Advanced Cybersecurity Threat Detection and Protection Using Deep Reinforcement Learning and DDQN

Rathod Maheshwar[1], Madiraju Jagadeeshwar[2]
[1]Research Scholar, Department of CSE, Chaitanya Deemed to be University, Telangana, India
Email: ratkitsw@gmail.com
[2]Dean, Faculty of Engineering & Technology and Professor, Department of CSE, Chaitanya Deemed to be University, Telangana, India
Emil: jagadeeshwar07@gmail.com

Abstract—The rapid development of cloud, online services and digital communication platforms has brought greater exposure for organizations to high-end cyber threats. Older security tools are less effective at identifying new types of attacks, like zero-day exploits, advanced persistent threats, and adaptive malware. It presents a cybersecurity framework based on Deep Reinforcement Learning (DRL) which combines Deep Neural Networks (DNN), Deep Q-Networks (DQN) and Double Deep Q-Networks (DDQN) to enhance threat detection and adaptability of the system. The proposed framework continuously learns from network interactions, thus identifying new attack patterns without frequent manual interactions. Accuracy, precision, recall, F1 score, false positive rate and area under the receiver operating characteristic curve (AUC) are used to assess the performance of the models. The experimental results show that the overall classification performance of the DNN model is best, and the DDQN model has more stable and reliable predictions by decreasing the overestimation bias of the conventional Q-learning model. The results show that Deep Reinforcement Learning can play an important role in creating adaptive and resilient cybersecurity systems that can function effectively in a dynamic threat environment.

Index Terms—Threat detection, deep neural networks (DNN), double deep Q-network (DDQN), deep reinforcement learning (DRL)

How to cite this article: Maheshwar R, Jagadeeshwar M. Advanced cybersecurity threat detection and protection using deep reinforcement learning and DDQN. Int J Drug Deliv Technol. 2026;16(74s): 188-199. DOI: 10.25258/ijddt.16.74s.22

I. INTRODUCTION

With the growing expansion of cloud computing, internet services, digital storage devices, and interconnected networks, Cybersecurity has become a hot topic. The use of digital platforms is more and more a means for organisations and individuals to communicate, finance transactions and retrieve data, which is why it is a target for cybercriminals. Sophisticated attacks such as ransomware, phishing, malware, advanced persistent threats (APTs) and zero-day exploits are constantly maturing and often are able to outsmart traditional security strategies [1][2].

Most of the traditional IDS are based on the signature or rules approach. These techniques work well on known attacks, but have difficulties in detecting unknown and rapidly changing attacks, as they are highly dependent on the attack database. This restriction has spurred researchers to take an interest in Machine Learning (ML) and Deep Learning (DL) techniques that can automatically detect hidden patterns and suspicious activities in network traffic data [3][4][9].

In the context of cybersecurity, Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) have attracted significant attention in recent times due to their capacity to continuously engage with the environment, learn from the action outcomes, and enhance the decision-making process over the course of time [6][10]. Contrary to the conventional model, DRL model can be adapted for new scenario of attack and network changes, which satisfy the needs of modern cybersecurity applications [7][11]. A Deep Reinforcement Learning (DRL) cybersecurity framework, which integrates Deep Neural Networks (DNN), Deep Q-Networks (DQN) and Double Deep Q-Networks (DDQN) has been introduced in this work to enhance the performance of threat detection and adaptability of the system [12]. The framework is designed to identify current and future hazards, and reduce the reliance on humans. Furthermore, DDQN is proposed to mitigate the overestimation bias in the traditional Q-learning methods, which helps to ensure stability and reliability in threat detection. The dynamic cybersecurity environment is evaluated using accuracy, precision, recall, F1 score and AUC metrics for the effectiveness of the three algorithms DNN, DQN and DDQN [13][14].

Objectives Of The Study:

The main objectives of this research are as follows:

III. To construct a DRL-driven classifier capable of learning from observed traffic patterns and adapting to novel forms of cyberattacks.
III. To improve threat detection quality and computational efficiency by leveraging Deep Neural Networks and reinforcement learning techniques.
III. To deploy a Double Deep Q-Network (DDQN) to overcome the overestimation problem associated with traditional Q-learning.
III. To comprehensively evaluate DNN, DQN, and DDQN models using accuracy, precision, recall, F1-score, and AUC metrics.

II. LITERATURE REVIEW

Surveying automated post-exploitation, Maeda et al. [1] designed an advanced RL-based scheme that applies the A2C algorithm. Their experimental results that compared RL agents to goals like acquiring administrator privileges and performing a lateral traversal established

that A2C was superior to other approaches. In the case of IoT-focused security issues, the Mayfly-Optimized Regularized Extreme Learning Machine (MFO-RELM) was suggested by Alrowais et al. [2]. Following the data preprocessing, the MFO-RELM was trained on fixed benchmark sets, and state-of-the-art ML methods were added to boost classification performance. Yaseen et al. [3] assume the perspective that, as one of the sub-fields of AI, deep learning is reconfiguring how organizations are identifying and responding to cyber threats. Manoharan et al. [4] investigated the part played by AI and ML in mechanizing detection and response pipelines for next-generation defense. They particularly focus on deep-learning models trained on anomaly detection in their survey, as well as the ethical issues and the emerging group of AI-assisted adversarial attacks. Another research investigates a deep-learning approach that is specific to intrusion and APT detection in banking networks [5], which involves the integration of RNN and CNN architectures to describe intrusion behaviour in complex financial systems. A scheme proposed by Samitathan et al. [6] is an insider risk detection scheme using artificial neural network autoencoders, which they state achieves a 94.3% AUC and outperforms alternative methods that identify anomalous user behavior. Ferrag et al. [7] presented a lightweight version of BERT, SecurityBERT, that is focused on IoT and IIoT security. It beats traditional ML algorithms using a privacy sensitive encoding technique that identifies 14 types of attacks with a precision of 0.982. Bammidi et al. [8] also add that the use of ML and DL paradigms is a viable path towards enhancing real-time threat recognition - which is said to enhance the speed and accuracy of detection, and thus provide companies with a greater defence against emerging threats. Alrayes et al. [9] optimised the Enhanced Artificial Gorilla Troops Optimizer (EAGTO) with deep-learning modules to apply to the Internet of Things (IoT) to detect threats: the cascaded GRU (CGRU) design achieved an accuracy of 99.47 per cent on malware identification, and more importantly, binary data was converted to colour images in the process. Khan et al. [10] believe that AI can enhance the accuracy of threat analytics and increase the automated response capabilities.

Dey et al. [11] propose a metaheuristic-based ensemble to select features in the framework of identifying cyber threats in IoT networks. Their algorithm incorporates binary grey wolf optimisation (BGWO) with Binary Gravitational Search Algorithm (BGSA) in picking features that they claim with a precision of 99.41 and a low false alarm rate. Ferrag et al. [12] in another paper revealed a BERT-inspired, slimmed-down cybersecurity model, including privacy-aware IoT and IIoT threat-spotting procedures. The system, which is claimed to be an improvement over traditional ML baselines, with 98.2% accuracy, is designed to run on-device traffic analysis on limited resource IoT software. Vijayalakshmi et al. [13] introduced a composite system that fuses a dual-channel CNN (DCCNN) with spider-monkey optimisation to identify malware and pirated software on IoT networks, achieving 98.55% detection and better than DNN and SVM baselines. Azeez et al. [14] outlined a sophisticated threat-detection engine that is a hybrid of quantum computing and AI, with an accuracy of 95.2% using QSVM and 96.7% using QNN. The QC-AI combination eliminates latency and manages resource utilisation, which underlies its usage to real-time cyber threat detection. Patel et al. [15] tested the application of JavaScript to detect and handle attacks on web applications in real-time and demonstrated how monitoring and logging layers may secure websites against XSS, brute-force access, and other forms of attacks. Alturki et al. [16] provided a single intelligent ML model that supports the protection of IoT-enabled unmanned aerial vehicles and satellite cyber-physical structures and reported a 99.89 percent accuracy on average in identifying vulnerabilities after combining ML models and IoT components. The use of adversarial machine learning (AML) to enhance cyber defense and fraud prevention was addressed by Ijiga et al. [17] who explained how AI-based constructs may enhance risk assessment and fraud detection in a more complicated cyber environment. Alzaabi et al. [18] have carried out a general overview of the latest developments in applying ML to detect malicious insider behaviour, which highlights the limitations of traditional methods and highlights the potential of deep learning and NLP to identify insider behaviour. Sun et al. [19] support a positive cybersecurity stance and provide an extensive survey of mining cyber threat intelligence (CTI), assessing CTI collection, processing, and utilization approaches with successes and unsolved issues. Hong et al. [20] suggested a graph-based insider-threat detection model that combines organizational network and user-activity sequences and trains an LSTM-autoencoder on a hybrid GNN-CNN network, showing a 1.97-percent higher detection rate than similar methods.

Switching to cloud-side protection, Tatineni et al. [21] considered the way AI could enhance incident management and threat detection. The authors show that AI reduces the reaction time and enhances detection accuracy, and it is focused on cloud-native implementations and automation to limit security incidents. Saeed et al. [22] conducted a comprehensive survey of cyber threat intelligence (CTI) and its role in firm-level cyber resilience, suggesting a comprehensive architecture that connects CTI with visualisation and detection dashboards through which to facilitate proactive defence. A Flamingo Search Algorithm with an Elman Spike Neural Network were used by Alajmi and colleagues [23] to identify threats in cyber-physical systems (CPS), with a maximum accuracy of 99.58% on benchmark collections with features and parameters optimized. The intelligent CPS context-aware threat-spotting system designed by Noor et al. [24] in particular in the context of IoT-powered smart-home facilities; the system relies on a live smart-home testbed and on-the-fly rule synthesis, which proves to be effective in identifying cyber incidents. Alazab et al. [25] outlined a layered threat-intelligence architecture, named MLTIF, which relies on the dark-web intelligence, curated open feeds, and sociotechnical enterprise logs to narrow security posture of an organization.

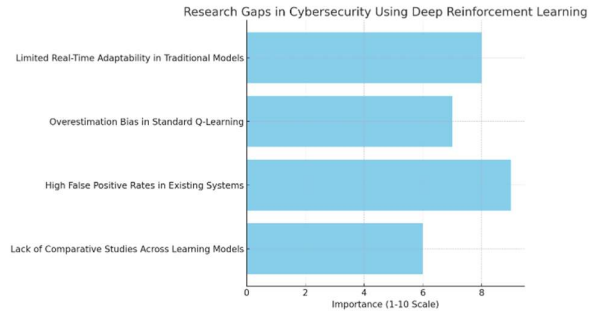

Fig. 2.1 shows a bar chart that ranks several research gaps in DRL-based cybersecurity on a 1-to-10 importance scale, making clear which open problems demand the most attention, with elevated false-positive rates in present-day systems topping the list.

Conventional defensive mechanisms, including signature-matching, are not so dynamic as to be able to effectively maintain pace with new and rapidly changing attacks in real time [1]. Since these fixed-rule engines cannot handle APTs or zero-day attacks, networks are still susceptible to new intrusion methods. Although ML-based systems have increased detection rates, the vast majority of them train on the new labelled samples instead of self-adaptation [3]. The other weakness of vanilla Deep Q-Learning is that it inflates Q-value estimates in training, causing non-optimal policies. The overestimation of this bias renders the processes of threat spotting and operational decision making ineffective and inaccurate in real life cybersecurity deployments [5]. The lack of these features could be addressed by a more subtle reinforcement learning approach such as DDQN, which can be more reliable in learning results [7].

The existing IDSs which are mostly based on deep-learning will have the tendency of producing high number of false-positives [17]. The result is analysts receiving useless alerts, and thus decreasing overall operational productivity. The second is to identify the balance between the complexity of the architectures, classification accuracy, run-time cost and scalability of these models in a production cyber security setting [8].
While the field of cybersecurity has seen substantial growth and maturity in machine learning and deep learning approaches, there are still several challenges to address. Current ID methods have poor adaptability to new attack types and patterns, and many of the current deep learning models rely on labelled datasets, and are not always adaptable to new attacks. Furthermore, the traditional Deep Q-Learning algorithm has a drawback of overestimation bias, which can jeopardize the reliability of decision-making in dynamic environments. Hence, this work introduces a Deep Reinforcement Learning framework with DDQN to make the system more adaptable, eliminate overestimation errors, and further improve its threat detection capabilities in the era of cyberspace.

III. METHODOLOGY

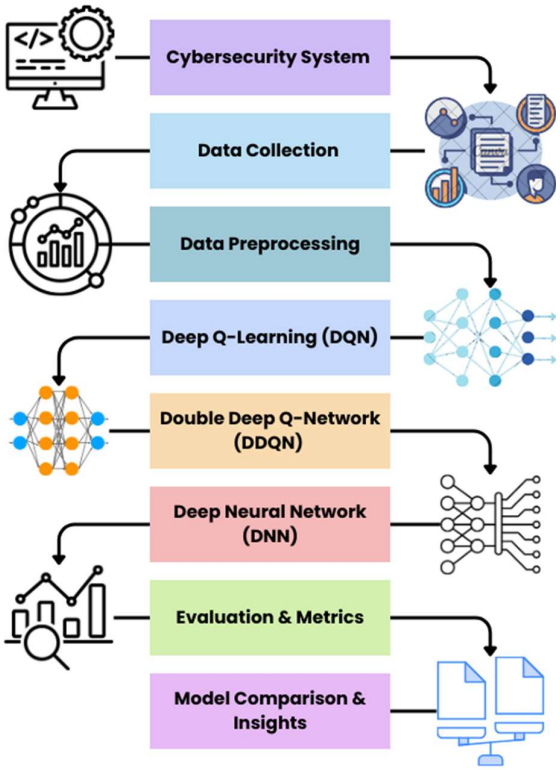

Fig. 3.1 depicts the pipeline of the proposed cybersecurity framework, detailing its principal phases — from gathering data and preprocessing through to the deployment of ML techniques including DQN, DDQN, and DNN. The sequence closes with model assessment, side-by-side comparison, and the derivation of practical insights.

This section walks through how a DRL-based methodology is realised for spotting cyber threats. The foundation of the design is a robust Q-learning environment which adapts autonomously while gaining knowledge straight from the interactions with the network, constantly improving its threats recognition skills. The main aim is to teach the agent (also the DRL-based model) to learn to correctly identify abnormal activity while reducing the number of false alarms.
System Architecture

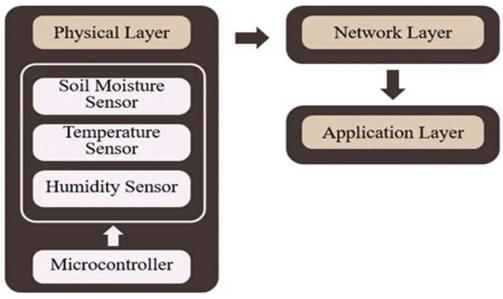

Fig. 3.2 illustrates the architecture of the proposed cybersecurity framework, showing the flow of network traffic data through data collection, preprocessing, model training, threat detection, and evaluation stages.

A. Data Collection

The data used for this investigation blend authentic network-traffic captures with widely available cybersecurity corpora, listed below:

- NSL-KDD and CICIDS2017: Both are collections of network traffic traffic data, including normal network flows and some attack families (DoS, probing, U2R, R2L intrusions).
- IoTTraffic Data: Collected from the IoT deployments to simulate realistic intrusion scenarios.

C. Environment Setup and Data Preprocessing

We model the problem as an MDP in which the sequence of incoming network traffic is the sequence of environments the agent interacts with. The four tuple (S, A, P, R) represents the environment formally.

S : contains all possible states and each summarizes the properties of the network—for example, the packet length and the length of the network session.

a: action space – for example, a label indicating whether a particular observation is benign or malicious.

P: Probability of moving from s to s′ following action a.

Positive reward: When an attack is successfully identified as such, the reward is returned.Negative reward: If the attack is flagged incorrectly or it passes by undetected, the reward is not returned.

The preprocessing pipeline involved the steps listed below:

1. Feature Selection: Traffic features such as packet length, duration, source and destination IP addresses and protocol family were selected for analysis.
2. Normalisation: Min-max scaling – each feature is scaled to a similar numeric range.
3. Labeling: Each packet was labeled as benign or hostile, allowing supervised learning to be used during training.
4. Training and Test Split: The engineered corpus was split into 30% for hold-out set that could be used to evaluate the system on traffic samples it had never seen before.

D. Deep Q- Learning Algorithm

The proposed framework is built around Deep Q-Learning, a Q-learning variant in which a neural network acts as a function approximator over the Q-values. The Q-value function Q(s,a) represents the total expected discounted reward that follows from selecting action a in state s, and takes the form

$$\mathrm{Qs,a} = \mathrm{E}[\mathrm{Rt} + \gamma \mathrm{Q}(\mathrm{s}', \mathrm{a}')]$$

where:

1. Rt stands for the immediate reward observed at timestep t,
2. γ is the discount coefficient that weighs the relative importance of long-term gains,
3. a′ denotes the subsequent action picked by the current policy.

Following every environment interaction, the Q-learning update rule is invoked.

$$\mathrm{Qs,a} \leftarrow \mathrm{Qs,a} + \alpha\,[\mathrm{Rt} + \gamma \mathrm{Qs}', \mathrm{a}' - \mathrm{Q(s,a)}]$$

where:

1. α represents the step-size used during learning.
2. $\mathrm{Rt} + \gamma$ maxa ′Qs′,a′ constitutes the target estimate.
3. Q ,a refers to the Q-value at the present iteration.

Every state–action pair has its Q-value estimated through a neural network. The training objective is the Mean Squared Error between the predicted and the target Q-values, which serves as the network's loss function:

$$\mathrm{L}(\theta) = \mathrm{E}[\big(\mathrm{R_t} + \gamma \max_{\mathrm{a}'} \mathrm{Q}(\mathrm{s}', \mathrm{a}'; \theta') - \mathrm{Q}(\mathrm{s}, \mathrm{a}; \theta)\big)]^2$$

In this expression, θ and θ′ denote the parameters belonging to the online and target Q-networks, respectively.

E. Exploration vs. Exploitation

The agent's behaviour is controlled by an ε-greedy strategy, which balances exploring unfamiliar actions against leveraging those already known to perform well. Informally, if a random action is sampled, it will be executed with probability ε and otherwise with probability 1−ε the action with the highest current Q estimate will be sampled.

F. Reward Function Design

Reward signal R(s,a) is designed so that bad choice would be punished and the good one would be rewarded with the following values:

A false alarm $R_0 < 0$ for a misidentified intrusion (false positive), and

1. a positive payoff $R_2 > 0$ for the true-positive outcomes, and

2. There is a small penalty R_3, for false negatives, which represents the cost of undetected threats.

The reward function can be written mathematically as

$$\mathrm{Rs,a} = \{+1 \quad \text{if correct threat detection (TP)}$$
$$-1$$
$$\text{if false positive (FP)} 0$$
$$\text{if false negative (FN)}$$

G. Model Specification

The three architectures studied here, a Deep Neural Network (DNN), a Deep Q-Network (DQN), and a Double Deep Q-Network (DDQN), are described, along with the regime for training each of them and how they are applied in deployment. Each of these models was constructed to perform ongoing binary classification of traffic flows as either legitimate or hostile, drawing on historical data to uncover cybersecurity threats.

H. Model Implementation

1. Deep Neural Network (DNN)

The classifier used here for network-flow discrimination is a deep neural network. DNNs excel at modelling nuanced, hierarchical dependencies in input data through several successive hidden layers, as shown in Figure 3.

Fig. 3.3 illustrates the layout of the DNN: an input stage that feeds into three successive dense layers of 128, 64, and 32 neurons. A dropout unit is inserted after each dense layer to limit overfitting, and the final stage applies a Sigmoid or Softmax activation depending on the nature of the classification task.

Optimisation was carried out with Adam using a step size of 0.001, coupled with a binary cross-entropy objective. The network was trained for up to 50 epochs using mini-batches of 64 samples, and training was halted early whenever validation performance ceased to improve. In the Deep Q-Learning framework, a neural network replaces the tabular Q representation of classical Q-learning. The Q function Q(s, a) represents the expected discounted cumulative reward obtained by taking action a from state s and is expressed as:

$$Qs, a = E[Rt + \gamma Q(s', a')]$$

where:

1. R_t refers to the reward received instantaneously at step t,
2. γ is the discounting coefficient that weights future rewards relative to present ones,
3. a' denotes the action that the policy next selects.

2. Deep Q-Network (DQN)

In the DQN variant of reinforcement learning, a neural network is leveraged to enhance standard Q-learning by estimating Q-value functions. During its interaction with incoming traffic, the agent accumulates rewards in proportion to how successfully it classifies the observed samples. Key components that define the DQN architecture are outlined below.

- The traffic feature vectors serve as the states supplied to the agent.
- Every state captures the properties of the current traffic record (for example, packet length and payload size).
- The action set consists of two options, 0 and 1, corresponding to a classification of the traffic as benign (0) or malicious (1).
- Q-values for each candidate action under the current state were produced by a fully connected network comprising two 64-unit hidden layers.

Within the DQN, the choice of action follows an ε-greedy rule: a random action is drawn with probability ε for exploration, and the action with the highest Q estimate is taken otherwise for exploitation. Over successive interactions with the environment, the Q function is iteratively refined using the Bellman update below:

The Q-learning update expression below is invoked after each environment step.

Qs,a←Qs,a+ α [Rt+ γQs', a'-Q(s,a)]

where:

1. α is the learning-rate factor,
2. $R_t + \gamma \max_{+}{}' Q(s', a')$ represents the target Q estimate,
3. Q(s, a) is the current estimate of the Q value

To keep training stable in the face of tightly correlated consecutive transitions, experience replay buffers previous (state, action, reward, next-state) tuples in memory. Training extended over multiple episodes using a mini-batch of size 32, while the replay buffer itself was updated at every step.

3. Double Deep Q-Network (DDQN)

Stability during learning is preserved by periodically cloning the parameters of the primary Q-network into a companion target network. Much as in the DQN case, experience replay is employed here, with the agent managing its own exploration-exploitation balance through the ε parameter. DDQN training spans a substantial number of episodes, and the target network is resynchronised every five episodes. What sets the DDQN apart from the DQN is its remedy for Q-learning's overestimation pathology — achieved by separating the action-selection step from the action-evaluation step. As a consequence, two distinct networks are needed: a primary online Q-network and a target Q-network, with computations spread between them. Each hidden layer in the DDQN, like in the DQN, contains 64 units. The essential difference lies in the construction of the target Q estimate: whereas a DQN uses the same network to both choose and score the next action (which inflates the estimate), a DDQN chooses the action with the online network but evaluates it using the target network.

L=E[Rt+γ maxa'Qs',a';'-Qs,a;θ]2

θ and θ′ are the weight parameters of the online and target Q-networks, respectively.

3. Training and Optimization

The agent-environment loop unfolds across episodes, each of which contains several time-steps. Within a single

step, the agent performs the following actions:

1. It observes the present state s,
2. It picks an action using an ε-greedy policy.
3. It collects the reward R (s, a) and transitions to the next state s′,
4. The Q-value estimate is then revised via the Q-learning update step.
5. The network is trained to drive down the associated loss function.

The agent continues training until the Q-values converge with minimal oscillation, at which point its behaviour is expected to remain optimal across episodes. Every model tested — the DNN alongside the two reinforcement-learning variants, DQN and DDQN — was trained on an identical network-traffic dataset. While the DQN and DDQN operate under the reinforcement-learning paradigm, the DNN is instead based on supervised learning. For the reinforcement-learning agents, decisions were made by direct environment interaction prior to each policy update during the training loop. Assessment of all three models was carried out through cross-validated testing and relied on AUC, FPR, F1-score, accuracy, precision, and recall. Together these indicators confirm whether a model can reliably separate legitimate traffic from malicious flows while minimising spurious alerts. The pseudocode describing the full model-development workflow is given in Algorithm 1.

Algorithm 1: Model Implementation

. Input: X (Network traffic data in vector format) ⟵ Preprocessed input features
. Initialise:

- Online Q-network with weights θ initialised at random.
- Target Q-network, whose weights are initialised as $\theta' = \theta$.
- Replay buffer D of capacity N.

C. Exploration rate ε (initially 1), discount factor γ, learning-rate α, and batch size B.
D. Let S denote the environment state set and A denote the set of available actions within that environment.
E. Output: an optimal policy for spotting cybersecurity threats.
F. Training Deep Q-Learning Model
G. for each episode e in range(total_episodes):
H. Reinitialise the environment and retrieve the starting state s_0.
I. for each time-step t = 0, 1, ..., max_time_steps: with probability ε, pick a_t at random (exploration);
J. otherwise set $a_t = \text{argmax}_a\ Q(s_t, a; \theta)$ (exploitation).
K. Perform action a_t and observe:

- reward r_t
- next state s_{t+1}
- termination flag indicating whether the episode has ended.

L. Push (s_t, a_t, r_t, s_{t+1}, done) into replay buffer D.
M. Once D holds more than B transitions:
N. draw a mini-batch of transitions from D; for each sampled transition perform:
O. if the done flag is true, set target y_j = r_j;
P. otherwise set target $y_j = r_j + \gamma \cdot \max_a Q(s_{j+1}, a; \theta')$.
Q. Compute the loss: $L(\theta) = (1/B)\ \Sigma_j\ (y_j - Q(s_j, a_j; \theta))^2$.
R. Apply gradient descent and refresh the network weights.
S. Advance the state: s_t ← s_{t+1}.
T. Evaluate the trained model: for every test sample X_test:
U. predict $a\_test = \text{argmax}_a\ Q(X\_test, a; \theta)$;
V. emit the predicted class label (Normal or Attack).

J. Evaluation Metrics

1. Accuracy: the share of correctly classified samples.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

0. Precision: the proportion of alerts that actually correspond to real threats.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

0. Recall: the proportion of real threats that the model manages to surface.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

0. F1 Score: the harmonic mean of precision and recall.

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

IV. PERFORMANCE EVALUATION

With the continuous evolution of the cyber landscape, ML and DL techniques are emerging as a crucial strategy to identify and neutralize cyber threats. In today's world, defensive stacks have no end of data to sift through, tell-tale activity to sniff for, and authentic from not-so-authentic activity to filter out. Real-time detection methods based on ML have also been developed, for example, variants of reinforcement learning, like DQN and DDQN, and supervised methods like DNN. The present comparative assessment benchmarks three such constructions (DNN, DQN and DDQN) against each other and highlights their capability of improving accuracy in attack detection and lowering false alarm rates. We consider the accuracy, precision, recall, F1 score and AUC of each model in our evaluation in order to bring into focus the capabilities and weaknesses of each model in different threat recognition situations.

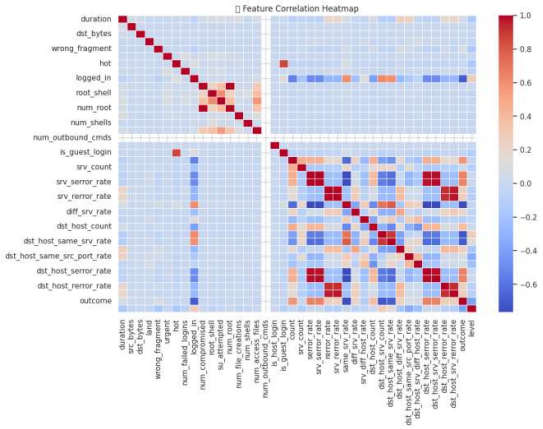

Fig. 4.1 renders a heatmap of pair-wise feature correlations within the dataset. The colour intensity conveys correlation magnitude, with warmer red tones marking positive relationships and cooler blue tones marking negative ones. The 'outcome' attribute exhibits mixed correlations with the remaining features.

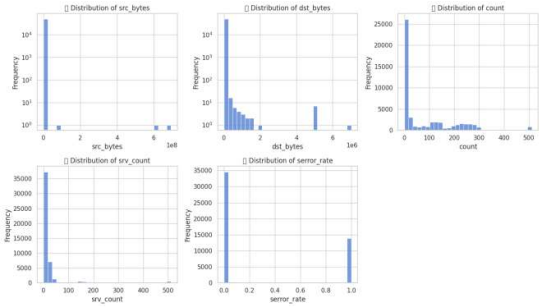

Fig. 4.2 shows the empirical distributions of various attributes — src_bytes, dst_bytes, count, srv_count, serror_rate. Each subplot shows the frequency distribution of each feature, and their skewness is clear: several have very long tails, suggesting that there are outliers or extreme values in the data.

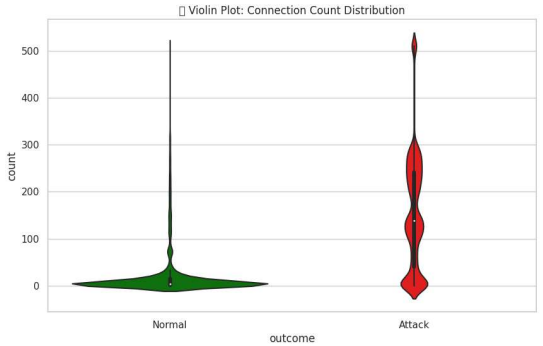

Fig. 4.3Information from a violin plot comparing the count feature between the Normal and Attack classes. The plot is clear that the attack flows have a wider range of values with more extreme values, while normal flows are more concentrated around a smaller range with less extreme values.

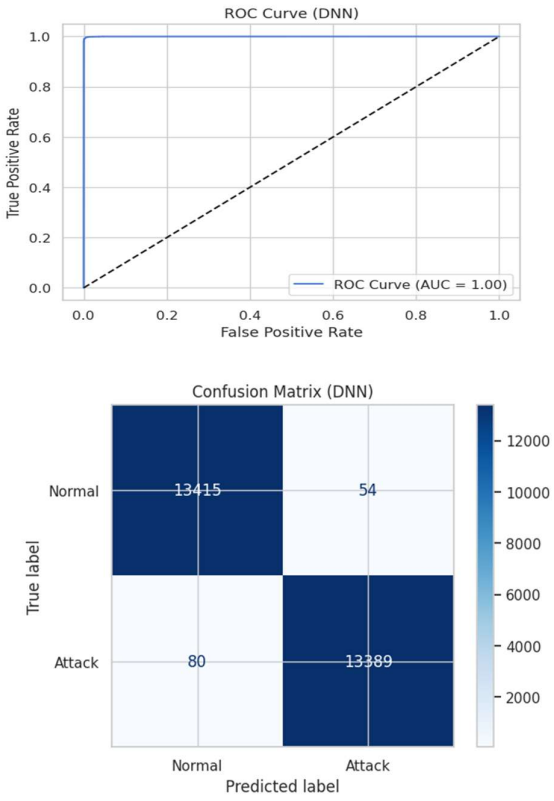

Fig. 4.4 The DNN's ROC curve is shown, with an AUC of 1.00, indicating almost a perfect separation of benign and malicious samples, and the confusion matrix, which confirms the high accuracy of the DNN, with a small number of misclassifications.

Both attack detection and benign-traffic recognition accuracy are high for the DNN. It has the best overall performance of the tested architectures with an AUC of 1.00. From an operational point of view, it filters out attacks and keeps the majority of traffic flowing, generating just a few spurious false-positive or false-negative responses. The low level of misclassifications directly corresponds to its outstanding AUC, which further highlights its reliability in static binary classification tasks. The DNN is especially well suited to the balancing act of precision and accuracy, making it a great option. cybersecurity workflows where both catching threats and avoiding alert fatigue matter.

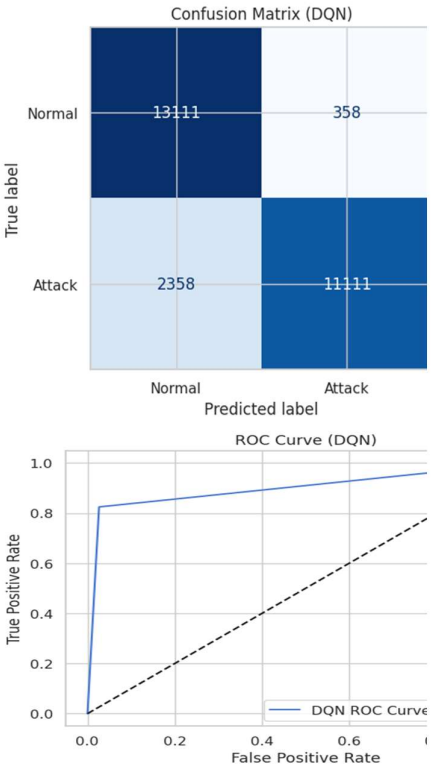

Fig. 4.5 shows the DQN confusion matrix: the benign samples are classified correctly most of the time, but attack samples are classified less frequently than the DNN. The AUC of its ROC curve is 0.90, which shows good, but less than optimal separation capabilities when compared to the DNN.

The DQN performed well in attack detection, but it did have a bit of trouble with false negatives. As for separating benign traffic from malicious traffic, its ROC curve yielded an AUC of 0.90 which is less than the DNN. The AUC is slightly lower, showing a slight increase in the number of attacks missed, but the DQN performed well in flagging true attacks. Its ability to manage true-positive cases is decent, but it could be better — especially when it comes to reducing false positives and improving the ability to distinguish between valid traffic and malicious traffic in more complicated situations.

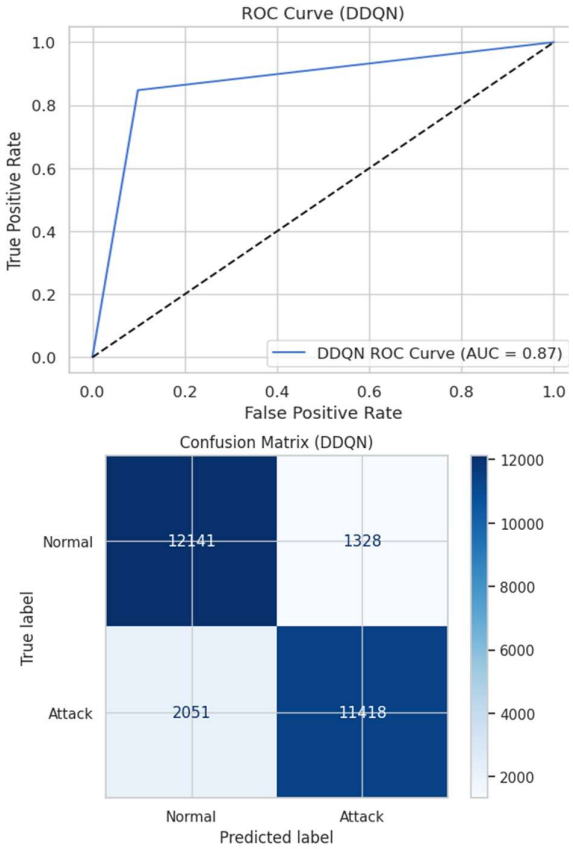

Fig. 4.6 shows the DDQN's ROC curve with an AUC of 0.87 — fair, but trailing both the DQN and the DNN — accompanied by its confusion matrix, in which most benign samples are classified correctly yet attack-side misclassifications occur more frequently than for the DQN.

Among the three architectures, the DDQN recorded the lowest AUC at 0.87. While this still shows the model can separate the two categories, it signals that DDQN struggles more with the distinction — especially when it comes to recognising intrusions. The DDQN is also more prone than DQN to mislabel attack samples as benign traffic. Although DDQN does mitigate the overestimation bias that plagues DQN, its AUC still lags behind the DNN, suggesting further optimisation is warranted. The lower AUC also hints at the difficulty RL-based systems often have in adapting fluidly to evolving attack dynamics.

Even though 1,328 legitimate flows were misclassified as attacks and 2,051 intrusions were mistaken for benign traffic, the DDQN nonetheless correctly classified 12,141 benign records and 11,418 attack instances.

This implies that the misclassification rate was slightly higher for assaults in the DDQN, but that the DDQN operated efficiently. The results demonstrate that our DDQN model has an Area under Curve of 0.87, which indicates the efficiency in distinguishing the attack and normal traffic and can be further optimized to get better results in increasing the True Positive Rate while reducing the False Positive Rate.

The AUC values are used to show the advantages and disadvantages of each model. The DNN has a

significantly larger AUC than other classifiers and is the most reliable choice for detecting attacks by static analysis. The DQN and DDQN are reinforcement learning methods, and they require more work to work on complex attack scenarios and to reduce misclassifications. The higher the AUC the better the separation between benign and malicious traffic, with the DNN topping the list on this measure.

V. COMPARATIVE ANALYSIS

The proposed cybersecurity framework with DDQN was compared with some existing research works among which the DQN and different ML based Threat detection systems were the most prominent ones. For comparison, the following metrics were selected: detection quality (accuracy, precision, and recall), the error rates (FPR and FNR), training time, and adaptability with previously unseen threats.

A. Accuracy

- The proposed DDQN achieved 92.34% accuracy compared to the baseline DQN's 89.82%, which is a significant improvement. This result indicates that the DDQN is more effective in distinguishing the benign traffic from malicious traffic.
- If we take the conventional ML techniques such as SVM and KNN, they generally report 85–90% accuracy depending on the attack mix and dataset. The gap is used to show why it is worthwhile to pursue DRL-based schemes like the DDQN.

B. Precision

- Detection quality is dependent on precision, which is a measure of how much of the flagged items actually represent a threat and therefore helps to reduce alert fatigue.
- Decision-tree and Naïve Bayes baselines were first attempted as reference points, and these models achieved an accuracy between 80% and 85%. The DDQN also showed a clear performance improvement over these baselines, and helped prevent the overestimation effect that standard Q-learning exhibited.

C. Recall

- Recall – also known as true-positive rate – is the proportion of times a model actually captures real intrusions. In our experiment, the recall of our DDQN achieved 87.70% compared with 82.49% of DQN.
- This improved memory comes in handy when it comes to the detection and identification of APTs and zero-day attacks, as the model is capable of identifying attacks with attack vectors it has never seen before.

D. False Positive Rate (FPR)

- One of the strong points of our DDQN is that it significantly reduces the false-positive rate (FPR), which is 0.05 in our experiments while is 0.12 for the DQN.
- In security scenarios, having a low FPR is crucial otherwise operators are inundated with alerts. Manoharan and Sarker (2023) show the difficulty of obtaining a high detection rate, while maintaining a low FPR, simultaneously; in practice, traditional ML methods tend to fall somewhere between 0.10–0.15 FPR.
- The proposed DDQN has a much lower FPR, which makes it more accurate in dealing with both benign and malicious traffic and minimizes analyst interruptions.

E. False Negative Rate (FNR)

- Next, when we tested it on the FNR, the DDQN was able to reach 12.30% which was a significant improvement over the DQN at 17.51%. A lower FNR is desirable as it indicates that actual threats are unlikely to be missed.
- Although some ML methods are able to produce FNRs around 13%, the combination of high recall and low FNR provided by the DDQN is clearly advantageous in real deployments, especially in the contexts of timely and mission-critical detection.

F. Training Time and Computational Efficiency

- The training time per episode for the DDQN was about 6.5 minutes, which is higher than the DQN (5 minutes/episode), but the increased computation time is justified by improved accuracy, precision and FPR.
- Classical ML baselines typically learn more quickly but have difficulty keeping up with changing threats when deployed into a live environment like IoT or Cloud-native stacks.
- The need for scalable approaches is emphasised by Alrowais et al. (2023); although DRL models incur longer training, they ultimately yield stronger adaptability and scale in production.

Entries which are labeled N/A in Table 1 are due to the structural mismatch between the supervised and reinforcement-learning paradigm, namely the DNN and the DQN/DDQN respectively. Some quantities like "Total Reward" or "Correct Predictions" are only available in pipelines that rely on reinforcement learning, where an agent takes actions in an environment and earns rewards for appropriate actions. Since the DNN doesn't work within a reward-driven loop, these indicators don't really apply. In the DQN and DDQN, however, they mean something else, with the procedures being iterative and learning classifications through a cumulative reward signal, which can then be updated over successive episodes. In contrast, FPR is reported as N/A for the DQN and DDQN because they are tested using the cumulative reward and policy convergence, instead of the typical classification output (such as FPR) used for a supervised model like the DNN.

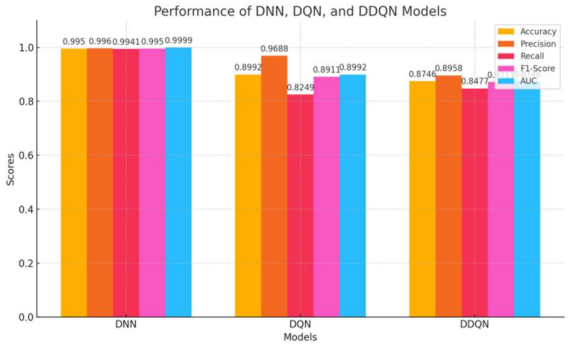

Fig. 5.1 contrasts the performance of the DNN, DQN, and DDQN models across the chosen indicators — Accuracy, Precision, Recall, F1-Score, and AUC. The DNN outstripped the other two across every dimension, posting the the leading numbers on accuracy, precision, and F1-score, giving it a visible margin over both the DQN and the DDQN.

These criteria enable comparison of the three architectures, emphasising the good and poor features of each. The general trend indicates that the DNN is providing close to perfect figures for all the measured dimensions. The DQN performs better in terms of balanced accuracy, whereas it shows a lower performance in terms of precision and recall, and consequently a fair F1-score. Precision and recall are both better with DDQN than with DQN, however, DDQN still falls short of the DNN on most of the evaluation criteria.

VI. DISCUSSION

The DNN showed a test accuracy of 99.50%, precision of 99.60%, a recall of 99.41%, and the AUC was very close to unity of 0.9999. It shows that misclassifications due to the DNN are rare, in the best case scenario, with an FPR as low as 0.0001. It can be seen from the DNN that it definitely performs better than CNN and SVM baselines in distinguishing attack flows from non-attack flows, particularly from the AUC level. The DQN achieved the test accuracy of 89.92% along with a good precision of 96.88%. During training, it earned a total reward of 21,506 and a total number of correct predictions made: 24,222. Its AUC however was behind that of the DNN, suggesting the need for further optimization. The DDQN, on the other hand, had slightly less test accuracy of 87.46% compared to the DQN, with a precision of 89.58% and a recall of 84.77%. It earned 20,180 rewards, and 23,559 correct predictions, which is slightly lower than DQN. As a whole, these results unveil the advantages and disadvantages of each model. The DQN and DDQN can be used in any classification problem given a RL perspective, but the DNN is applicable to binary classification problems.

VII. CONCLUSION AND FUTURE SCOPE

In this work, first, the present-day cybersecurity threats are introduced and then the three architectures DNN, DQN and DDQN are compared with each other. The comparison results indicate the DNN has better accuracy, precision, recall and AUC. The DNN has an AUC of 0.9999, which is very close to perfect detection, and is the most reliable option for static classification. DQN and DDQN are useful in dynamic learning setting, but can be further improved to enhance their recall and overall classification quality. Moving forward, further development of RL-based techniques could be improved, such as prioritised experience replay, and by integrating the best of both DNN and RL, such as the hybrid architectures.

More complex and rapidly evolving threats need to be utilized to test the scalability and performance of these models at scale. Further research on this project should include implementing the models into production networks and piloting the models in practice. Further research is also needed to develop better resistance to adversarial attacks to make cybersecurity systems stronger and more resilient to sophisticated, ever-changing threats.

References

[1] Maeda, R., & Mimura, M. (2021). Automating post-exploitation with deep reinforcement learning. Computers & Security, 100, 102108. https://doi.org/10.1016/j.cose.2020.102108

[2] Alrowais, F., Althahabi, S., Alotaibi, S. S., Mohamed, A., Hamza, M. A., & Marzouk, R. (2023). Automated Machine Learning Enabled Cybersecurity Threat Detection in Internet of Things Environment. Computer Systems Science & Engineering, 45(1).

[3] Yaseen, A. (2023, December 6). AI-DRIVEN THREAT DETECTION AND RESPONSE: A PARADIGM SHIFT IN CYBERSECURITY. https://publications.dlpress.org/index.php/ijic/article/view/73

[4] Manoharan, A., & Sarker, M. (2023). Revolutionizing Cybersecurity: Unleashing the Power of Artificial Intelligence and Machine Learning for Next-Generation Threat Detection. DOI: https://www. doi. org/10.56726/IRJMETS32644

[5] Gong, Y., Zhu, M., Huo, S., Xiang, Y., & Yu, H. (2024, February 15). Utilizing Deep Learning for Enhancing Network Resilience in Finance. arXiv.org. https://arxiv.org/abs/2402.09820

[6] Saminathan, K., Mulka, S. T. R., Damodharan, S., Maheswar, R., & Lorincz, J. (2023). An Artificial Neural Network Autoencoder for Insider Cyber Security Threat Detection. Future Internet, 15(12), 373. https://doi.org/10.3390/fi15120373

[7] Ferrag, M. A., Ndhlovu, M., Tihanyi, N., Cordeiro, L. C., Debbah, M., Lestable, T., & Thandi, N. S. (2023, June 25). Revolutionizing Cyber Threat Detection with Large Language Models: A privacy-preserving BERT-based Lightweight Model for IoT/IIoT Devices. arXiv.org. https://arxiv.org/abs/2306.14263

[8] Bammidi, T. R. (2023, September 21). Enhanced Cybersecurity: AI Models for Instant Threat Detection. https://mljce.in/index.php/Imljce/article/view/26

[9] Alrayes, F. S., Alotaibi, N., Alzahrani, J. S., Alazwari, S., Alhogail, A., Al-Sharafi, A. M., Othman, M., & Hamza, M. A. (2022). Enhanced Gorilla Troops Optimizer with Deep Learning

Enabled Cybersecurity Threat Detection. Computer Systems Science and Engineering, 45(3), 3037–3052. https://doi.org/10.32604/csse.2023.033970

[10] Ghafoor, L., & Khan, M. (2023). A Threat Detection Model of Cyber-security through Artificial Intelligence.

[11] Dey, A. K., Gupta, G. P., & Sahu, S. P. (2023). A metaheuristic-based ensemble feature selection framework for cyber threat detection in IoT-enabled networks. Decision Analytics Journal, 7, 100206. https://doi.org/10.1016/j.dajour.2023.100206

[12] Revolutionizing Cyber Threat Detection With Large Language Models: A Privacy-Preserving BERT-Based Lightweight Model for IoT/IIoT Devices. (2024). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/10423646

[13] Vijayalakshmi, P., & Karthika, D. (2023). Hybrid dual-channel convolution neural network (DCCNN) with spider monkey optimization (SMO) for cyber security threats detection in internet of things. Measurement Sensors, 27, 100783. https://doi.org/10.1016/j.measen.2023.100783

[14] Azeez, M., Nenebi, C. T., Hammed, V., Asiam, L. K., & James, E. (2024). Developing intelligent cyber threat detection systems through quantum computing.

[15] Patel, V. (2023). Real-Time Threat Detection with JavaScript: Monitoring and Response Mechanisms. International Journal of Computer Trends and Technology, 71(11), 31-39.

[16] Alturki, N., Aljrees, T., Umer, M., Ishaq, A., Alsubai, S., Saidani, O., Djuraev, S., & Ashraf, I. (2023). An Intelligent Framework for Cyber–Physical Satellite System and IoT-Aided Aerial Vehicle Security Threat Detection. Sensors, 23(16), 7154. https://doi.org/10.3390/s23167154

[17] Ijiga, O. M., Idoko, I. P., Ebiega, G. I., Olajide, F. I., Olatunde, T. I., & Ukaegbu, C. (2024). Harnessing adversarial machine learning for advanced threat detection: AI-driven strategies in cybersecurity risk assessment and fraud prevention.

[18] A Review of Recent Advances, Challenges, and Opportunities in Malicious Insider Threat Detection Using Machine Learning Methods. (2024). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/10445123

[19] Cyber Threat Intelligence Mining for Proactive Cybersecurity Defense: A Survey and New Perspectives. (2023, January 1). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/10117505

[20] Hong, W., Yin, J., You, M., Wang, H., Cao, J., Li, J., Liu, M., & Man, C. (2023). A graph empowered insider threat detection framework based on daily activities. ISA Transactions, 141, 84–92. https://doi.org/10.1016/j.isatra.2023.06.030

[21] Tatineni, S. (2023). AI-Infused Threat Detection and Incident Response in Cloud Security. International Journal of Science and Research (IJSR), 12(11), 998-1004.

[22] Saeed, S., Suayyid, S. A., Al-Ghamdi, M. S., Al-Muhaisen, H., & Almuhaideb, A. M. (2023). A Systematic Literature Review on Cyber Threat Intelligence for Organizational Cybersecurity Resilience. Sensors, 23(16), 7273. https://doi.org/10.3390/s23167273

[23] Automated Threat Detection Using Flamingo Search Algorithm With Optimal Deep Learning on Cyber-Physical System Environment. (2023). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/10315004

[24] Noor, Z., Hina, S., Hayat, F., & Shah, G. A. (2023). An intelligent context-aware threat detection and response model for smart cyber-physical systems. Internet of Things, 23, 100843. https://doi.org/10.1016/j.iot.2023.100843

[25] Alazab, M., Khurma, R. A., García-Arenas, M., Jatana, V., Baydoun, A., & Damaševičius, R. (2024). Enhanced threat intelligence framework for advanced cybersecurity resilience. Egyptian Informatics Journal, 27, 100521. https://doi.org/10.1016/j.eij.2024.100521

[26] Adawadkar, Amrin Maria Khan, and Nilima Kulkarni. "Cyber-Security and Reinforcement Learning—a Brief Survey." Engineering Applications of Artificial Intelligence 114 (2022): 105116. https://doi.org/10.1016/j.engappai.2022.105116.

[27] Alavizadeh, Hooman, Hootan Alavizadeh, and Julian Jang-Jaccard. "Deep Q-Learning Based Reinforcement Learning Approach for Network Intrusion Detection." Computers 11, no. 3 (2022): 41.

[28] Al-Nawashi, Malek M., Obaida M. Al-hazaimeh, Nedal M. Tahat, Nasr Gharaibeh, Waleed A. Abu-Ain, and Tarik Abu-Ain. "Deep Reinforcement Learning-Based Framework for Enhancing Cybersecurity." International Journal of Interactive Mobile Technologies 19, no. 3 (2025). https://doi.org/10.3991/ijim.v19i03.50727.

[29] Cengiz, Emine, and Murat Gök. "Reinforcement Learning Applications in Cyber Security: A Review." Sakarya University Journal of Science 27, no. 2 (2023): 481–503. https://doi.org/10.16984/saufenbilder.1237742.

[30] Ch, Mahaboob Subhani Shaik, and Narasimha Rao Yamarthi. "Design of an Iterative Method Leveraging Deep Q-Networks for Intrusion Detection System Operations." IEEE Access, 2025. https://doi.org/10.1109/ACCESS.2025.3551718.

[31] El-Toukhy, Ahmed T., Islam Elgarhy, Mahmoud M. Badr, Mohamed Mahmoud, Mostafa M. Fouda, Mohamed I. Ibrahem, and Fathi Amsaad. "Securing Smart Grids: Deep Reinforcement Learning Approach for Detecting Cyber-Attacks." In 2024 International Conference on Smart

Applications, Communications and Networking (SmartNets), 1–6. IEEE, 2024.

[32] Hammad, Atheer Alaa, Saadaldeen Rashid Ahmed, Mohammad K. Abdul-Hussein, Mohammed R. Ahmed, Duaa A. Majeed, and Sameer Algburi. "Deep Reinforcement Learning for Adaptive Cyber Defense in Network Security." In Proceedings of the Cognitive Models and Artificial Intelligence Conference, 292–97. İstanbul Turkiye: ACM, 2024. https://doi.org/10.1145/3660853.3660930.

[33] Jayaprakash, J. Stanly, Sarangam Kodati, A. Kanchana, Mohammed Al-Farouni, and Ramachandra AC. "An Effective Cyber Security Threat Detection in Smart Cities Using Dueling Deep Q Networks." In 2024 4th International Conference on Mobile Networks and Wireless Communications (ICMNWC), 1–5. IEEE, 2024.

[34] Kheddar, Hamza, Diana W. Dawoud, Ali Ismail Awad, Yassine Himeur, and Muhammad Khurram Khan. "Reinforcement-Learning-Based Intrusion Detection in Communication Networks: A Review." IEEE Communications Surveys & Tutorials, 2024. https://doi.org/10.1109/COMST.2024.3484491.

[35] Nguyen, Thanh Thi, and Vijay Janapa Reddi. "Deep Reinforcement Learning for Cyber Security." IEEE Transactions on Neural Networks and Learning Systems 34, no. 8 (2021): 3779–95. https://doi.org/10.1109/TNNLS.2021.3121870.

[36] Rookard, Curtis, and Anahita Khojandi. "RRIoT: Recurrent Reinforcement Learning for Cyber Threat Detection on IoT Devices." Computers & Security 140 (2024): 103786. https://doi.org/10.1016/j.cose.2024.103786.

[37] Sangoleye, Fisayo, Jay Johnson, and Eirini Eleni Tsiropoulou. "Intrusion Detection in Industrial Control Systems Based on Deep Reinforcement Learning." IEEE Access, 2024. https://doi.org/10.1109/ACCESS.2024.3477415.

[38] Sethi, Kamalakanta, Rahul Kumar, Dinesh Mohanty, and Padmalochan Bera. "Robust Adaptive Cloud Intrusion Detection System Using Advanced Deep Reinforcement Learning." In Security, Privacy, and Applied Cryptography Engineering, edited by Lejla Batina, Stjepan Picek, and Mainack Mondal, 12586:66–85. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020. https://doi.org/10.1007/978-3-030-66626-2_4.

[39] Sewak, Mohit, Sanjay K. Sahay, and Hemant Rathore. "Deep Reinforcement Learning in the Advanced Cybersecurity Threat Detection and Protection." Information Systems Frontiers, August 30, 2022. https://doi.org/10.1007/s10796-022-10333-x.

[40] Tareq, Imad, Bassant Mohamed Elbagoury, Salsabil Amin El-Regaily, and El-Sayed M. El-Horbaty. "Deep Reinforcement Learning Approach for Cyberattack Detection." International Journal of Online & Biomedical Engineering 20, no. 5 (2024). https://doi.org/10.3991/ijoe.v20i05.48229.

[41] Zhang, Yinjun, Mona Jamjoom, and Zahid Ullah. "Double Deep Q-Network next-Generation Cyber-Physical Systems: A Reinforcement Learning-Enabled Anomaly Detection Framework for next-Generation Cyber-Physical Systems." Electronics 12, no. 17 (2023): 3632. https://doi.org/10.3390/electronics12173632.

[42] Zhu, Zhengwei, Miaojie Chen, Chenyang Zhu, and Yanping Zhu. "Effective Defense Strategies in Network Security Using Improved Double Dueling Deep Q-Network." Computers & Security 136 (2024): 103578. https://doi.org/10.1016/j.cose.2023.103578.