Online-Journals.org

iJOE International Journal of Online and Biomedical Engineering

iJOE | eISSN: 2626-8493 | Vol. 20 No. 5 (2024) | OPEN ACCESS

https://doi.org/10.3991/ijoe.v20i05.48229

PAPER

Deep Reinforcement Learning Approach for Cyberattack Detection

Imad Tareq[1](✉), Bassant Mohamed Elbagoury[1,2], Salsabil Amin El-Regaily[1], El-Sayed M. El-Horbaty[1]

[1]Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt

[2]Faculty of Computer Science and Computer Engineering, King Salman International University, El-Tor, Egypt

emadtariq1982@gmail.com

ABSTRACT

Recently, there has been a growing concern regarding the detrimental effects of cyberattacks on both infrastructure and users. Conventional safety measures, such as encryption, firewalls, and intrusion detection, are inadequate to safeguard cyber systems against emerging and evolving threats. To address this issue, researchers have turned to reinforcement learning (RL) as a potential solution for complex decision-making problems in cybersecurity. However, the application of RL faces various obstacles, including a lack of suitable training data, dynamic attack scenarios, and challenges in modeling real-world complexities. This paper suggests applying deep reinforcement learning (DRL), a deep framework, to simulate malicious cyberattacks and enhance cybersecurity. Our framework utilizes an agent-based model that is capable of continuous learning and adaptation within a dynamic network security environment. The agent determines the most optimal course of action based on the network's state and the corresponding rewards received for its decisions. We present the outcomes of our experimentation with the application of DRL on a specific model, double deep Q-network (DDQN), utilizing policy gradient (PG) on three distinct datasets: NSL-KDD, CIC-IDS-2018, and AWID. Our research demonstrates that DRL can effectively improve cyberattack detection outcomes through our model and specific parameter adjustments.

KEYWORDS

network security, deep reinforcement learning (DRL), cyberattacks, cyber defense, double deep Q-network (DDQN), policy gradient (PG)

1 INTRODUCTION

With recent breakthroughs in artificial intelligence (AI), both defensive and offensive parties have started to utilize AI techniques, especially machine learning (ML) methodologies. As technology advances and the complexity of systems increases, cybersecurity defenders must constantly adapt their strategies and tactics. Sophisticated attackers use ML approaches to identify vulnerabilities, evade detection, and amplify the impact of their attacks. Defenders use adaptive ML approaches

Tareq, I., Elbagoury, B.M., El-Regaily, S.A., El-Horbaty, E.-S.M. (2024). Deep Reinforcement Learning Approach for Cyberattack Detection. *International Journal of Online and Biomedical Engineering (iJOE)*, 20(5), pp. 15–30. https://doi.org/10.3991/ijoe.v20i05.48229

Article submitted 2023-12-08. Revision uploaded 2024-01-27. Final acceptance 2024-01-28.

to prevent and minimize the impact of threats or damages. Defenders have widely employed supervised and unsupervised learning approaches for intrusion detection, spam filtering, zero-day vulnerability forecasts, and malware detection [1, 2]. Nevertheless, typical ML approaches cannot provide sequential and dynamic responses to cyberattacks with unidentified patterns and rapidly changing behaviors. In the real world, creating autonomous cyber system protection plans and recommending actions is a challenging task. Providing decision support for cyber system security mechanisms requires integrating the dynamics between attackers and defenders and characterizing the uncertainty in the system state dynamically. To discover a solution to this issue, reinforcement learning (RL), a branch of ML, is a learning model that can learn from past experiences by exploring and exploiting changing and unfamiliar environments. RL provides ample resources for defenders to execute optimal sequential actions with minimal prior information about the environment or the attacker. RL techniques enable the defender to capture various protective and defensive activities, whether discrete or in high-dimensional continuous state space, and diverse system states. RL fits cyberspace well, where cyberattacks are becoming complex and rapidly spreading [3, 4]. In the last few years, the advancement of deep learning (DL) has spurred the effective integration of DRL. DRL aims to utilize a neural network to estimate complex functions from high-dimensional inputs. Integrating DL improves traditional RL approaches for capturing the vast scale of various network-connected systems, such as wireless networks and Internet of Things devices [5]. We utilized the DRL techniques, double deep Q-network (DDQN), and policy gradient (PG) to detect intrusions on the CIC-IDS-2018, AWID, and NSL-KDD datasets. We compare the results with previous studies, considering various standard criteria such as precision, accuracy, F1 score, and recall. The comparison findings reveal that our proposed framework, which leverages the DDQN technique, outperforms modern and comparable models. The major contributions of this study are to highlight these benefits and present them as a viable alternative to traditional ML models. Among the perks are: (1) the neural networks used for constructing the classifier: value, policy, and Q-functions are simple and fast, making them ideal for new high-demand networks; (2) the reward function used for detection is highly flexible and does not need to be differentiable; (3) the developed neural network (NN) can be deployed on modern high-performance distributed systems; and (4) we provide intrinsic insights into the optimal methods for adjusting various hyperparameters of deep reinforcement learning techniques (e.g., decay rate, learning rates, discount factor, etc.) to enhance network cyberattack detection tasks; (5) the model's design facilitates easy parameter updates. The paper is structured as follows: Section 2 discusses related works. Section 3 describes the dataset used in the tests. Section 4 provides a detailed description of the suggested model. Section 5 presents the gathered results, while Section 6 provides conclusions.

2 RELATED WORK

Q-learning is a model-free technique that has received acclaim as a viable solution, particularly in complex decision-making processes. The benefits of Q-learning encompass its ability to achieve favorable outcomes, its aptitude for learning, and the possibility of interaction with other models. Numerous research endeavors have delved into the application of machine learning, including deep reinforcement learning (DRL), in cybersecurity. The researchers in [6] examined the application of multiple DRL algorithms, such as DDQN, deep Q-network, actor-critic, and PG,

in intrusion detection using the AWID and NSL-KDD datasets. [7] The researchers proposed an approach for network intrusion detection that blends RL with a deep feed-forward neural network. The model can autonomously learn in a network environment and identify various intrusions using an automated trial-and-error technique. The empirical findings presented in this paper are based on the NSL-KDD dataset. In [8], the authors introduce a GAN framework augmented with DRL to investigate the generation of semantically coherent samples. A DRL agent is used to challenge the GAN's discriminator, which functions as a botnet detector. The discriminator is trained on the perturbations intentionally created by the agent during the GAN training process. Utilizes the GAN generator to achieve convergence faster than in scenarios where DRL is not employed. In order to conduct experiments, three distinct botnet datasets, namely CIC-2017, ISCX-2014, and CIC-2018, are utilized. In [9], the authors compare neural episodic control to double-deep Q-networks (DQNs) and DQNs for cyber security in software-defined networks. The results suggest that both algorithms are effective network defense methods. Given the lack of significant differences between the two techniques, DDQN is preferred due to its simplicity. The authors of [10] focus on applying DRL to enhance cybersecurity defense strategies against strategic multiple-stage attackers. A DRL defense agent aims to compute context-aware defensive strategies by learning network and multi-stage attack patterns while minimizing effects on benign system operations. The Dragon DRL method developed by the authors in [11] aims to enhance autonomous grid operation and attack detection capabilities by effectively managing power operations and automatically recognizing cyberattacks. To evaluate the performance of DRAGON, the researchers conducted simulations of various attack scenarios on the IEEE 14-bus power transmission system paradigm.

3 DATASET DESCRIPTIONS

3.1 NSL_KDD datasets

The standard KDD99 network traffic dataset contains duplicated data that can negatively impact the effectiveness of the training model. To tackle this challenge, Tavallaee et al. introduced the NSL-KDD dataset in 2009 [12]. The dataset eliminates superfluous information from the original dataset and adjusts the data quantity in the testing and training sets compared to the authentic dataset, improving the rationality of the KDD dataset and enhancing model training performance. The training dataset consists of 125973 records, while the test dataset contains 22544 records and 41 features that can be classified as either attacks or normal. The dataset contains four categories of attacks: U2R, R2L, DOS, and probe attacks. In this paper, we employ DDQN models on the NSL-KDD dataset to acquire essential performance metrics, including accuracy, recall, precision, and F1, for attack detection. All the findings are based on the training dataset, which consists of 122550 records, and the test dataset, which consists of 22544 records [13].

3.2 AWID datasets

The Aegean wi-fi intrusion dataset (AWID) is freely available, containing normal network traffic and three attacks on IEEE 802.11 networks. Among the various datasets offered by AWID [14], we have selected the AWID-CLS-R dataset, which provides

distinct training and test datasets. This dataset includes four labels for classification: normal, injection, flooding, and impersonation. It contains 154 features, 1795574 samples for training, and 575642 samples for testing. We have reduced the number of features to 24 by removing those with null or constant values, as well as features with sample-specific network addresses that do not apply to the test data. The continuous features have been standardized to a range of 0 to 1, and the categorical features have been encoded. It is vital to note that this dataset is highly imbalanced, with 91% of the samples being normal and only 9% associated with anomalies, such as 2.7% flooding, 2.7% impersonation, and 3.6% injection. The level of imbalance in the AWID dataset is even greater than that of the NSL-KDD dataset, although the label distribution in the dataset is similar for both the testing and training datasets. Hence, the AWID and NSL-KDD datasets present distinct challenges for classification algorithms. In this paper, we have applied the DDQN model to the AWID dataset and evaluated its effectiveness based on key measures such as F1 score, accuracy, precision, and recall, specifically for detecting attacks. All the findings presented in this study are based on the training dataset, which includes 1339406 samples, and the test dataset, which includes 389185 samples.

3.3 CSE-CIC-IDS2018 datasets

The University of New Brunswick compiled a dataset for analyzing DDoS data, covering seven distinct assault scenarios: Web attacks, DDoS, Heartbleed, brute-force attacks, DoS, penetration, and botnet attacks. To simulate these assaults using 50 machines, the target organization comprised five departments, 420 workstations, and 30 servers. The dataset itself encompasses 16000,000 instances collected over ten days. In terms of content, it includes 80 characteristics derived from captured traffic using CICFlowMeter-V3 and the system logs of each computer and network traffic [15]. In this study, DDQN models are applied to the CSE-CIC-IDS dataset to assess key performance measures such as accuracy, F1, recall, and precision for detecting two label values: normal and anomalous. The results are based on a training dataset comprising 6311436 samples and a test dataset comprising 1577859 samples.

4 MODEL DESCRIPTION

This section offers an overview of the DRL model examined in this study. Deep learning models, such as neural networks, use approximators to represent the value functions and policies in reinforcement learning. The foundation of reinforcement learning lies in the Markov decision process (MDP) theory, which includes various components: a collection of states (s), a collection of possible actions (A) that the agent can take in the environment, a transition function (T) that governs the movement from one state to another, and a reward function (R) that assigns a value to every state-action pair. The transition function (T) represents the probability distribution of transitioning to a new state (s_0) from the current state (s). The reward function (R) provides an absolute value for every state-action pair, and (γ) is the discount factor value between 0 and 1 that indicates the significance of future rewards. The transition function (T) satisfies the Markov property in an MDP, indicating that the probability of transitioning to a new state is solely determined by the action and current state, regardless of past events. Once an MDP is specified, a policy is established to

map every state to an action in order to discover the optimal policy that maximizes the expected total rewards. The optimality requirement can be measured by simply summing the rewards, taking an average, or applying a discount factor to prioritize immediate rewards. An MDP provides a theoretical framework for an agent to interact with its environment and make sequential decisions. The environment carries out the transition and reward functions while the agent executes the policy. The interaction between the environment and the agent is typically divided into discrete time steps. The agent takes an action, which leads to a change in state and potentially receiving a reward. The connection between policy and optimality criteria is established by creating a value function, which estimates the value associated with each state. The value function represents the advantages of being in a particular state, assuming that the current policies will be followed. Two types of value functions can be employed: the V-function, which assigns a value to every state, and the Q-function, which assigns a value to each state-action pair. The Q-function is calculated by adding the total reward for a specific state-action combination and the value of the V-function for the subsequent state generated by the environment. In this scenario, we acquire knowledge of the most efficient policy directly. We can determine the transition probabilities from one state to another using policy gradient-based strategies. Policy gradient techniques utilize descent to maximize the predicted total of discounted rewards for a given policy, thereby enabling the direct learning of a policy function. To achieve an optimal policy, thoroughly exploring as much of the state action space as possible is crucial. We use the epsilon-greedy approach to select an optimal action with a probability of p and a random action with a probability of $1 - p$. On the other hand, action probability exploration involves the policy providing a probability for each action, enabling sampling based on this distribution. It can be inferred that policy functions or learning values involve numerous iterative processes, with each iteration resulting in a modification of the function. DRL approaches utilize function approximators based on various types of neural networks.

4.1 Proposed models

In the realm of RL, DDQN outperforms DQN. It addresses a critical issue with DQN: overestimating action values, which can lead to poor decisions. The issue with DQN is that it estimates the Q-values (anticipated future rewards) for each action and state using a single neural network. This network determines the next action and updates the Q-values based on previous experiences. This connection can result in an overestimation bias, where the network overestimates Q-values, especially for infrequently selected actions. This may cause the agent to exhibit poor behavior more frequently. DDQN employs two distinct networks: The main network calculates the Q-values for all actions in a state. The target network gradually updates its settings from the main network and functions as a "frozen" version. Action selection and estimation are separate processes. The main network selects the action with the greatest Q-value in the current state as the next action. However, the action is decided using the target network to update the Q-values of the main network. This removes the update process's direct effect on potentially inflated Q-values. Decreased overestimation bias: This results in a more accurate Q-value estimate and perhaps greater performance, reducing the risk of "catastrophic forgetting," where the agent might forget previously learned optimal actions.

4.2 Models detail

This section comprehensively explains the DRL model used for this investigation. The agent interacts with its surroundings by taking actions, monitoring rewards, and predicting future states. A replay buffer stores these experiences (state, action, reward, and future state). The buffer enables us to randomly select batches of experiences for learning, ensuring that recent events do not unduly influence the agent. A Q-function identifies the predicted highest reward based on a given action and state. Consequently, $Q(s, a)$ depends on both the state and action combinations. Once the Q-function is established, we can deduce the policy function, which defines what action to perform in every state. The policy function relies on the state and is generated from the Q-function in the following way:

$$Policy(s) = arga\ max\ (Q(s, a)) \tag{1}$$

An epsilon-greedy policy is a training technique that enables the agent to explore potential actions and determine the best policy as the number of explorations grows. It involves selecting an action at random with a probability of ε or predicting it with a probability of $(1 - \varepsilon)$. The policy π establishes a probability distribution for selecting different actions based on every states. Once the fixed policy is chosen, the distribution of the reward sequence is determined. The policy is evaluated using the action value function, which computes the potential cumulative adversarial reward obtained by acting in a specific state and following the policy. An MDP solution is employed to derive the optimal policy (π^*), maximizing the potential lower reward across all states. The following equation elucidates the unique characteristics and existence of the fixed-point solution to the ideal Bellman equation.

$$Q^*(a, s) + R\ (a, s) + \gamma \int_{s'} T\ (S'/s, a)\ maxQ^*\ (a', s') \tag{2}$$

The action effectively optimizes the Q-value, thereby enhancing its performance. The method begins with a standard sample that includes the current state (s_t), the correct label for that state (a_t), and the subsequent state (s_{t+1}). This n sample is just one of the many samples in a mini-batch, which is a subset of randomly chosen samples from the dataset. Every training iteration uses a different mini-batch, randomly sampled from the dataset. Before starting the procedure, every mini-batch is created by randomly rearranging the dataset and picking $(n + 1)$ using a random index (t). During each training iteration, the samples will be processed in a mini-batch and a new mini-batch will be created. The Q-function is approximated using a neural network (NN) as a function approximator. We utilized three hidden layers. Each layer contains Dense (neurons = 124, activation = "ReLU"), Dense (filters = 124, activation = "ReLU"), and Dense D (filters = 124, activation = "ReLU"). The activation function of the output layer is designed to be linear to achieve a positive Q-value. The NN is trained using a Huber loss function, which calculates the difference between the Q-value predicted by the NN for the current state ($\hat{q}_t$) and a reference value, q_{ref}. This reference value is calculated by adding the current reward (r_t) to the Q-value for the following state ($\hat{q}_{t+1}$), which is then multiplied by a discount factor (λ). A reward of 1 or 0 is associated with a correct or incorrect prediction, respectively. The label value for the current state is (a_t^*), while the anticipated value is represented as ($\hat{a}_t$). If the values are equal, the reward is 1; otherwise, it is 0. To determine the expected value for the current state ($\hat{a}_t$), the Q-function is iterated with the current state (s_t),

and all possible label values ({a}). This iterative process is denoted as $Q(s_t, \{a\})$, a vector of values visually depicted as a bold arrow in Figure 1.

$$Q(s_t, \{a\}) = [Q(s_t, \{a\}0), Q(s_t, \{a\}1), \ldots Q(s_t, \{a\}p)] \quad (3)$$

The set {a} represents all potential actions, and its cardinality is indicated by p. The action Q-value with the best result is chosen from the iteration argamax (Q (s_t, {a})). This selected action is then inputted into the ε-greedy algorithm, which determines whether to choose that value with a certain probability p or a random action with another probability 1 – p. The result of the final stage is the expected action ($\hat{a}_t$). Similarly, the projected action for the following state ($\hat{a}_{t+1}$) is determined without considering the ε-greedy selection. The predicted action and the subsequent state s_{t+1} are used to calculate q_{ref} by obtaining the Q-value for the next state ($\hat{q}_{t+1}$). The stated forecast of the Q-value for the following state ($\hat{q}_{t+1}$) can be reduced to a certain extent $(\hat{q}_{t+1}) = \max a\, Q(s_{t+1}, \{a\})$.

Once the model is trained, the two neural networks in DDQN are utilized: one for implementing a current Q-function and the other for constructing a goal Q-function. The target Q-function is a replica of the current Q-function with synchronization delays, indicating that the copy is created after a specific number of training repetitions. The Q-value for the following state ($\hat{q}_{t+1}$) is calculated using the target Q-function. This additional Q-function, known as the target Q-function, aims to prevent the shifting target effect while performing gradient descent. $(\hat{q}_{t+1} - q_{ref})^2$ Consequently, it eliminates the recursive reliance of q_{ref} on the training network (Q-current). Aside from including the target Q-function, the model was trained for 100 epochs, where each epoch corresponds to the repetitions required to cover the entire dataset.

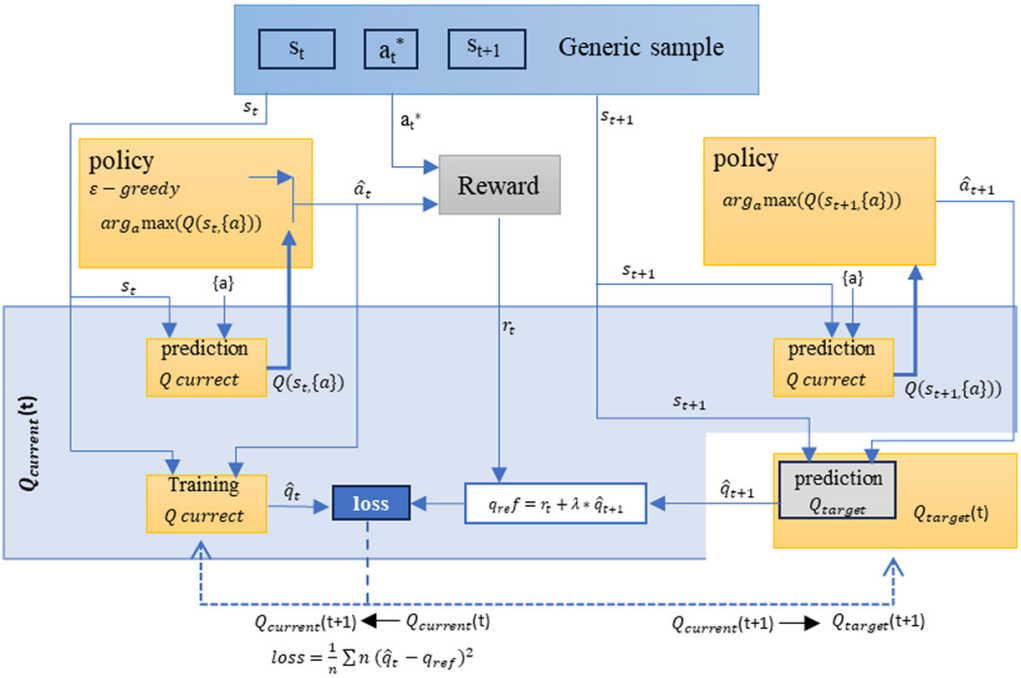


Fig. 1. The DDQN model [16]

The DDQN agent controls the environment where the datasets are utilized. The initial phase involves initializing the algorithm and the parameters of the models.

The values of the dataset features (F1–Fm) represent the DDQN state variables (*s*). It is worth noting that the batch size (bs) for the DDQN procedure is 100. This implies that for each state, 100 dataset records are acquired from memory and fed into a single state (*s*). Nevertheless, numerous state variables exist, each of which may possess a distinct value. As the number of state-value pairs increases, it becomes challenging to maintain them in a Q-table. Consequently, the DDQN agent uses a neural network as a function approximator to calculate Q values based on actions and states. The training sample (Batch *n*) provides the state (s_n) at every discrete state, as depicted in Figure 2. In the concluding state, a comprehensive sequence of actions, states, and rewards is obtained at the end of each episode. The agents receive the initial batch (100 data points) throughout the training process, corresponding to the environment's initial state (s_1). To ensure a positive Q-value, we utilize a deep neural network consisting of three layers, as depicted in Figure 2, with ReLU activation for all layers, including the final one.

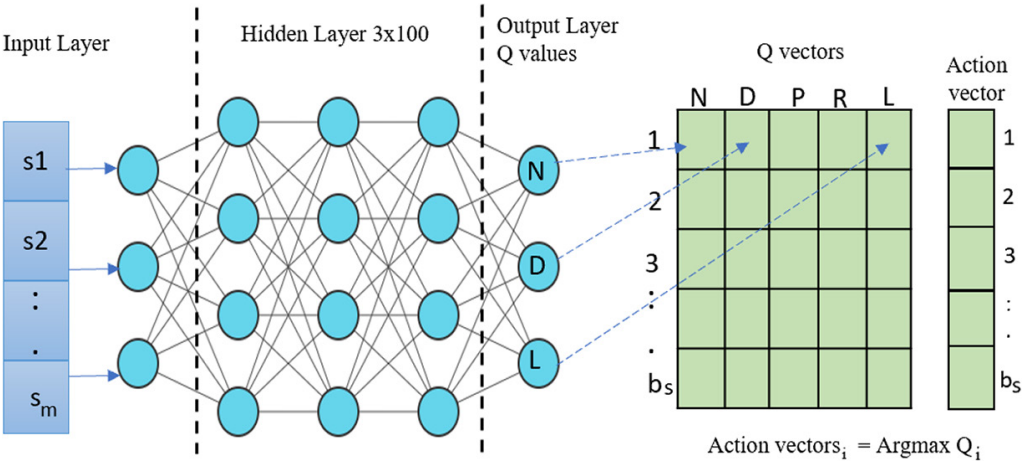


Fig. 2. The double deep Q-network (DDQN) model, which produces predictions from a deep neural network and states. The model produces Q-values, generating actions using the argmax of the current state's Q-values

5 EXPERIMENTAL RESULTS

In our tests, the proposed model based on DRL: DDNQ was applied to three datasets: NSL-KDD, CIC-IDS-2018, and AWID. It is important to highlight that we utilized the model with both two and three layers and conducted a comparison between them. We present the following performance measures to examine the prediction performance of the different models: F1 score, accuracy, precision, and recall. We base our definitions of these performance measures on widely acknowledged standards. We will give greater significance to the accuracy and F1 score due to the imbalanced nature of the datasets. To present the findings for the three datasets, we will begin with the NSL-KDD and AWID datasets and then proceed to the CIC-IDS-2018 dataset.

5.1 Performance metrics

We evaluate the efficacy of the DDQN model used to detect network attacks by employing various metrics such as accuracy, F1 score, recall, and precision.

However, since it assesses the proportions of accurately identified samples, the model's performance cannot be solely determined by the accuracy values. It disregards the misclassified samples. These metrics were established using true positive (TP), true negative (TN), false positive (FP), and false negative (FN) data [17]. The misidentified valid and attack vectors were classified as FP and FN, respectively. The TN and TP represent the number of genuine attack vectors that were accurately classified.

Table 1. Neural Network parameters and DDQN agent

Parameters	Values	Parameters	Values
Number – episode	100	Minimum epsilon	0.01
Hidden_layers	3 or 2	gamma	0.001
Number – iteration	100	Decoy rate	0.99
Number_units	3 × 124	Learning_rate	0.001
Activation function	ReLU	Batch–size	100
Initial weight value	Normal	Optimizer	Adam
epsilon	1		

Accuracy: The metric of accuracy assesses the total count of accurate predictions generated by the model with respect to the overall number of predictions made. Accuracy metric can be calculated as shown in Equation (4).

$$\text{Accuracy} = \frac{TN + TP}{TN + FN + TP + FP} \times 100 \tag{4}$$

Precision is the measure of the percentage of positive cases in relation to the overall projected positive cases. It indicates the model's accuracy by determining its accuracy, as defined in Equation (5).

$$\text{Precision} = \frac{TP + TN}{TP + FP} \times 100 \tag{5}$$

Recall: Ratio overall number of positive instances. This metric represents the number of correct examples the model disregarded when presenting correct instances, as illustrated in Equation (6).

$$\text{Recall} = \frac{TP}{TP + FN} \times 100 \tag{6}$$

F1 score: The performance metric is calculated by averaging the accuracy and recall scores. It considers the contributions of both values. Equation (7) illustrates that the F1 score is derived from recall and precision values.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Recall + Precision} \tag{7}$$

5.2 Results for NSL-KDD datasets

Demonstrating the suitability of DRL models for cyberattack detection in networking is a significant contribution to this work. Figure 3 presents the outcomes

obtained from the investigation of DRL models using the NSL-KDD dataset. To ensure consistent percentages across all classes, we divided the dataset into 20% for testing and 80% for training, with the option of stratifying. 122550 samples were included in the training set, while 22544 samples were allocated to the test set. These samples were classified into five categories, encompassing 162 features. The findings of this study revealed that the 3-layer model produced favorable results. Specifically, Figure 3 displays an accuracy rate of 82.40%, an F1 score of 81.26%, a precision rate of 81.35%, and a recall rate of 82.40%. Conversely, the 2-layer model achieved an accuracy rate of 81.87%, an F1 score of 81.29%, a precision rate of 81.34%, and a recall rate of 81.87%. These findings confirm the model's effectiveness in predicting attacks across the five classes of the confusion matrix depicted in Figure 7.

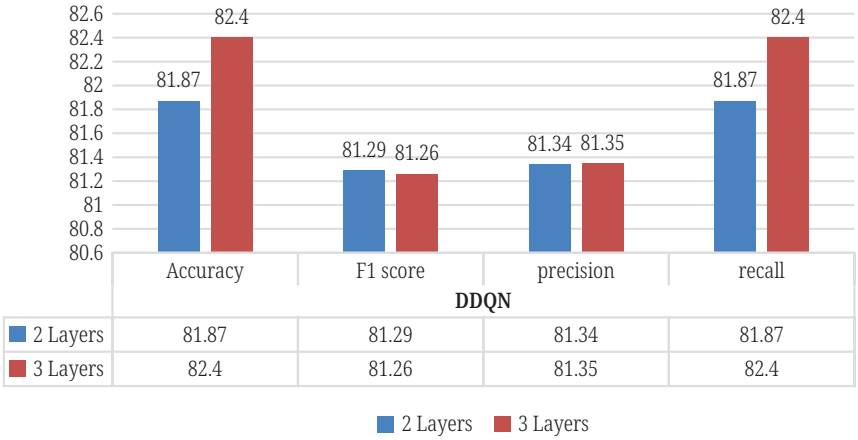


Fig. 3. The NSL-KDD dataset's DDQN model outcomes

The NSL-KDD dataset presents various challenges to classifiers, particularly due to the composition of the training and test sets. The metrics serve as the primary performance indicators for an attack detection system that aims to accurately identify as many attacks as possible.

Table 2. Evaluation metrics of the DDQN model using the NSL-KDD dataset, broken down by class

Metric	Attack Categories				
	Normal	DoS	R2L	Probe	U2R
Accuracy	87.61	94.77	87.94	95.63	98.81
F1-score	86.85	91.77	39.12	79.28	21.17
Precision	80.02	95.88	51.05	80.93	25.71
Recall	94.96	87.99	31.71	77.69	20.33

5.3 Results for AWID database

The current investigation aimed to assess the efficacy of the RL model in detecting cyberattacks by utilizing the AWID database. The database comprises 1339406 training data, 389,185 test data, and 67 features, ensuring the reliability of the assessment.

The dataset was divided into 20% for testing and 80% for training, with stratification applied to ensure consistent percentages across all classes. Multiple classification measures were utilized, including accuracy, F1-score, precision, and recall. The 3-layer model achieved a top accuracy of 94.64%, a precision of 96.20%, an F1-score of 95.11%, and a recall of 94.64%. The 2-layer model, on the other hand, achieved an accuracy of 89.99%, an F1-score of 89.73%, a precision of 89.53%, and a recall of 89.99%. The performance of the models is depicted in Figure 4, which shows the accurate prediction percentages for the four classes of the confusion matrix presented in Figure 7.

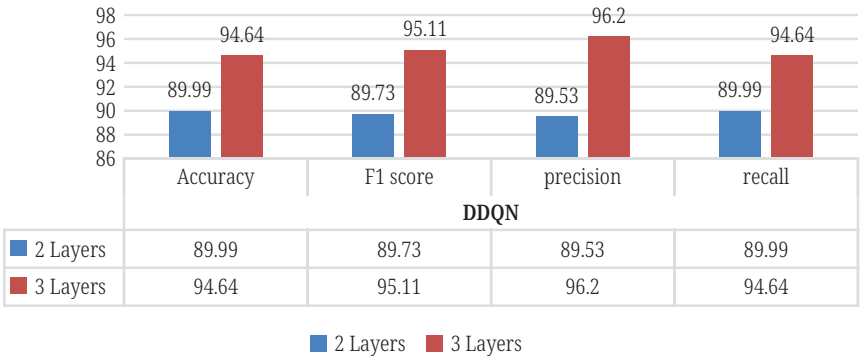


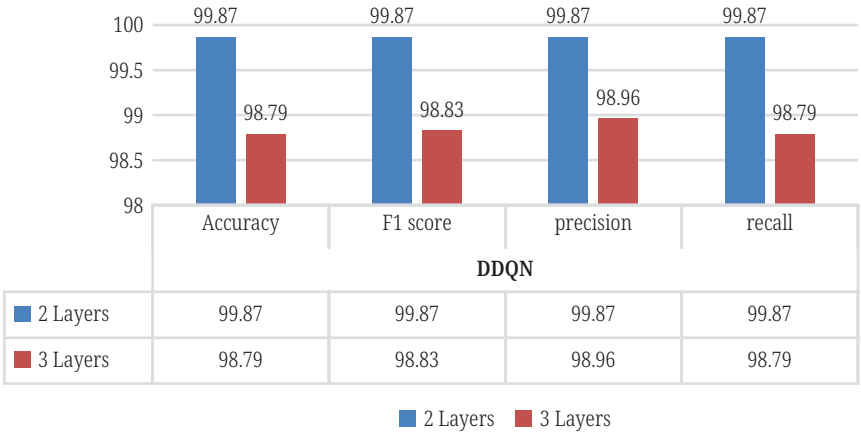
Fig. 4. Displays the AWID dataset's DDQN model outcomes

Table 3. Presents the evaluation metrics of the DDQN model using the AWID dataset, broken down by class

Metric	Attack Categories			
	Normal	Impersonation	Injection	Flooding
Accuracy	94.67	95.84	99.29	99.46
F1-score	96.94	69.69	92.36	82.67
Precision	99.26	55.84	85.81	83.83
Recall	94.72	92.66	99.99	81.54

5.4 Results for CIC_IDS database

The investigation explored the outcomes of the DRL model when applied to the CIC_IDS dataset on the fourth day of the dataset, which occurred on February 20, 2018. This dataset was partitioned into a testing set, which comprised 20% of the data, and a training set, which comprised 80% of the data. The option of stratifying the dataset was available to ensure consistent percentages across all classes. The dataset consisted of 6311436 samples for the training set and 1577859 samples for the test set, encompassing 37 features categorized into two groups: Benign and DDoS. The findings of the investigation demonstrated that the model produced positive results. As depicted in Figure 5, accuracy results revealed a 3-layer model with an accuracy of 98.79%, a precision of 98.96%, an F1 score of 98.83%, and a recall of 98.79%. Furthermore, the highest accuracy achieved with a 2-layer model was 99.87%, along with an F1 score of 99.87%, a precision of 99.87%, and a recall of 99.87%. These results represent the percentage of correctly predicted attacks for the two classes of the confusion matrix, as illustrated in Figure 7.

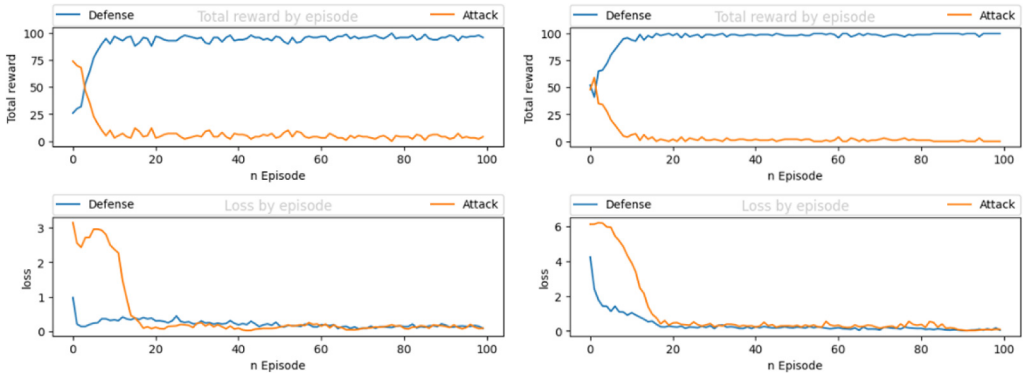


	Accuracy	F1 score	precision	recall
2 Layers	99.87	99.87	99.87	99.87
3 Layers	98.79	98.83	98.96	98.79

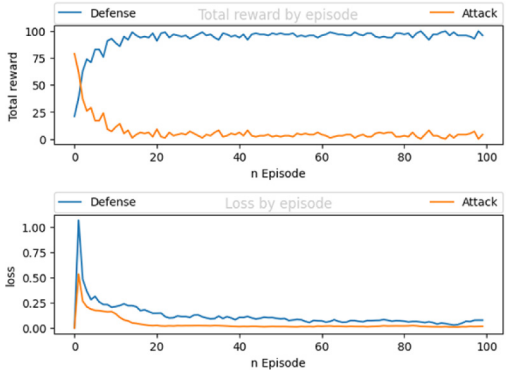
Fig. 5. CIC_IDS dataset's DDQN model outcome

Table 4. The evaluation metrics of the DDQN model using the CIC_IDS dataset, broken down by class

Metric	Attack Categories	
	Benign	DDoS
Accuracy	98.78	98.78
F1-score	99.34	92.34
Precision	99.99	85.79
Recall	98.69	99.97



(NSL-KDD Datasets) **(CIC-IDS-2018 Datasets)**



(AWID Datasets)

Fig. 6. Environment training to reward and loss values achieved during DDQN training

The RL consists of states, actions, and rewards in the environment during training. In other words, rewards are obtained when moving from state S1 to state S2 with an action. When the defender counters against the attacker, the action is correct, and the reward result is high.

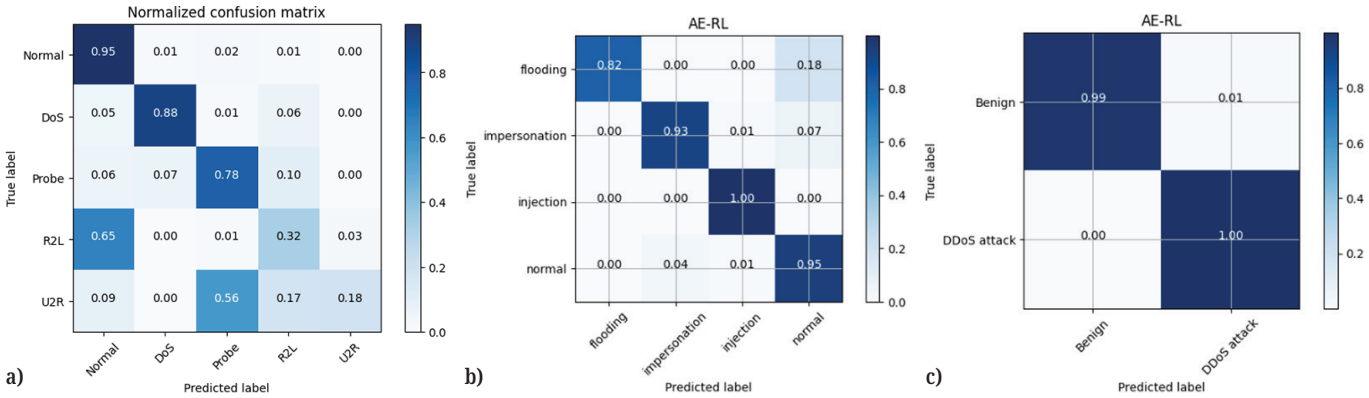


Fig. 7. The confusion matrix in the DDQN model using the (a) NSL-KDD dataset validation of five classes: normal, R2L, DOS, U2R, and probe attacks, all values between 0.95 and 0.18. (b) AWID dataset validation of four classes: normal, flooding, impersonation, and injection, all values between 0.95 and 0.82. (c) CIC_IDS dataset validation of two classes: Benign and DDoS, all values between 0.99 and 100

Table 5 provides a comparison between the RL models and the earlier research in identifying and categorizing various attacks across the CIC-IDS-2018, NSL-KDD, and AWID databases.

Table 5. Comparison of the results with the prior studies utilizing the three datasets (CIC-IDS-2018, NSL-KDD, and AWID)

Research	Method	Datasets	Acc	F1	Pre	Rec	Features
Lopez-Martin [5]	DDQN	NSL-KDD	89.78	91.20	89.44	93.03	122
		AWID	95.70	93.94	92.35	95.70	24
Alavizadeh, H [6]	DQL	NSL-KDD	78.07	81.41	77.84	76.76	41
Caminero, G [18]	AE-RL	NSL-KDD	80.16	79.40	79.74	80.16	41
		AWID	95.90	96.26	97.20	95.90	24
Dong, S [19]	DDQN	AWID	96.47	96.73	97.40		49
		NSL-KDD	73.43	69.02	66.61		122
Ren, K [20]	DQN	CIC_IDS	94.11	92.51			13
Our proposal	DDQN	NSL-KDD	82.40	81.26	81.35	82.40	162
		AWID	94.64	95.11	96.20	94.64	67
		CIC_IDS	98.79	98.83	98.96	98.79	37

6 CONCLUSION

We introduce a reinforcement learning model, the DDQN, to detect and categorize various classes of network cyberattacks. The model is verified using three datasets (NSL-KDD, CIC-IDS-2018, and AWID) and employs DQNs to establish a DRL strategy. The DNN is integrated with reinforcement learning to interact with the

network environment. Network traffic is captured and analyzed to detect malicious network payloads using the autonomous behavior of DDQN agents. Additionally, DDQN, along with other DRL methods, offers the advantage of significantly reducing prediction times, making them highly suitable for online detection and the requirements of modern network services. Furthermore, to enhance learning capabilities, we thoroughly analyze various parameters of the DDQN agent, such as the discount factor, the number of learning episodes, and batch size, to identify optimal fine-tuning strategies for network cyberattack detection tasks. Our experimental findings demonstrate the effective learning capacity of the proposed DDQN model, as it can autonomously classify different types of network attacks with a high level of accuracy. As part of our future work, we plan to implement our proposed solution in an actual cloud environment. This deployment will enable the DDQN agent to enhance its self-learning capabilities and achieve accurate threat classification in real-time scenarios. Additionally, we plan to apply the DDQN model to detect ransomware to assess its generalizability and practicality.

7 ACKNOWLEDGMENTS

We thank Eng. Mohamed Ahmed for his great efforts in coding this research.

8 REFERENCES

[1] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "IoT security techniques based on machine learning: How do iot devices use ai to enhance security?" *IEEE Signal Processing Magazine*, vol. 35, no. 5, pp. 41–49, 2018. https://doi.org/10.1109/MSP.2018.2825478

[2] S. A. Jebur, K. A. Hussein, H. K. Hoomod, L. Alzubaidi, and J. Santamaría, "Review on deep learning approaches for anomaly event detection in video surveillance," *Electronics*, vol. 12, no. 1, p. 29, 2022. https://doi.org/10.3390/electronics12010029

[3] A. Uprety and D. B. Rawat, "Reinforcement learning for IOT security: A comprehensive survey," *IEEE Internet of Things Journal*, vol. 26, no. 8, pp. 8693–8706, 2021. https://doi.org/10.1109/JIOT.2020.3040957

[4] L. R. Ali, S. A. Jebur, M. M. Jahefer, and B. N. Shaker, "Employing transfer learning for diagnosing COVID-19 disease," *Int. J. Online Biomed. Eng.*, vol. 18, no. 15, pp. 31–42, 2022. https://doi.org/10.3991/ijoe.v18i15.35761

[5] A. Ferdowsi, U. Challita, W. Saad, and N. B. Mandayam, "Robust deep reinforcement learning for security and safety in autonomous vehicle systems," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 307–312. https://doi.org/10.1109/ITSC.2018.8569635

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

[7] H. Alavizadeh, H. Alavizadeh, and J. Jang-Jaccard, "Deep Q-learning based reinforcement learning approach for network intrusion detection," *Computers*, vol. 11, no. 3, p. 41, 2022. https://doi.org/10.3390/computers11030041

[8] R. H. Randhawa, N. Aslam, M. Alauthman, M. Khalid, and H. Rafiq, "Deep reinforcement learning based evasion generative adversarial network for botnet detection," *SSRN*, 2023. https://doi.org/10.2139/ssrn.4333338

[9] L. Borchjes, C. Nyirenda, and L. Leenen, "Model-free deep reinforcement learning in software-defined networks," *arXiv preprint*, arXiv:2209.01490, 2022.

[10] A. Dutta, S. Chatterjee, A. Bhattacharya, and M. Halappanavar, "Deep reinforcement learning for cyber system defense under dynamic adversarial uncertainties," *arXiv preprint*, arXiv:2302.01595, 2023.

[11] M. Landen, K. Chung, M. Ike, S. Mackay, J. P. Watson, and W. Lee, "DRAGON: Deep reinforcement learning for autonomous grid operation and attack detection," in *Proceedings of the 38th Annual Computer Security Applications Conference*, 2022, pp. 13–27. https://doi.org/10.1145/3564625.3567969

[12] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 2009. https://doi.org/10.1109/CISDA.2009.5356528

[13] KDD'99 datasets, 2017. Available on: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[14] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 184–208, 2015. https://doi.org/10.1109/COMST.2015.2402161

[15] R. B. Basnet, R. Shash, C. Johnson, L. Walgren, and T. Doleck, "Towards detecting and classifying network intrusion traffic using deep learning frameworks," *J. Internet Serv. Inf. Secur.*, vol. 9, no. 4, pp. 1–7, 2019.

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

[17] S. A. Jebur, K. A. Hussein, H. K. Hoomod, and L. Alzubaidi, "Novel deep feature fusion framework for multi-scenario violence detection," *Computers*, vol. 12, no. 9, p. 175, 2023. https://doi.org/10.3390/computers12090175

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

[19] S. Dong, Y. Xia, and T. Peng, "Network abnormal traffic detection model based on semi-supervised deep reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4197–4212, 2021. https://doi.org/10.1109/TNSM.2021.3120804

[20] K. Ren, Y. Zeng, Z. Cao, and Y. Zhang, "ID-RDRL: A deep reinforcement learning-based feature selection intrusion detection model," *Scientific Reports*, vol. 12, no. 1, p. 15370, 2022. https://doi.org/10.1038/s41598-022-19366-3

9 AUTHORS

Imad Tareq is currently employed at the Dewan Al-Waqf Al-Sonny in Iraq, received a master's degree in computer science—Middle East University, Jordan, in 2016, where he is currently pursuing Ph.D. degree. His current research interests include cybersecurity, network and computer security, the Internet of Things, and artificial intelligence applications (E-mail: emadtariq1982@gmail.com).

Bassant Mohamed Elbagoury received Ph.D. and M.Sc. degrees in Computer Science from the Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt, in 2009 and 2005, respectively. She worked as a Ph.D. Researcher with the NAO Robot team Humboldt, Germany. Since 2003 she has participated in international conferences in Paris, Poland, Germany, Jordan, and the US. Since 2005, she has been a Reviewer in AAAI USA conferences. She is currently working as an Assistant Professor of computer science atthe Faculty of Computer

Science and Computer Engineering King Salman International University, KSIU and as Faculty of Computer and Information Sciences, at the Ain Shams University. Her research areas are robotics, artificial intelligence, mobile computing, and cloud computing (E-mail: drbassantcs@gmail.com).

Salsabil Amin El-Regaily is a Lecturer in the faculty of Computer and Information Sciences, Ain Shams University in Cairo Egypt. She got her PhD degree in Computer Science in 2019 and currently works in the Basic Sciences Department in the faculty. She is a member of the Ain Shams University Ranking team and the Deputy Director of the Assessment and Evaluation Unit at the Faculty of Computer and Information Sciences. Her professional activities include being an academic advisor for the credit hours programs, a member of the Quality Assurance unit, and a member of the Egypt Government Excellence Award team. Dr. Salsabil has presented many research papers in local and international journals and conferences related to the Image Processing field, especially Medical Imaging. Her Scopus H-Index is 3, with 103 citations to date. In 2021, she received an award from the Association of Arab Universities for the second-best PhD thesis in the Artificial Intelligence field. In 2023, she was awarded the Ain Shams Incentive Award in the advanced medical technological science field. Her areas of interest include structured programming, image processing, and deep learning (E-mail: salsabil_amin@cis.asu.edu.eg).

Professor El-Sayed M. El-Horbaty received Ph.D. (1985) in Computer Science from London University, UKhis M.Sc. (1978), and B.Sc. (1974) in Mathematics from Ain Shams University, Egypt. His has work experience as an academic in Egypt (Ain Shams University), Qatar (Qatar University), and Emirates (Emirates University, Ajman University, and ADU University). His work appeared in many journals such as: *Parallel Computing*, *International Journal of High Performance and Grid Computing*, *International Journal of Information Security*, *International Journal of Computer Communication and Information Security*, and *International Review on Computers and Software*, *Advances in Intelligent Systems and Computing*, *Inter. J. of Mobile Network Design and Innovation*, *Inter. J. of Bio-Medical Information*, and *e-Health*. His current areas of research are distributed and parallel computing, cloud computing, mobile cloud computing, edge computing, e-health computing, IoT, and optimization of computing algorithms (E-mail: shorbaty@cis.asu.edu.eg).