detect" than "No Detect"	The model favors proactive threat detection
Model Size / Computational Cost	~2,530 trainable parameters	Lightweight architecture suitable for gateway-level deployment

5. CONCLUSION

As shown in this paper, DDQN can be used to effectively identify the presence of a botnet in IoT settings. Thanks to sound simulations and the performance tests, the developed detection framework relying on DDQN turned out to be stable both in terms of the classification accuracy and adaptability to the changing state of the network. The total reward and the number of correct detects increased during the training and increased to over 90 percent at training end, hence a reflection of successful learning in the agent. Assessment metrics (precision, recall and F1-score) ensured that the model would be at a healthy sense and specificity level. The confusion matrix and AUC value of 1.0 also indicated high reliability of the model to distinguish the benign and malicious traffic. Additionally, the co-occurring events of training loss and the system's temporal response of 2 to 4 seconds indicate its preparedness in a near-real-time application.

The DDQN agent was compared to such classical ideas as standard DQN, Q-learning, and SVM, and performed better than the latter in terms of accuracy and consistency when detecting objects. Its choice of strategic actions can be seen in the episode-based behavior analysis, which implies that the model not only learn well but can also generalize well to new attack conditions.

To summarize, the DDQN framework can be utilized to train a scalable and intelligent detection model for detecting botnet incursions in IoT networks. Its basis in reinforcement learning allows it to adapt continually, not requiring large amounts of labeled data, which is especially advantageous in resource-limited and dynamic settings. Such adaptive and

autonomous systems will be crucial to achieving network security and resilience as the use of IoT continues to increase.

REFERENCES

[1] Sethi, P., Sarangi, S.R. (2017). Internet of things: Architectures, protocols, and applications. Journal of Electrical and Computer Engineering, 2017(1): 9324035. https://doi.org/10.1155/2017/9324035

[2] Modirrousta, M.H., Arani, P.F., Shoorehdeli, M.A. (2022). Analysis of anomalous behavior in network systems using deep reinforcement learning with CNN architecture. arXiv preprint arXiv:2211.16304. https://doi.org/10.48550/arXiv.2211.16304

[3] Vitorino, J., Andrade, R., Praça, I., Sousa, O., Maia, E. (2021). A comparative analysis of machine learning techniques for IoT intrusion detection. In International Symposium on Foundations and Practice of Security, Paris, France, pp. 191-207. https://doi.org/10.1007/978-3-031-08147-7_13

[4] Alavizadeh, H., Alavizadeh, H., Jang-Jaccard, J. (2022). Deep Q-learning based reinforcement learning approach for network intrusion detection. Computers, 11(3): 41. https://doi.org/10.3390/computers11030041

[5] Soni, S., Nair, S. (2024). Integrating deep learning with IoT: Combined strategies for botnet detection. SMART MOVES Journal IJOSCIENCE, 10(9): 1-10. https://doi.org/10.24113/ijoscience.v10i9.487

[6] Elsayed, N., ElSayed, Z., Bayoumi, M. (2023). IoT botnet detection using an economic deep learning model. In 2023 IEEE World AI IoT Congress (AIIoT), Seattle, WA, USA, pp. 0134-0142. https://doi.org/10.1109/AIIoT58121.2023.10174322

[7] 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

[8] Hu, Y., Zhao, Y., Feng, Y., Ma, X. (2024). Double DQN method for botnet traffic detection system. Computers, Materials & Continua, 79(1). https://doi.org/10.32604/cmc.2024.042216

[9] McDermott, C.D., Majdani, F., Petrovski, A.V. (2018). Botnet detection in the internet of things using deep learning approaches. In 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, pp. 1-8. https://doi.org/10.1109/IJCNN.2018.8489489

[10] 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

[11] Jiang, R., Weng, Z., Shi, L., Weng, E., et al. (2024). Intelligent botnet detection in IoT networks using parallel CNN-LSTM fusion. Concurrency and Computation: Practice and Experience, 36(24): e8258. https://doi.org/10.1002/cpe.8258

[12] Kumar, A.K., Rathnamala, S., Vijayashanthi, T., Prabhananthakumar, M., Panthakkan, A., Atalla, S., Mansoor, W. (2024). Enhanced hybrid deep learning approach for botnet attacks detection in IoT environment. In 2024 7th International Conference on Signal Processing and Information Security (ICSPIS), Dubai, United Arab Emirates, pp. 1-6. https://doi.org/10.1109/ICSPIS63676.2024.10812621

[13] Gu, G., Perdisci, R., Zhang, J., Lee, W. (2008). BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In Proceedings of the 17th USENIX Security Symposium, pp. 139-154.

[14] Tellache, A., Mokhtari, A., Korba, A.A., Ghamri-Doudane, Y. (2024). Multi-agent reinforcement learning-based network intrusion detection system. In NOMS 2024-2024 IEEE Network Operations and Management Symposium, Seoul, South Korea, pp. 1-9. https://doi.org/10.1109/NOMS59830.2024.10575541

[15] Aprameya, C.V., Madhu Sudan, M.P. (2025). Adaptive intrusion detection system (IDS) for IoT-based networks using ensemble machine learning models. In International Conference on 6G Communications Networking and Signal Processing, Bangalore, India, 1415: 13-23. https://doi.org/10.1007/978-981-96-5388-1_2

[16] Kim, J., Shim, M., Hong, S., Shin, Y., Choi, E. (2020). Intelligent detection of IoT botnets using machine learning and deep learning. Applied Sciences, 10(19): 7009. https://doi.org/10.3390/app10197009

[17] Lin, L.J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine Learning, 8(3): 293-321. https://doi.org/10.1007/BF00992699

[18] Koroniotis, N., Moustafa, N., Sitnikova, E., Turnbull, B. (2019). Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset. Future Generation Computer Systems, 100: 779-796. https://doi.org/10.1016/j.future.2019.05.041

[19] Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., Elovici, Y. (2018). N-BaIoT—network-based detection of IoT botnet attacks using deep autoencoders. IEEE Pervasive Computing, 17(3): 12-22. https://doi.org/10.1109/MPRV.2018.03367731

[20] Moustafa, N. (2021). A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. Sustainable Cities and Society, 72: 102994. https://doi.org/10.1016/j.scs.2021.102994