Received: 12 August 2025 | Revised: 11 June 2026 | Accepted: 9 July 2026 | Published online: 23 November 2026

RESEARCH ARTICLE

Journal of Computational and Cognitive Engineering
2026, Vol. 00(00) 1–12
DOI: 10.47852/bonviewJCCE62027179

BON VIEW PUBLISHING

DQLearnCapable: A Comprehensive Intrusion Detection and Prevention System for Network Traffic Analysis using Deep Q-Learning

Norjihan Abdul Ghani[1], Swee Nien Kuck[1], Fahmid Al Farid[2], Rimuljo Hendradi[3], Mohd Shahrul Nizam Mohd Danuri[1], Muneer Ahmad[4], Jia Uddin[5] and Hezerul Bin Abdul Karim[2,*]

[1] *Department of Information Systems, Universiti Malaya, Malaysia*

[2] *Center for Image and Vision Computing, Multimedia University, Malaysia*

[3] *Information Systems Department, Universitas Airlangga, Indonesia*

[4] *Department of Computer Science, University of Roehampton London, UK*

[5] *AI and Big Data Department, Woosong University, Republic of Korea*

Abstract: As technology advances, the variety of cyber threats becomes more apparent in the networking world, presenting new and dynamic attack challenges to network users. Therefore, there is a need for artificial intelligence-assisted or self-governing threat detection systems capable of adaptive learning. In recent years, reinforcement learning methods and traditional machine learning (ML) techniques have been proposed for automated threat detection systems. A new generation of threat detection methods with adaptive learning abilities, combining Q-learning from reinforcement learning with deep neural networks is introduced. This study proposes a Deep Q-learning (DQL) model that learns through a subset of the environment iteratively using a trial-and-error approach to optimize its decision-making. The proposed DQL model is benchmarked against conventional ML and ensemble methods, including logistic regression, stochastic gradient descent, and random forest (RF), using standard performance metrics and execution time analysis. Based on extensive comparisons using the VHS-22 dataset, RF and extra tree classifier achieve noticeably higher detection performance than DQL in terms of classification metrics, while DQL demonstrates substantially lower inference latency and competitive overall effectiveness, although DQL still outperforms other similar ML approaches. Experimental results show that while ensemble models achieve higher detection accuracy, the DQL-based approach shows a competitive performance with significantly lower inference latency, making it more suitable for fast-streaming and time-sensitive intrusion detection scenarios. These findings also highlight the complementary role of reinforcement learning–based intrusion detection systems in environments where adaptability and real-time responsiveness are critical.

Keywords: reinforcement learning, network security, intrusion detection, deep Q networks, heterogeneous dataset

1. Introduction

The frequency of cyberattacks has surged as hackers take advantage of system vulnerabilities to pilfer intellectual property, profit financially, or even affect the entire network infrastructures [1]. Traditional security solutions such as firewalls, intrusion detection systems (IDSs), and virus scanners use a signature-based method that compares a payload hash to a database of known harmful signatures [1–3]. By no means are those security measures enough, since hackers employing the newest attack strategies can easily circumvent them.

With the increasing interest in applying machine learning (ML) to cybersecurity, ML-based applications have gained importance [4, 5]. Real-time identification and response to these threats is made possible for organizations by modeling their threat detection systems using ML approaches. This enhances an organization's overall cybersecurity posture by helping it recognize and prepare for possible cyberthreats [6, 7]. However, there are limitations with traditional ML-based threat detection as it was trained on historical data from one source and has limited generalizability. Most ML-based security systems are trained on historical data and often lack of capability to detect unseen attack types which

***Corresponding author:** Hezerul Bin Abdul Karim, Center for Image and Vision Computing, Multimedia University, Malaysia. Email: hezerul@mmu.edu.my

© The Author(s) 2026. Published by BON VIEW PUBLISHING PTE. LTD. This is an open access article under the CC BY License (https://creativecommons.org/licenses/by/4.0/).

absent from the training dataset. Their limited generalizability reduces their effectiveness in real-world scenarios.

With artificial intelligence (AI) training, AI can hone its superior decision-making skills to develop new attack strategies against intricate cyber systems that may not have occurred to human experts before [6]. An advantageous integration of AI relies on large datasets of historical data that must be available, as they are used for training and testing ML algorithms [8].

Hence, the modeling difficulty and dynamic attack scenarios are still a threat to our cyber-space nowadays. Other challenges are the lack of heterogeneous training data to boost ML-based threat detection systems. The existing datasets usually contain homogeneous network traffic data; therefore, the ML algorithms get easily overfitted to their characteristics [9].

In conclusion, the limitations of traditional ML-based cybersecurity approaches, such as a lack of generalization, contextual awareness, and the capability to adapt dynamically in real time, strongly justify the need for AI-assisted cybersecurity. Therefore, this study proposes integrating advanced AI techniques to continuously monitor network traffic and threat detection environments. ML models are incorporated mainly to address the issue of generalizability in threat detection with adaptive learning capabilities. These models are expected to identify high-risk IP addresses. Additionally, network traffic data may contain too many features for effective ML model training, so dimensionality reduction techniques are employed to solve this problem [10]. To achieve this project's goal, fraudulent threats to an institution will be identified through IDPS and network traffic analysis using information from IP addresses. The effectiveness and generalizability of the threat detection model will be evaluated using appropriate metrics.

The main contributions of this study are as follows:

1) We design and evaluate a Deep Q-learning (DQL)–based intrusion detection, trained on a large-scale heterogeneous network traffic dataset, which integrating original and autoencoder-generated features.
2) We conduct a performance comparison between DQL, traditional ML models, and ensemble classifiers.
3) We analyzed the generalization behavior of DQL among mixed attack and attack-only scenarios and providing valuable insights for network defense environments.

2. Literature Review

Reinforcement learning (RL) with the capability of adaptive learning becomes a common approach in threats detection using automated agents. This agent can learn the behavior of threats through rewarding or penalizing its actions after receiving feedback from the environment (e.g., in a trial-and-error interaction to identify threats). Given its robust conceptual framework, several RL-based intrusion detection techniques have been proposed in recent years to facilitate autonomous cyber defense solutions. This agent learns threat behaviors by receiving rewards or penalties based on feedback from the environment, which typically through trial-and-error interactions in identifying threats. In recent years, there are numbers of RL-based intrusion detection techniques have been proposed due to this powerful framework. This is important to support solutions for autonomous cyber defense. A shortcoming of these RL methods is the problem of state explosion when they encounter very large learning states, which can be solved by utilizing deep neural networks throughout the learning process; hence, the proposed DQL methods have been proposed [11]. Despite the abundance of threat detection systems, the generalizability of the ML model is most preferred to better cope with the limitation of new or unknown cyber-attack scenarios [12]. The techniques are the use of a heterogeneous dataset [9, 11, 13]. Moreover, different dimensionality reduction techniques are used in order to counter the problem of the curse of dimensionality [10].

Aljehane et. al., [14] stated that Q-learning is the best approach for intrusion detection when traditional supervised learning and optimization techniques are underperforming. With this RL method, a root means square error of 0.02883 is obtained with the ISCX NSL-KDD dataset. The advantage of Q-learning is the flexibility of the decision agent for the knowledge of the environment, and the agent is able to perform analysis and detection. Another variation of RL is deep reinforcement learning (DRL), introduced by [15], which has developed a DRL-based powertrain control framework where the policy of the RL can be directly trained based on the environment given to the RL while having a "model-free" concept. This approach has achieved improved performance by 74.6% compared to its previous study using their own augmented data. On the other hand, [16] has developed a distributed DRL intrusion detection system incorporated with an attention mechanism using multiple agents. This research is performed based on NSL-KDD and CICIDS2017 network datasets with an F1-score of 97.8% and 98.9%, respectively. The evaluation against adversarial attacks and zero-day attacks and an F1-score of more than 95% is achieved for both scenarios. Based on [17], DRL is responsive, adaptive, and scalable. It is shown that a comprehensive review of DRL in cybersecurity presents notable DRL applications in cybersecurity and summarizes the challenges, future works, and various techniques in DRL.

Caminero et al. [18] proposed another variation of RL, which is the adversarial environment reinforcement learning algorithm (AE-RL). This AE-RL is the integration of RL and supervised learning, where a simulated environment is provided as a guideline for their model's environment. Setting up the simulated environment involves an interaction of various datasets with pre-recorded samples formed by network information and labels of the data. A sample with an optimized policy is created for the AE-RL agent. The notable achievement by AE-RL is minimizing computation time, a flexible result classifier, and versatility for highly unbalanced datasets. The accuracy of AE-RL on NSL-KDD datasets and AWID datasets is 80.1% and 95.9%, respectively.

Lopez-Martin et al. [19] have provided comprehensive variants of RL algorithms for intrusion detection for supervised cybersecurity problems. In their research, they have applied 4 DRL models, namely DQN, Double Deep Q- Q-Network (DDQN), Policy Gradient (PG), and Actor-Critic (AC), and found out that the DDQN algorithm has the best performance. They have applied novel methods in defining their environment for RL, which replace this environment by using a generic sampling method. The accuracy of DDQN on NSL-KDD datasets and AWID datasets is 89.7% and 95.7%, respectively.

Based on research by [20], DRL is implemented using Markov decision process (MDP). In the experiments, the performance of DQL and *Q*-learning are evaluated using learning episodes, execution times, and cumulative reward. The intensive experimental results showed that MDP-based DRL is suitable for large-scale systems against large volumes of network attacks. Through the study by [21], DQL with a deep neural network that optimizes the *Q*-learning of RL is proposed. Besides proposing a DQL, intensive fine-tuning for the model is also performed, and the optimized parameters for the DQL are used to get an

optimized threat detection result using the NSL-KDD datasets. The average accuracy obtained in predicting the class of a network is 89.1%.

Besides this, federated and collaborative anomaly detection frameworks have been explored in addressing privacy, scalability, and latency challenges, especially in distributed environments. The recent study in [22] proposed an anomaly detection method for industrial control systems (ICS) that extracts multilevel features from application-layer traffic without prior protocol knowledge. Feature automatic marking algorithm is introduced to labelling feature sequences. A combination of convolutional neural network (CNN) and gated recurrent units (GRU) are used to classify between normal and anomalous behavior. A federated DNN model for anomaly detection for the IoT data is introduced in [23]. The results show that the model achieves greater performance compared to existing federated deep learning approaches for IoT anomaly detection. In [23], a federated learning framework which automatically searching for lightweight CNN architectures is optimized for intrusion detection in ICS environments. This automated federated learning approach for intrusion detection in industrial control systems shows a balancing model performance with computational cost and communication efficiency under a federated setting. In addition, federated and end-edge collaborative learning approaches have gained increasing attention for distributed intrusion and anomaly detection. Wang et al. [24] introduced a federated deep learning framework for IoT anomaly detection, which enable multiple nodes to collaboratively train detection models while preserving data privacy. However, unlike federated and industrial-focused approaches, most of the existing works focus with evaluation of the adaptability and generalization capability of a DQL-based intrusion detection model using a heterogeneous network traffic dataset

Model generalization is then studied using heterogeneous datasets. As stated by [8], the authors developed the VHS-22 datasets by combining ISOT, CICIDS2017, CTU-13, Booters, and the MTA dataset to address the challenges of model overfitting caused by the use of homogeneous datasets. In this study, random forests (RF) and stochastic gradient descent (SGD) are employed, achieving the highest accuracy of 92% and 97% respectively, when trained on flow-level and network-level features. In addition to evaluating the models' performance, the data collection methods used to create VHS-22 are also noteworthy. All network-level and flow-level features are extracted from at least one packet capture (PCAP) file, with the information processed and transformed before being compiled into a single dataset. Furthermore, [25] used the VHS-22 dataset as well. They employed models including RF, Decision Tree (DT), Multi-Layer Perceptron, K-Nearest Neighbor (KNN), Naive Bayes, Bernoulli Naive Bayes, and Logistic Regression. The findings showed that the best-performing models were RF, DT, and KNN, achieving accuracies of 99.8%, 99.76%, and 99.6%, respectively.

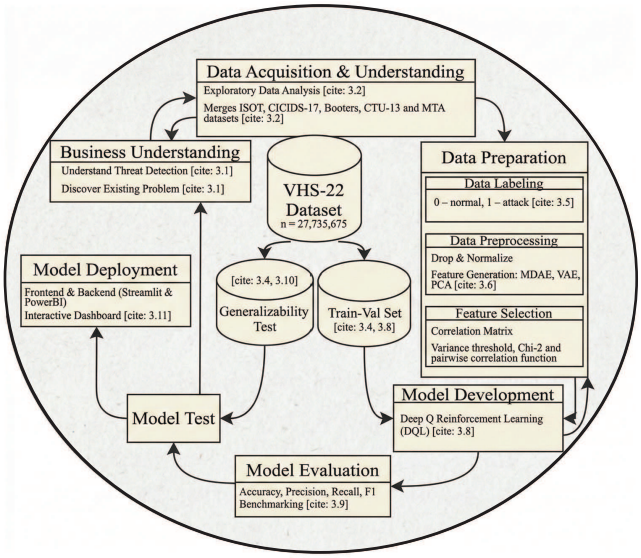
3. Methodology

The Cross-Industry Standard Process for Data Mining (CRISP-DM) Model is applied in this research. CRISP-DM consists of six phases that guide for data mining process as shown in Figure 1.

3.1. Data acquisition

The dataset is VHS-22, which is a heterogeneous, flow-level, and network-level dataset that integrates ISOT, CICIDS-17,

Figure 1
CRISP-DM model



Booters, CTU-13, and Malware Traffic Analysis (MTA) datasets to increase the variety of any malicious or legitimate traffic flows. The combination of these datasets was chosen based on the justification discussed in [9]. The information and features are extracted from at least one packet capture (PCAP) file and distributed into 45 features, apart from classical NetFlow features and 3 labels. The integration of 5 datasets into 1 increases the variation of malicious and legitimate traffic flows, which has the potential to increase the generalizability of the model. It contains 27.7 million flows (20.3 million legitimate and 7.4 million attacks) collected between January 1, 2022, to January 23, 2022, as shown in Table 1. Each flow is labeled with the class label (0 - normal, 1 - attack) [9, 25]. Therefore, this research will utilize the VHS-22 dataset, which was created from such a diversity of sources and is heterogeneous.

Table 1
VHS-22 attack statistics

Attack	Flow numbers
DoS	6 778 273
Botnet	520 600
MTA Traffic	85 331
Brute Force	7 022
Web	4 048

3.2. Data preprocessing

For data cleaning and wrangling, missing values and duplicates are truncated. A subset of datasets of 1 million records from the main VHS-22 dataset is sampled using stratified sampling with a purpose to reduce computational load during training and evaluation while preserving the characteristics of the population dataset [25]. Validity of the sampled dataset is checked on the characteristic comparison between the original and sampled datasets. Then, 0-valued, time, identifier, and repeating information columns are dropped, and the datasets are normalized,

and extra features are generated using variational autoencoder (VAE), principal component analysis (PCA), and multiple denoising autoencoder (MDAE). There are flows from sampled VHS-22 are then divided to training and validation subsets of 80:20. Two test datasets are sampled for generalizability testing. The first test datasets are sampled using stratified sampling, and it consists of 1 million records with no duplication from the train and validation samples. The second test dataset consists of 300k records with attack networks only, with no duplication from the train and validation samples. Although all samples originate from VHS-22, the dataset integrates traffic from five independent sources with different attack characteristics and collection environments. For unseen conditions simulation, where training, validation, and test sets were strictly separated with no overlapping flows. Additionally, an attack-only test was developed to assess generalization across heterogeneous attack distributions.

3.3. Exploratory data analysis

The distribution of flow-level features and inferred the distribution to all other features is visualized. Figure 2 and 3 show the distribution of the data is skewed to the left, with zero-valued features taking the majority of counts. The histogram for flow duration (Figure 2(a)) contains only flow under 160 seconds, omitting flows that are too long. It can notice that the majority of flow lasts between 0 to 50 seconds. The omitted flows originate from ISOT dataset, with duration length around 33 minutes, 1 hour 43 minutes and 2 hour 15 minutes. In Figure 2(b), most of the network flows contain fewer than 500 packets. Meanwhile in Figure 3(a), incoming connections contain only UDP and TCP protocols, with 62% and 38%, respectively, and Figure 3(b) highlights the number of attack networks and breached data across 1 million connected networks over 22 days.

Tables 2 and 3 show characteristics of normal traffic and attack traffic based on connected durations and number of packets. In normal circumstances, 27% of connected networks are threats, while 83% of attack flows are associated with the 'Packet no. = 0' category. Figure 4 shows features with only 0 values, and these columns contribute no value for our model development; therefore, these features were dropped.

3.4. Feature extraction

After feature generation technique using VAE, PCA, and MDAEs, the total features are 40. The relationships between features are first observed using a correlation matrix with Spearman's rank correlation coefficient, as shown in Figure 5. Then, the 10 most correlated features (Table 4) are chosen through the variance threshold filter, 25 k-best with chi-squared filter, 15 k-best with mutual info classifier, and pairwise correlation function. The selected features are 'ip_protocol', 'sport', 'b_packet max', 'b_packet median', 'b_packet_third_q', 'connections_from_this_port', 'connections_ratio_to_this_host', connections_ratio_from_this_port', 'PCA1', 'MDSAE2'. In this study, VAE and MDAE were used to generate extra features from the original input variables. Both models were trained using the training split only to avoid data leakage. The VAE encoder and decoder were implemented using fully connected layers with ReLU activations. The VAE was trained by minimizing reconstruction loss together with KL divergence. For MDAE, multiple levels of noise were added to the input, and the autoencoder was trained to reconstruct the clean input. The output from the bottleneck layer was used as the generated feature representation and added to the final feature set used for model training.

Figure 2
Histogram of flow duration of 95%: (a) histogram of flow duration; (b) histogram of flow length (in packets)

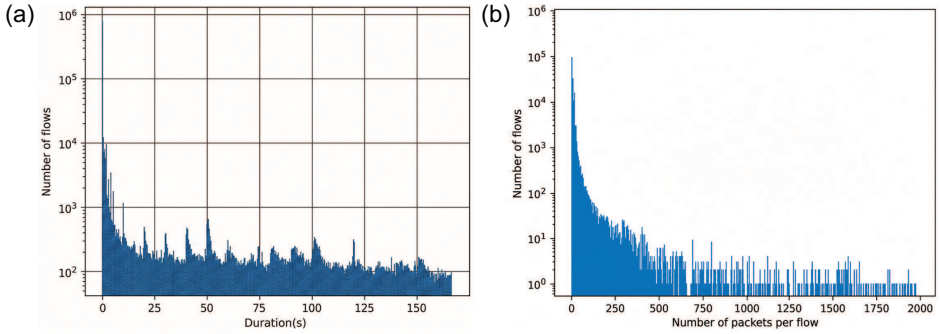


Figure 3
Connections by protocol; (b) connections over time

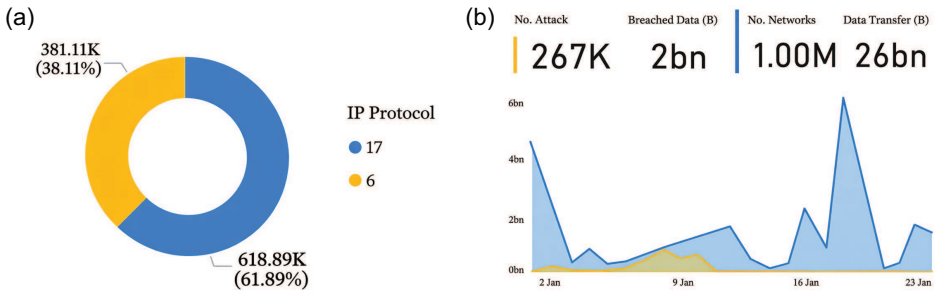


Table 2
Distribution of normal traffic and attack data

Normal traffic		Attack	
Duration > 0	Duration = 0	Duration > 0	Duration = 0
403,559	329,807	45,044	221,589
733,366 (73%)		266,633 (27%)	
Total: 999,999			

Table 3
Zero-duration flows (1-packet flow) (description & footnote)

Normal traffic		Attack	
Packet no. = 0	Packet no. => 0	Packet no. = 0	Packet no. > 0
329,540 (45%)	403,826	221,564 (83%)	45,069
733,366		266,633	
Total: 999,999			

Figure 4
Zero-valued columns: (a) zero-valued columns of connections ratio rst from this host; (b) zero-valued columns of connections rst from this host

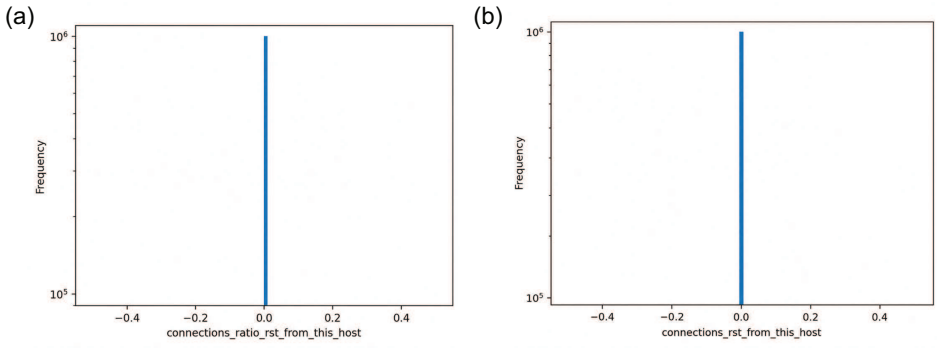


Figure 5
Correlation matrix of chosen features

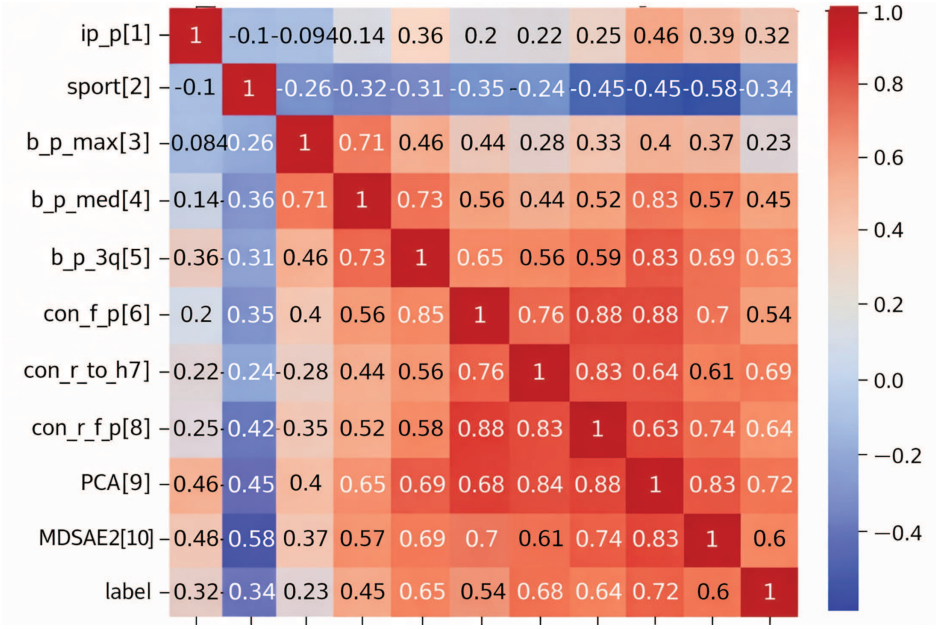


Table 4
Description of selected features

Features	Description
ip_protocol	Fourth layer protocol
Sport	Source port
b_packet max	Size of the largest packet
b_packet median	Median packet size
b_packet_third_q	3rd quantile of packet size
connections from this port	Number of connections with the same source port number
connection ratio to this host	% of collections from the host with the same source address
connection ratio from this host	% of collections from the host with the same source port
PCA1	Engineered feature by principal component analysis using 45 original features
MDSAE2	Engineered feature by a multi-noise autoencoder using 45 original features
Label	Label for attack or normal traffic

3.5. Deep Q-Learning Model Development

3.5.1. Deep Q-learning model

To build a threat detector with adaptive learning capability, a DQL-based RL model is trained [21]. Q-learning is one of the most effective variants of RL. By incorporating deep neural networks into the Q function approximator, the model becomes DQL. Q function approximator is a learning agent represented by $Q(S, A; \theta)$. The S, A, θ denotes the state, action, and neural network weights of the Q function, respectively. When an action A is selected by a DQL agent, reward R is awarded. Moreover, the next state for the DQL model is represented by S', where the DQL agent shifts to the next state S' after performing action A on the previous state S. To perform action A, deep Q-networks must estimate and predict target Q-values, and action A is obtained through the argmax operation. The neural network weights θ is optimized through the loss function of deep Q-networks based on Q-values of current and previous states.

DQL has 5 main concepts. The concepts are environment, agent, states S, actions A, and rewards R. The environment is the dataset that used to train the model.

State representation

In our implementation, each state represents one network flow record from the training dataset. The state is a numeric feature vector created from the 10 selected features shown in Table 4. Therefore, the state at time step t is defined as:

$$S(t) = [f1,\ f2,\ \ldots,\ f10]$$

where f1 to f10 are the selected features (ip_protocol, sport, b_packet max, b_packet median, b_packet_third_q, connections_from_this_port, connections_ratio_to_this_host, connections_ratio_from_this_port, PCA1, MDSAE2). Each feature value is normalized during preprocessing.

Action space

The agent selects one of two actions for each state (flow):

Action 0: normal traffic

Action 1: attack traffic

Reward function

After the agent selects an action, the environment returns a reward based on whether the prediction is correct compared to the true label. The reward $R(t)$ is defined as:

$R(t) = +1$ if predicted label equals true label

$R(t) = -1$ if predicted label is different from true label

This reward design encourages the agent to maximize correct classifications over time.

Next state

After processing one flow, the next state $S(t + 1)$ is the next flow record sampled from the dataset. This process continues until the end of the episode.

The DQL agent will obtain features from the environment to create action vectors AV and compute reward vectors based on the model prediction. The values in the environment are unaffected by this process. In DQL, agents train the model for the best future reward values RV. Based on the current state S and selected action A, the agent interacts with the environment and receives a corresponding reward R. Feedback from agents is used to optimize the deep Q-networks. During the training process, the agent will perform actions based on an epsilon-greedy exploration policy. Under this strategy, the agent either selects a random action with a probability of ϵ to promote exploration or exploits current knowledge by selecting the action that maximizes the value function with a probability 1-epsilon. State S describes the batch with a certain size from the environment that is chosen for the current training. Action A is a decision made by an agent after the agent has processed the environment for a given episode. Rewards R is the feedback corresponding to the action done by an agent. The reward value RV serves as a parameter to calculate target Q QT value to optimize the Q-networks through the loss function.

Based on the DQL training phase flowchart proposed in [21], The training data are added to the environment. The DQL agent then starts by defining the Q value vector, the state variables, and the possible actions according to the chosen batch size. In the learning iteration, the reward function is operated. The target Q-value QT_i is then calculated based on Equation (1) the rewards RV_i, next state's Q-value $Q'V_i$, and discount factor for future rewards γ. Learning is improved by computing the loss using the target Q-value QT_i and current Q-value QV_i. During the learning process, each outer iteration represents a new episode where state values are reset after each of the iteration. However, the DNN keeps its learned parameters where it is not initialized for each of the iteration. The state Sn on every discrete state is determined based on the training data sample provided at that point (Batch_n). Upon reaching the end of an episode, a complete sequence of states, rewards, and actions are obtained in the terminal state. Next, while the training starts, the agents will receive the first batch and indicate the starting state S1 of the environment.

$$QT_i = RV_i + \gamma . QV_i \quad (1)$$

$$QV = Predict(S) \quad (2)$$

The deep Q-networks consist of 4 layers as proposed in [21]. ReLu is known as the activation layer for to guarantee a positive Q-value. The 2 hidden layers in between consist of 100 neurons, and the output layer has a size of 2 to identify attack networks using 2 labels. The training of the Q-network is optimized using mean squared error (MSE) loss using target Q-value QTi and current Q-value QVi based on Equation (3).

$$Loss = \frac{1}{n} \sum_{n} \left(\frac{QV_n}{Prediction} - \frac{QT_n}{Target} \right)^2 \quad (3)$$

Parameter fine-tuning is performed as the optimized parameters can be referred to in Table 5. There are four performance metrics used: accuracy, precision, recall, and F1 score. Accuracy is the quality of being correct, using True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) as shown in Equation (4). Precision is the quality of being exact and correct, as it measures how many of the "positive" predictions made by the model are correct using TP and FP as shown in Equation (5). Recall is the action of remembering learned experience, and it measures the correctly predicted positive class of the sample using TP and FN as shown in Equation (6). F1 score is an evaluation metric that determines a model's accuracy by calculating the harmonic mean of precision and recall as shown in Equation (7). The values of TP, TN, FP, and FN are listed in Table 6.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (4)$$

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

$$Recall = \frac{TP}{TP + FN} \quad (6)$$

$$F1\ Score = \frac{2\ x\ Precision\ X\ Recall}{Precision + Recall} \quad (7)$$

3.5.2. Model evaluation

During the DQL training process using fine-tuned parameters, the best accuracy, precision, recall, and F1 score on the 200k-element validation set are 96.12%, 95.05%, 92.1%, and

Table 5
DQL fine-tuned parameters

Hyperparameter	value	Description
Neural network layer	4	This refers to the number of layers in the neural network used to approximate the Q-function in DQL. The complexity of the problem often dictates the number of layers
Hidden layer size	100	This is the number of neurons in each hidden layer of the neural network. A larger size can increase the capacity of the model to learn complex patterns, but it may also lead to overfitting
Batch size	512	This is the number of experiences sampled from the memory to train the network at each step. A larger batch size can lead to more stable updates, but it also requires more computational resources. This is the number of complete sequences of interaction between the agent and the environment. Each episode is a complete game or sequence from start to finish This refers to the maximum number of steps in each episode. It determines how long an episode can last
Number of episodes	200	
Episode iteration	100	
Epsilon	0.1	This is the exploration rate in the epsilon-greedy policy. It determines the probability of taking a random action. A high epsilon value encourages more exploration, while a low value encourages more exploitation. The epsilon will change over the training period based on gamma and decay rate
Decay rate	0.99	This is the rate at which epsilon decreases over time. A high decay rate makes the agent shift from exploration to exploitation more quickly
Gamma	0.001	This is the discount factor used in the Q-learning update. It controls how much the agent values future rewards. When gamma is close to 0, the agent focuses mainly on immediate rewards. When gamma is close to 1, the agent gives more importance to rewards that may come later.

Table 6
Position of TP, TN, FP, and FN in confusion matrix

		Predicted	
		Normal traffic	Attack traffic
Actual	Normal traffic	True Negative (TN)	False Positive (FP)
	Attack traffic	False Negative (FN)	True Positive (TP)

93.57%, respectively. Figure 6 illustrates the values for loss and reward.

4. Experimental Results Analysis and Discussion

The following subsections present the model performance based on experimental results along with a details discussion.

4.1. Model performance

The performance of DQL is assessed by two test datasets. Both of the test datasets are chosen in a way where the test data does not exist in the training dataset. The first test dataset is sampled using stratified random sampling with 1 million records. The second test dataset consists only of attack network data with a size of 300k records. After evaluating the performance of DQL, its performance is compared with other traditional ML models, linear support vector machine (LSVC), logistic regression (LR), and SGD, and ensemble ML models, RF classifier, and extra tree (ET) classifier. This study compares the results by using evaluation metrics such as accuracy, recall, precision, and F1 score.

Referring to Figures7(a) and 7(b), it is observed that the values of FN and FP for RF are 3 times and 5 times lower, respectively, at 6855 and 2694, compared to DQL, which has values of 21137 and 12893, respectively. This proves that RF has a lower tendency to misclassify attack traffic as legitimate traffic than DQL.

As shown in Table 7, DQL, with an F1 score of 93.52%, outperforms traditional ML models (LSVC, SGD, and LR), which have scores of 82.33%, 82.40%, and 82.08%, respectively. However, its performance is not as good as the ensemble models (RF, ET), which have scores of 98.20% and 97.97%. Despite this, DQL is 5 times and 9 times faster than RF and ET in terms of scoring time, taking 21 seconds compared to RF and ET, which take 129 seconds and 178 seconds, respectively.

Figure 6
Training process of DQL

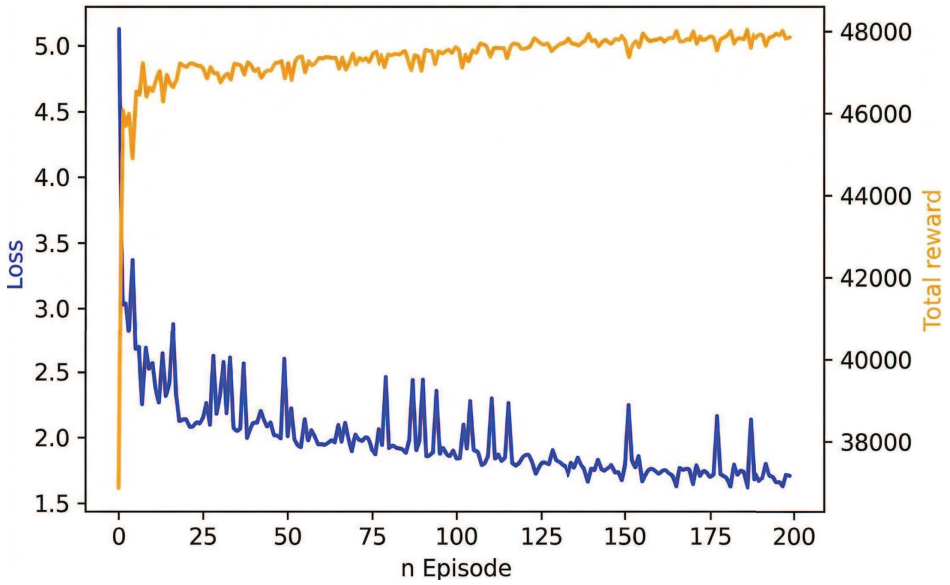
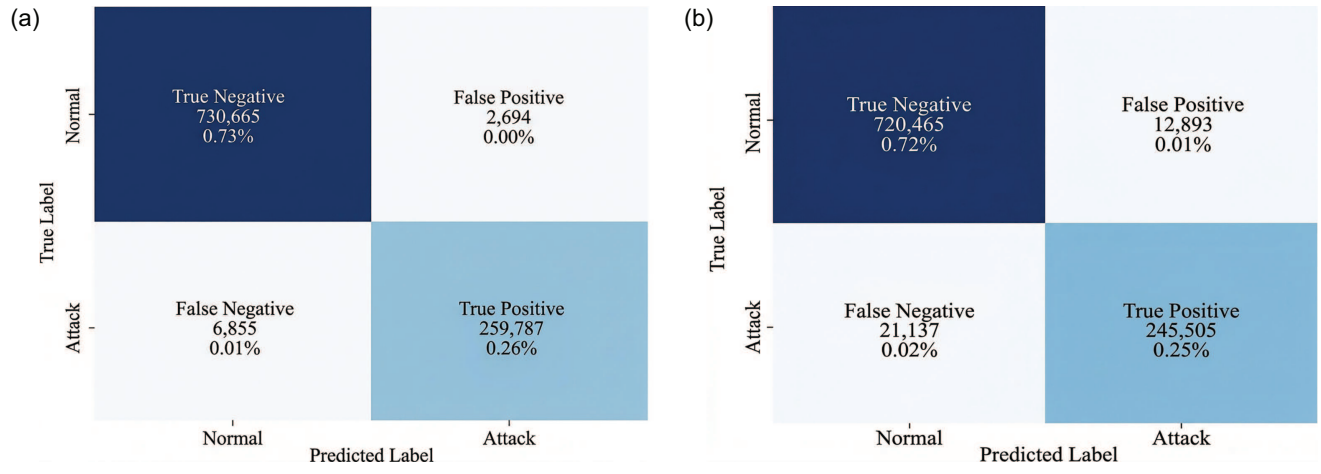


Figure 7
Confusion matrix unseen attack and normal test data (1 M): (a) RF; (b) DQL



Besides testing the models on a stratified sample of the dataset, it is also needed to assess the ability of the models to generalize to all patterns of attack in 5 of the datasets. To evaluate this, a total of 300k attack traffic is sampled from the datasets using stratified sampling based on the type of attack. The performance of the models can be evaluated in Table 7.

To evaluate a model's ability to identify threats in network traffic, the recall score is evaluated. This performance metric considers TP and FN, where FN represents attack traffic that was falsely identified, and TP represents attack traffic that was correctly identified. Based on Table 8, RF has the highest recall score with a value of 87.79%, while SGD has the lowest recall score with a value of 70.22%. DQL with a recall score of 75.48% is higher than traditional ML models (LSVC, SGF, and LR) with recall scores of 74.48%, 70.22%, and 75.19%, respectively. However, DQL is outperformed by ensemble models (RF and ET) with a recall of 87.79% and 82.21%, respectively.

Based on Figures 8(a) and 8(b), it can be observed that the performance of RF is twice as good as DQL. RF falsely classified 36,630 instances of attack traffic, compared to DQL, which falsely classified 73,436 instances.

4.2. Discussion

Numerous studies and research have been conducted on DQL, but still lack of researches that combine original and generated features for model training and testing. The evaluation and analysis reveal the key insights regarding the effectiveness, generalization, and operational suitability of the proposed approach. Based on the results, the most balanced model in terms of threat detection power and evaluation time is DQL, while the best-performing model with a high evaluation time is random forest (RF). It can be concluded that DQL performs best in an environment with fast streaming network data, while RF excels in an environment where false predictions are undesirable.

Comparative performance evaluation against established classifiers, including Random Forest (RF), SGD, and others, confirms the competitiveness of the DQL model. While traditional models like RF achieve high classification accuracy, they lack the adaptability and real-time responsiveness demonstrated by DQL. Furthermore, the study includes generalization testing to assess performance stability across unseen data, reinforcing the robustness of the DQL approach in real-world deployments.

The F1-score of DQL ranges from 86.03% to 93.52%, and the recall score ranges from 75.48% to 92.07%. On the other hand, the F1-score of RF ranges from 93.495% to 98.20%, and the recall score ranges from 87.79% to 97.43%. The evaluation time for 1 million instances of traffic data for DQL and RF is 21 seconds and 129 seconds, respectively. Since these models are able to perform well in a 5-dataset integrated environment with a high detection rate of threats, we can comfortably say that these models generalize well in detecting threats in network systems.

Although the experiments only used the VHS-22 dataset, the heterogeneous construction of VHS-22 allows for evaluation across multiple traffic domains. The experimental results show that DQL maintains stable performance across the distribution of unseen attacks, suggesting reasonable generalizability. However, evaluation on fully independent datasets and live traffic flows remains an important direction for future work.

Both DQL and RF have their strengths and are suited to different environments. DQL is ideal for situations where data is streaming quickly and decisions need to be made rapidly, while RF is more suited to situations where the cost of a false prediction is high and more time can be allocated for evaluation. The choice between these two models depends on its specific requirements needed and constraints provided by the use case.

The adoption of DQL offers an improved approach that employs a RL-based approach, particularly DQL, for intrusion detection. By learning optimal policies through interaction with the environment, the agent adapts to dynamic network conditions and evolving threat patterns, which is a significant advantage over static ML models. One of the primary strengths of this study is that it lies in its use of the VHS-22 dataset, which is known for being a heterogeneous, multi-source dataset. This diversity allows the model to learn robust representations, enhancing its ability

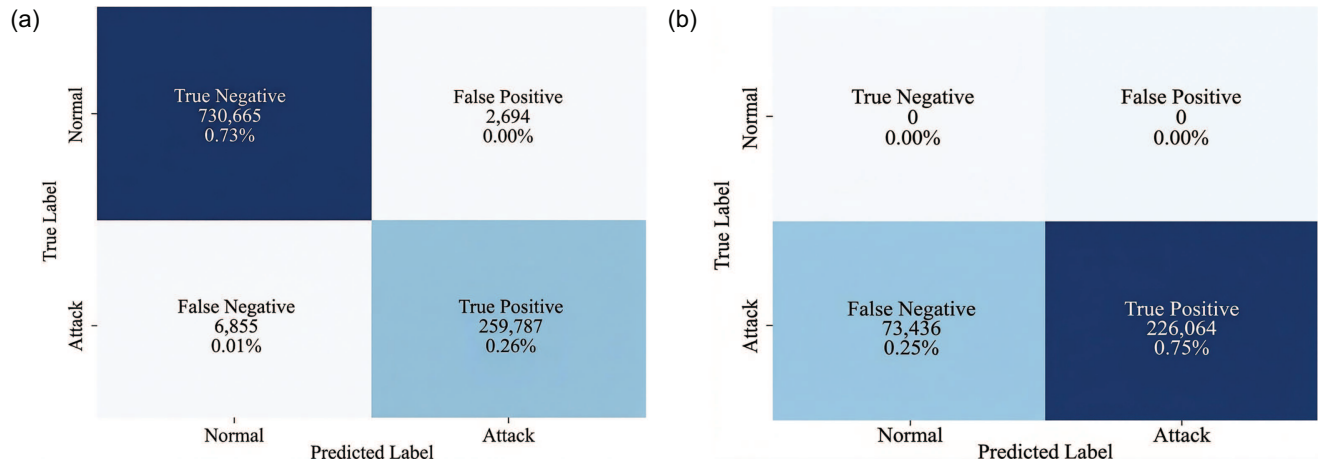
Table 7
Performance on unseen data with attack and normal traffic (1 M)

Model	Score Time	Accuracy	Precision	Recall	F1
RF	129.218	99.05	98.97	97.43	98.20
ET	177.948	98.93	99.25	96.71	97.97
LSVC	0.032	91.42	91.31	74.96	82.33
SGD	0.035	91.51	92.13	74.53	82.40
LR	0.027	91.27	90.71	74.95	82.08
DQL	20.676	96.60	95.01	92.07	93.52

Table 8
Performance on unseen data with attack traffic only (300K)

Model	Score time	Accuracy	Precision	Recall	F1
RF	17.850	87.79	1	87.79	93.49
ET	21.357	82.21	1	82.21	90.23
LSVC	0.015	74.48	1	74.48	85.37
SGD	0.015	70.22	1	70.22	82.50
LR	0.015	75.19	1	75.19	85.84
DQL	4.491	75.48	1	75.48	86.03

Figure 8
Confusion matrix unseen attack and normal test data (300k): (a) RF; (b) DQL



to generalize across varied input patterns. The inclusion of both original and synthetically generated features in the training process further strengthens model diversity and resilience to data imbalance or sparsity.

It is acknowledged that while Table 1 presents five distinct attack categories with notably skewed distributions, ranging from DoS (6,778,273 flows) to Web attacks (4,048 flows), the evaluation presented in this study was conducted at the binary classification level (attack vs. normal traffic). Although the 300K attack-only test set was constructed using stratified sampling to preserve the proportional representation of all five attack categories, per-class detection performance for minority attack types such as Brute Force (7,022 flows) and Web attacks (4,048 flows) was not explicitly disaggregated in the reported results. Binary aggregate metrics, while informative at the system level, may not fully reveal the model's sensitivity toward these underrepresented attack classes. This constitutes a limitation of the current study, and future work will address this through class-stratified evaluation

4.2.1. Ensemble model performance and hyperparameter requirement

To ensure transparency in the comparative evaluation, Table X summarizes the hyperparameter configurations used for each baseline model. All traditional and ensemble classifiers—random forest (n_estimators = 100, max_depth = None, min_samples_split = 2), extra trees (n_estimators = 100, max_depth = None, min_samples_split = 2), linear SVC (C = 1.0, max_iter = 1000), logistic regression (C = 1.0, solver = 'lbfgs', max_iter = 100), and SGD (loss = 'hinge', alpha = 0.0001, max_iter = 1000)—were evaluated using their standard scikit-learn default configurations, which are well-established and widely benchmarked in the intrusion detection literature. The DQL model, by contrast, underwent explicit hyperparameter fine-tuning as detailed in Table 5.

The results show that ensemble models such as RF achieves higher accuracy and F1-score compared the DQL-based approach. This finding is static with supervised intrusion detection tasks involving tabular network traffic data. In contrast, the DQL model is suitable for a decision-making and low-latency inference rather than maximizing classification accuracy on a fixed dataset.

The models were trained and evaluated using identical data partitions and feature configurations. In this experiment, hyperparameters for traditional and ensemble models were selected based on the most commonly used. This study also uses a consistent tuning strategy across models.

5. Conclusion

In conclusion, this study introduced a DQL RL model for detecting threats in network traffic. The proposed DQL model takes both flow-level and network-level information from heterogeneous datasets, as well as features generated using principal component analysis and multiple denoising autoencoder, as input. It then applies a deep RL strategy on Deep Q Networks.

The DQL model proposed in this study combines intensive feature formatting methods with a deep feed-forward neural network, which is used to interact with the network environment. It learns through a trial-and-error strategy to identify malicious threats in network flow. The design of the DQL model takes into account the connection with the network environment, the DQL agent, batches of states, actions, and the reward system. The performance of the DQL model is further enhanced through fine-tuning of its training episodes, discount factor, and batch sizes. Through intensive experimental efforts in parameter fine-tuning, shown that the optimal discount factor (gamma) for the DQL model is 0.001, with 200 episodes of learning and a decay rate of 0.99.

The experimental results show that the proposed DQL model can effectively learn from the heterogeneous environment and outperform traditional ML- based models in terms of accuracy and ensemble models in terms of evaluation time. The autonomous learning manner in the heterogeneous environment enables the proposed model to generalize well across five different datasets: ISOT, CICIDS-17, Booters, CTU-13, and Malware Traffic Analysis datasets. One limitation of this study is the real-time learning capability of the DQL model when identifying threats in real-time network environments. The model can be costly when identifying zero-day attacks, where the attack patterns are novel, as the DQL model will only learn after a period of time in batches. For future work, it would be beneficial to explore methods for reducing the learning time of the DQL model, especially in the context of zero-day attacks. Additionally, investigating ways to improve the model's generalizability across different types of network traffic could also be valuable. Finally, ensembling the DQL model with other ML or deep learning models might enhance its

performance and make it more robust against various types of network threats.

Based on our results, DQL is more suitable for time sensitive and fast streaming environments because it gives much faster scoring time during deployment. For example, on 1 million flows, DQL took about 21 seconds, while RF took about 129 seconds. DQL is a better option for low latency detection, while RF is a better option for high accuracy and safer decision-making.

This study also has some limitations that should be acknowledged. First, the proposed framework was evaluated using labeled historical traffic, so it may face challenges in detecting zero-day attacks that show patterns not present in the training data. Second, real deployment in backbone networks may require additional system-level optimizations such as model compression, parallel processing, or edge-based execution to maintain high throughput at scale, which may also introduce new tradeoffs. Addressing these issues will be an important direction for future research and further system-level validation.

Additionally, future work should extend the evaluation to a per-class breakdown across the five attack categories in VHS-22, with particular attention to minority classes such as Brute Force and Web attacks, and explore class-weighted training objectives or reward shaping strategies to improve detection sensitivity on underrepresented attack types.

Funding Support

This research was funded by Multimedia University, Cyberjaya, Selangor, Malaysia [Grant Number: PostDoc (MMUI/240029)].

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

The data that support the findings of this study are openly available in Kaggle at https://www.kaggle.com/datasets/h2020simargl/vhs-22-network-traffic-dataset.

Author Contribution Statement

Norjihan Abdul Ghani: Conceptualization, Validation, Writing – review & editing, Supervision. **Swee Nien Kuck:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Visualization. **Fahmid Al Farid:** Writing – review & editing, Funding acquisition. **Rimuljo Hendradi:** Supervision, Project administration. **Mohd Shahrul Nizam Mohd Danuri:** Supervision, Project administration. **Muneer Ahmad:** Writing – review & editing, Funding acquisition. **Jia Uddin:** Writing – review & editing, Funding acquisition. **Hezerul Bin Abdul Karim:** Writing – review & editing, Funding acquisition.

References

[1] Singh, R., Kumar, H., Singla, R. K., & Ketti, R. R. (2017). Internet attacks and intrusion detection system: A review of the literature. *Online Information Review*, *41*(2), 171–184. https://doi.org/10.1108/OIR-12-2015-0394

[2] Bridges, R. A., Glass-Vanderlan, T. R., Iannacone, M. D., Vincent, M. S., & Chen, Q. (2019). A survey of intrusion detection systems leveraging host data. *ACM Computing Surveys*, *52*(6), 1–35. https://doi.org/10.1145/3344382

[3] Tariq, U., Ahmed, I., Bashir, A. K., & Shaukat, K. (2023). A critical cybersecurity analysis and future research directions for the internet of things: A comprehensive review. *Sensors*, *23*(8), 4117. https://doi.org/10.3390/s23084117

[4] Apruzzese, G., Laskov, P., Montes de Oca, E., Mallouli, W., Brdalo Rapa, L., Grammatopoulos, A. V., & Di Franco, F. (2023). The role of machine learning in cybersecurity. *Digital Threats: Research and Practice*, *4*(1), 1–38. https://doi.org/10.1145/3545574

[5] Mohamed, N. (2023). Current trends in AI and ML for cybersecurity: A state-of-the-art survey. *Cogent Engineering*, *10*(2), 2272358. https://doi.org/10.1080/23311916.2023.2272358

[6] Applebaum, A., Miller, D., Strom, B., Korban, C., & Wolf, R. (2016). Intelligent, automated red team emulation. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, 363–373. https://doi.org/10.1145/2991079.2991111

[7] Safitra, M. F., Lubis, M., & Fakhrurroja, H. (2023). Counterattacking cyber threats: A framework for the future of cybersecurity. *Sustainability*, *15*(18), 13369. https://doi.org/10.3390/su151813369

[8] Szumelda, P., Orzechowski, N., Rawski, M., & Janicki, A. (2022). VHS-22: A very heterogeneous set of network traffic data for threat detection. In *Proceedings of the 2022 European Interdisciplinary Cybersecurity Conference*, 72–78. https://doi.org/10.1145/3528580.3532843

[9] Yang, Y., Gu, Y., & Yan, Y. (2023). Machine learning-based intrusion detection for rare-class network attacks. *Electronics*, *12*(18), 3911. https://doi.org/10.3390/electronics12183911

[10] Sewak, M., Sahay, S. K., & Rathore, H. (2021). Deep reinforcement learning for cybersecurity threat detection and protection: A review. In R. Krishnan, H. R. Rao, S. K. Sahay, S. Samtani, & Z. Zhao (Eds.), *Secure knowledge management in the artificial intelligence era* (pp. 51–72). Springer. https://doi.org/10.1007/978-3-030-97532-6_4

[11] Shaukat, K., Luo, S., Varadharajan, V., Hameed, I. A., Chen, S., Liu, D., & Li, J. (2020). Performance comparison and current challenges of using machine learning techniques in cybersecurity. *Energies*, *13*(10), 2509. https://doi.org/10.3390/en13102509

[12] Li, Y., & McClelland, J. L. (2022). Systematic generalization and emergent structures in transformers trained on structured tasks. *arXiv Preprint: 2210.00400*. https://doi.org/10.48550/arXiv.2210.00400

[13] Stefanova, Z. S., & Ramachandran, K. M. (2018). Off-policy Q-learning technique for intrusion response in network security. *World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering*, *12*(4), 266–272. https://publications.waset.org/10008916.pdf

[14] Aljehane, N. O., Mengash, H. A., Hassine, S. B. H., Alotaibi, F. A., Salama, A. S., & Abdelbagi, S. (2024).

Optimizing intrusion detection using intelligent feature selection with machine learning model. *Alexandria Engineering Journal*, *91*, 39–49. https://doi.org/10.1016/j.aej.2024.01.073

[15] Hu, B., & Li, J. (2021). Shifting deep reinforcement learning algorithm toward training directly in transient real-world environment: A case study in powertrain control. *IEEE Transactions on Industrial Informatics*, *17*(12), 8198–8206. https://doi.org/10.1109/TII.2021.3063489

[16] Sethi, K., Madhav, Y. V., Kumar, R., & Bera, P. (2021). Attention based multi-agent intrusion detection systems using reinforcement learning. *Journal of Information Security and Applications*, *61*, 102923. https://doi.org/10.1016/j.jisa.2021.102923

[17] Nguyen, T. T., & Reddi, V. J. (2023). Deep reinforcement learning for cyber security. *IEEE Transactions on Neural Networks and Learning Systems*, *34*(8), 3779–3795. https://doi.org/10.1109/TNNLS.2021.3121870

[18] Caminero, G., Lopez-Martin, M., & Carro, B. (2019). Adversarial environment reinforcement learning algorithm for intrusion detection. *Computer Networks*, *159*, 96–109. https://doi.org/10.1016/j.comnet.2019.05.013

[19] Lopez-Martin, M., Carro, B., & Sanchez-Esguevillas, A. (2020). Application of deep reinforcement learning to intrusion detection for supervised problems. *Expert Systems with Applications*, *141*, 112963. https://doi.org/10.1016/j.eswa.2019.112963

[20] Iannucci, S., Cardellini, V., Barba, O. D., & Banicescu, I. (2020). A hybrid model-free approach for the near-optimal intrusion response control of non-stationary systems. *Future Generation Computer Systems*, *109*, 111–124. https://doi.org/10.1016/j.future.2020.03.018

[21] 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

[22] Shao, J.-M., Zeng, G.-Q., Lu, K.-D., Geng, G.-G., & Weng, J. (2024). Automated federated learning for intrusion detection of industrial control systems based on evolutionary neural architecture search. *Computers & Security*, *143*, 103910. https://doi.org/10.1016/j.cose.2024.103910

[23] Du, X., Xu, C., Li, L., & Li, X. (2024). Multigranularity feature automatic marking-based deep learning for anomaly detection of industrial control systems. *IEEE Open Journal of Instrumentation and Measurement*, *3*, 1–10. https://doi.org/10.1109/OJIM.2024.3418466

[24] Wang, X., Wang, Y., Javaheri, Z., Almutairi, L., Moghadamnejad, N., & Younes, O. S. (2023). Federated deep learning for anomaly detection in the internet of things. *Computers and Electrical Engineering*, *108*, 108651. https://doi.org/10.1016/j.compeleceng.2023.108651

[25] Rahman, F., Tanna, R. Z., Habiba, U., Shaikh, R., Rahman, Z., & Imtiaz, H. (2022). Cyber threat detection using machine learning algorithms on heterogeneous MiniVHS-22 dataset. In *2022 25th International Conference on Computer and Information Technology*, 866–869. https://doi.org/10.1109/ICCIT57492.2022.10055036

How to Cite: Abdul Ghani, N., Kuck, S. N., Farid, F. A., Hendradi, R., Mohd Danuri, M. S. N., Ahmad, M., …, & Abdul Karim, H. B. (2026). DQLearn-Capable: A Comprehensive Intrusion Detection and Prevention System for Network Traffic Analysis using Deep Q-Learning. *Journal of Computational and Cognitive Engineering*. https://doi.org/10.47852/bonviewJCCE62027179