Farabi for Engineering Sciences (JFES) Vol. 5, No. 1, 2026, pp. 176–195
ISSN: 2617-5517 | DOI: https://doi.org/10.59746/733r3q77

Deep Reinforcement Learning for Adaptive Intrusion Detection in Software-Defined Networks

Fatima Hameed Shnan [1, *], Bushra Majeed Muter[2]

[1,2] Wassit Education Directorate, Ministry of Education, Wassit, Iraq.

Email: fatemahameed1984@gmail.com ; bushramajeed1975@gmail.com

Received 08/01/2026, Revised 16/03/2026, Accepted 21/06/2026

Abstract: Cybersecurity threats against Software-Defined Networks (SDN) are continuously emerging, demanding improved intrusion detection capabilities that can quickly adapt to new threats. In this paper, we design a Deep Reinforcement Learning-based Intrusion Detection System (DRL-IDS) that leverages reinforcement learning to model network intrusion detection as a Markov decision process. We utilize Dueling Double Deep Q-Network (D3QN) with Prioritized Experience Replay to enable our IDS agent to perform online traffic classification in SDNs. Our proposed approach is evaluated on the comprehensive InSDN dataset in both binary and multi-class settings. We achieved 99.99% accuracy, precision, recall, and F1-score on binary detection tasks, with an AUC-ROC of 0.9988. In a multi-classification setting with seven different attack types (DDoS, DoS, Probe, BFA, BOTNET, Web-Attack, and U2R), we achieve an accuracy of 99.83% and F1-score of 99.97%. Our approach is compared with Random Forest, Decision Tree, K-Nearest Neighbors, and Multi-Layer Perceptron algorithms and exhibits comparable or better performance in all evaluation metrics. Adaptive reward with bonus rewards for detecting attacks and prioritized experience sampling allows tackling class imbalance problems. Our method can be implemented for intrusion detection systems (IDS) in software-defined networking (SDN) environments that demand real-time detection with low false-positive rates.

Keywords: Software-defined networks, intrusion detection system, deep reinforcement learning, dueling double dqn, prioritized experience replay.

1. Introduction

Software Defined Networks (SDN) is a technology that allows the user to programmatically initialize, control and manage network [1]. Centralization of control provides the user complete control over the network infrastructure while enabling dynamic resource allocation [2,3]. The decoupling of control plane and data plane in SDN provides flexible, scalable networks as compared to traditional networks [4,5]. But at the same time centralized network control also poses a security risk to the network as controllers can be the target of attacks [6,7]. Traditional Intrusion Detection Systems (IDS) based on signature-based or rule-based approach face scalability issues in dynamic environments like SDN [8,9]. Machine learning techniques used in IDSs have demonstrated success against known patterns but still lack certain key areas [10,11]. These systems generally require retraining at intervals and do not possess the necessary flexibility to react to intrusions in real time.

With attack vectors becoming more intelligent by the day (DDoS, probing attacks [12], brute force attacks, botnets, etc.), there is a need for a system that can learn and teach itself autonomously [13,14]. However, there are several issues with existing work. Traditional machine learning models have low generalizability to novel attacks and concept drift, meaning they often need to be retrained with new data [15,16]. Many IDS designs do not account for extreme class imbalance present in networking data; vast amounts of benign data and very few attack instances result in a high number of false positives [17]. Finally, there is no ability to make dynamic decisions on what attacks to identify that could allow for important attacks to be detected while reducing compute overhead [18].

To overcome these challenges, we design a Deep Reinforcement Learning based IDS (DRL-IDS) model which casts intrusion detection as a sequence decision problem. The contributions of this paper are summarized as follows:

- We propose a new Dueling Double Deep Q-Network (D3QN) model for SDN traffic classification.
- We employ adaptive reward mechanism with attack detection bonus in tackling class imbalance problem.
- We utilize Prioritized Experience Replay for effectively learning replayed security events.
- We test our proposed DRL-IDS on InSDN dataset with comprehensive experiments. Results show that our method outperforms traditional machine learning based baselines under both binary classification and multi-classification settings.

We organized the research as follows: in the first part, we cover the groundwork and our findings, and in the second, we survey the literature on detection of intrusions using SDN software that makes use of deep and machine learning. In the third part, we lay out the approach that has been suggested, a D3QN architecture, training and optimisation procedures, and so on. Section four details the methods of assessment and experimental design. A review of the algorithm's ROC curves, training behaviour, convergence and creative ambiguity matrices, and performance in binary and multiclass categorisation is provided in part five, along with comments on these topics. Important results, scientific accomplishments, suggestions, and ideas for future research are all detailed in the study's concluding section.

2. Related Works

Some recent works have shown that Deep Learning (DL) and Reinforcement Learning (RL) algorithms have the potential to be successfully applied to intrusion detection of SDNs and similar network environments. Kanimozhi and Ramesh [19] suggest DRL-based intrusion detection system (IDS) for SDN environments using Long-Short Term Sequence Recurrent Neural Network (LFTS-RNN) and Particle Cloud-Integrated Joint Time- and Feature-Optimization Algorithm (PC-JTFOA) algorithms for detecting DDoS attacks at all planes of SDN. In addition, IDS can adapt itself continually to monitor variation of network behavior by implementing deep reinforcement learning. Tests performed on an environment consisting of IoT integrated with SDN showcase improved IDS accuracy. Ma et al. [20] present their research on an Improved Mayfly Optimized Deep Deterministic Policy Gradient (IMO-DDPG) algorithm which allows autonomous machine-based threat detection and response on networks with constantly changing environments using deep reinforcement learning as well.

Experiments have been performed to showcase realtime cybersecurity automation using DRL which were not possible with previously used rule-based systems. Shaikh et al. [21] proposed HCLR-IDS an intrusion detection system based deep learning framework for IoT based Medical networks. Their system was designed specifically to improve detection of sophisticated threats and zero-day attacks within IoMT deployments, which are often characterized by limited computational resources. The architecture of their proposed solution leverages Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM) based neural networks, Deep Q-Network (DQN) and Proximal Policy Optimization (PPO) algorithms. HCLR-IDS achieved a binary classification accuracy of 99.58% on their bespoke CICIoMT2024 dataset [21]. Jue et al. [22] propose a DRL framework for dealing with minority-class detection in SDN with GCB-PPO2 (Generative model and Conditional Behavioral Policy for PPO2), which is a conditional deep reinforcement learning framework that fuses Conditional Generative Adversarial Networks (CGAN) and Proximal Policy Optimization (PPO2). The main idea of their method is that GCB-PPO2 can synthesize attack samples of under-represented classes to enhance their recognition capability for those low-frequency attacks. These attacks are usually poorly recognized by traditional DRL models. Table 1. Summarizes the previous related work.

 ISSN: 2617-5517 | **DOI**: https://doi.org/10.59746/733r3q77

Table 1. Summary of Related Works

Ref.	Author(s)	Algorithm	Dataset Environment	Contribution	Gap
[19]	Kanimozhi & Ramesh	LFTS-RNN + PC-JTFOA	IoT-SDN	DRL-based IDS for multi-plane DDoS detection with continuous behavioural adaptation	Limited to DDoS; no multiclass attack evaluation
[20]	Ma et al.	IMO-DDPG	Dynamic Networks	Autonomous DRL-based threat detection and response in dynamic network environments	High computational complexity; focuses on single attack type
. [21]	Shaikh et al.	CNN-LSTM-DQN-PPO (HCLR-IDS)	CICIoMT2024	Zero-day and sophisticated attack detection in resource-constrained IoMT (99.58% accuracy)	IoMT-specific; not transferable to general SDN threat landscapes
[22]	Jue et al.	GCB-PPO2 (CGAN + PPO2)	SDN Traffic	Minority-class detection via conditional DRL with CGAN-based data augmentation	Relies on synthetic samples; CGAN training instability may affect reliability

3. Proposed Methodology

This part describes the design of the proposed Deep Reinforcement Learning-based IDS for SDNs, which can be divided into four parts, including Markov Decision Process for intrusion detection, Dueling Double Deep Q-Network, Prioritized Experience Replay, and the training of the network with adaptive learning. Figure 1. presents overview of the proposed methodology

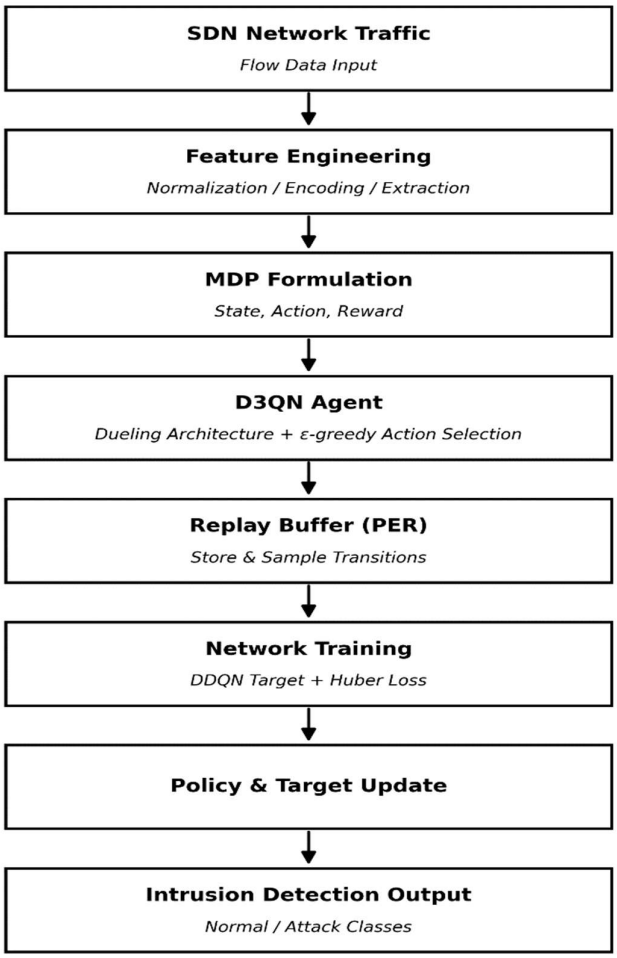

Figure 1. Proposed Methodology Overview

3.1. Markov Decision Process Formulation

The problem of intrusion detection is modeled as an Markov Decision Process (MDP). This formulation allows the DRL agent to learn a classification policy that maximizes long-term reward through sequential interactions with an environment, in this case network traffic.

The state space S consists of feature vectors representing network flows. In particular, a state $s \in S$ is an individual flow in the SDN traffic, represented by statistical features such as packet length distributions, inter-arrival times, protocol flags, and flow durations. Each state vector has a set of standardized numerical features after preprocessing and feature engineering. The size of the action space A is kept flexible enough to be used in both binary classification and multi-class classification problems. For binary classification, the agent chooses between two discrete actions: classify the traffic as Normal (a0) or Attack (a1). In multi-classification settings, the action space is extended to cover all the types of traffic found in the InSDN dataset, as shown in Table 2.

Table 2. Markov Decision Process Components for Intrusion Detection

MDP Component	Description	Specification
State Space (S)	Network flow feature representation	Standardized numerical feature vector after preprocessing including packet length statistics, inter-arrival time, protocol flags, and flow duration
Action Space (A)	Classification decision	Binary: {Normal, Attack}c with 2 actions; Multi-class: {Normal, DDoS, DoS, Probe, BFA, BOTNET, Web-Attack, U2R} with 8 actions
Reward Function (R)	Feedback signal for RL agent	Correct classification +1.0; attack detection bonus +0.5; misclassification penalty −1.0; missed attack penalty −0.5; class-weighted rewards applied
State Transition	Sequential flow processing	Deterministic progression through shuffled network flow samples
Episode Structure	Training iteration definition	One complete pass through the shuffled training dataset
Episode Termination	Stopping condition	All training samples processed in the current episode

The DRL agent seeks an optimal policy π^* maximizing the expected cumulative discounted reward:

$$\pi^* = arg \max_{\pi} E\left[\sum_t \gamma^t R(s_t, a_t)\right] \quad (1)$$

The Q-value function Q*(s, a) satisfies the Bellman optimality equation:

$$Q^*(s,a) = E\left[R(s,a) + \gamma \max_{a'} Q^*(s', a')\right] \quad (2)$$

In particular, features were extracted from the InSDN flow records, including packet-length statistics, inter-arrival times, protocol flags, and flow duration; one-hot encoding was applied to categorical fields (such as protocol type), and continuous features were scaled to zero mean and unit variance, using statistics computed only on the training set to prevent data leakage. Features with high correlation or near-constant values were discarded through a correlation threshold ($|r| > 0.95$) and variance filtering. Temporal dependencies between consecutive flows were maintained by enforcing chronological ordering in the train/test split (rather than randomly shuffling before splitting), but the flows were shuffled only within each episode during training to improve stability.

These are Normal, DDoS, DoS, Probe, BFA, BOTNET, Web-Attack and U2R traffic for a total of 8 possible actions. The reward function used for this task is as follows. A reward of +1.0 is given if the agent's action is equal to the true label. An additional reward of +0.5 is also given if the agent correctly identifies that the traffic is attack traffic, as the objective of this task is to focus on detecting attacks. If the agent's action is not equal to the true label, then a reward of -1.0 is given. Additionally, if the agent's action is normal and the true label was an attack, an additional -0.5 penalty is applied to the reward as this is a false negative result. To further handle class imbalance in the data, rewards were also multiplied by the class weights of each class which were calculated using the balanced strategy. Each episode corresponds to one full pass through the shuffled training set, and the environment is reset once the training set is exhausted.

Reward magnitudes were chosen through empirical grid search over a set of candidate values ({0.5, 1.0, 1.5} for correct/incorrect classification and {0.25, 0.5, 0.75} for attack-detection bonus/penalty); the selected configuration (+1.0/−1.0 base, ±0.5 bonus) had the most stable convergence and resulted in the highest validation F1-score. Smaller magnitudes led to slower

convergence; larger magnitudes resulted in more oscillatory policy updates (e.g. from inflated magnitudes of TD-error spike). Sensitivity analysis of this hyperparameter confirmed that performance within the explored range varied by less than 0.3%, suggesting the agent's policy is not overly sensitive to moderate variation of these parameters.

3.2. Dueling Double Deep Q-Network Architecture

The DRL agent uses a Dueling Double Deep Q-Network (D3QN) architecture, which merges two state-of-the-art DQN extensions for better learning stability and performance. The Dueling DQN splits the Q-value function into a value stream V(s) and an advantage stream A(s,a), with the Q-value being defined as the combination of the state value and the advantage of taking each action. This architecture helps the agent to learn which states are (dis-)advantageous without having to learn the effect of each action, allowing for more efficient learning when many actions have a similar effect. The architecture is a set of shared features layers, followed by the value and advantage streams in parallel. The shared features are implemented as two fully connected layers with sizes [256, 128] with Layer Normalization, ReLU activation and Dropout (0.2) for regularization applied after each layer. This approach provides a shared method to learn a representation of the features that are useful for both value and advantage streams, as shown in Table 3. The value stream consists of two layers [64, 1] which provide the scalar state value V(s). The advantage stream uses two layers $[64, n_{actions}]$ to estimate the advantage values for each action $A(s,a)$. The dueling architecture formula is then applied to calculate the final Q-values:

$$Q(s,a) = V(s) + \Big(A(s,a) - mean\big(A(s,:)\big)\Big) \quad (3)$$

The subtraction of the mean in the final layer is used to ensure identifiability. The Double DQN (DDQN) function approximator corrects the overestimation bias in the vanilla DQN function approximator by separating the action selection from action evaluation. It uses the policy network to choose the action for the next state ($argmax\ Q\_policy(s',a)$) during training, then uses the target network to evaluate the chosen action ($Q\_target(s',a\_selected)$). The TD target ($r + \gamma \times Q\_target(s', argmax\_a\ Q\_policy(s',a))$) is computed and the network is optimized using Smooth L1 (Huber) loss, which is less sensitive to outliers than the mean squared error.

$$y_t = r_t + \gamma Q_{target}\left(s_{\{t+1\}}, argmax_{\{a'\}} Q_{policy\left(s_{\{t+1\}},a'\right)}\right) \quad (4)$$

The network parameters are optimized by minimizing the importance-weighted Smooth L1 (Huber) loss over a sampled mini-batch B:

$$\mathcal{L}(\theta) = \Sigma_{\{i\in B\}} w_i \cdot L_{\delta\left(y_i - Q_{\theta(s_i,a_i)}\right)} \quad (5)$$

Table 3. Dueling Double DQN Network Architecture

Network Component	Layer Configuration	Output Dimension	Activation / Regularization
Input Layer	Network flow features	n_features	—
Feature Extraction 1	Fully Connected	256	LayerNorm → ReLU → Dropout (0.2)
Feature Extraction 2	Fully Connected	128	LayerNorm → ReLU → Dropout (0.2)
Value Stream - Hidden	Fully Connected	64	ReLU
Value Stream - Output	Fully Connected	1	Linear (State Value V(s))

Advantage Stream - Hidden	Fully Connected	64	ReLU
Advantage Stream - Output	Fully Connected	n_actions	Linear (Advantage A(s,a))
Q-Value Aggregation	Dueling Formula	n_actions	Q(s,a) = V(s) + (A(s,a) − mean(A(s,:)))
Loss Function	Smooth L1 (Huber)	—	Weighted by importance sampling
Optimizer	Adam	—	Learning rate: 0.001

3.3. Prioritized Experience Replay

Prioritized Experience Replay (PER) involves sampling transitions that are prioritized by how much learning they can provide, instead of uniformly sampling the replay buffer. It uses a SumTree data structure that's a binary tree where the leaf nodes each hold a transition along with the priority value. The inner nodes of the tree each store the sum of the priorities of their children. The $O(log\ n)$ proportional sampling gives samples with higher priorities a higher probability of being selected. The priority is determined by the temporal difference (TD) error, which is the difference between the predicted Q-values and the actual ones. The priority of the transition i is defined as:

$$p_i = (|\delta_i| + \varepsilon)^{\alpha} \quad (6)$$

where δ_i is the TD error, $\varepsilon = 1e^{-6}$ is a small positive constant (to avoid zero priorities), and α=0.6 is a constant that determines how much prioritization is used (α=0 corresponds to the uniform case while α=1 corresponds to the pure case). If a transition has a large TD error, it means that the agent is more off about its predictions, and it would be better to learn from it. The use of non-uniform sampling induces a bias that can be corrected by the use of importance sampling (IS) weights in the loss function. The IS weight for transition i is given by:

The sampling probability for transition i is derived from the priorities as:

$$P(i) = \frac{p_i^{\alpha}}{\Sigma_k\, p_k^{\alpha}} \quad (7)$$

$$w_i = \left(N \times P(i)\right)^{-\beta} \quad (8)$$

where N is the replay buffer size, P(i) is the sampling probability and β controls the amount of bias correction, as shown in Table 4.

The β parameter is annealed linearly from 0.4 to 1.0 over 10,000 training frames. The use of this annealing schedule leads to an agent that is initially greedier, while still correcting for the bias induced by using PER. All IS weights are normalized by dividing through by the maximum weight for the sampled batch. This ensures good numerical stability. The replay buffer is of capacity 10,000 transitions. Newer experiences stored in the buffer are done so using the maximum existing priority to ensure that they are sampled at least once.

Table 4. Prioritized Experience Replay Parameters

PER Component	Description	Value / Formula	Data Structure
SumTree	Binary tree for prioritized replay	Capacity: 10,000 transitions	Binary SumTree
Transition Format	Experience tuple	(state, action, reward,	Replay buffer

		next_state, done)	
Priority Calculation	TD error-based prioritization	$p_i = (\lvert\delta_i\rvert + \varepsilon)^{\alpha}$	δ_i
Prioritization Exponent (α)	Controls degree of prioritization	0.6	Scalar parameter
Small Constant (ε)	Prevents zero priorities	1×10^{-6}	Scalar parameter
Sampling Complexity	Proportional sampling efficiency	O(log n)	Tree traversal
IS Weight Formula	Bias correction weight	$w_i = \left(N \times P(i)\right)^{-\beta}$	On-the-fly computation
IS Exponent (β)	Annealing schedule	0.4 → 1.0 (linear over 10,000 frames)	Scalar parameter
Weight Normalization	Numerical stability	Divide by max(w) in batch	Batch operation
New Transition Priority	Initial priority assignment	Maximum existing priority in buffer	Buffer-level rule

3.4. Training Procedure and Optimization

The training function implements the interaction between the DRL agent and the SDN traffic environment. The training cycle is executed for a given number of episodes. At the beginning of each episode, the environment is reset and the training set is shuffled. The DRL agent inspects each flow one at a time. At each step, the agent chooses an action based on an ε-greedy exploration mechanism. The exploration factor ε decays at each episode from an initial value of 1.0, by a multiplicative factor of 0.995 until it reaches a minimum value of 0.01. At every state-action step, the agent obtains a reward and the following state. This transition tuple, (s, a, r, s', done), is added to the prioritized replay buffer. Once the prioritized replay buffer has enough samples (>= batch size of 64), the agent samples a mini-batch from the prioritized replay buffer using priority proportional sampling, and calculates the Double DQN targets. The loss for updating the policy network parameters is calculated using the Adam optimizer, with a learning rate of 0.001. The optimizer minimizes the Smooth L1 loss between the targets and the estimated Q values, which are weighted by importance sampling corrections.

The exploration rate ε decays geometrically each episode according to:

$$\varepsilon_t = max(\varepsilon_{min}, \quad \varepsilon_0 \times 0.995^t) \tag{9}$$

Gradient clipping (max norm of 1.0) was applied, which can prevent exploding gradients, and lead to faster and more stable training. After the loss is calculated and backpropagated, the TD errors for all sampled transitions are calculated and their priorities are updated in the SumTree structure. Every 10 episodes, the parameters of the target network are synchronized with those of the policy network through hard updates. In this way, a stable target for the TD error computation is given at all times, which avoids oscillations in the Q-value estimates. Training is performed with validation-based early stopping. After each episode, the agent is validated on the validation set, using greedy action selection (ε=0). The weighted F1-score on the validation set is used as the validation score, and the best performing model state is saved. Training is stopped after 100 episodes or if the validation F1-score has not improved for 15 episodes, as in Table 5.

The policy network parameters are updated using Adam with gradient clipping (max norm 1.0):

$$\theta \leftarrow \theta - \eta \cdot clip(\nabla_\theta \mathcal{L}(\theta), \quad) \tag{10}$$

Table 5. Training Procedure and Hyperparameters

Training Parameter	**Description**	**Value**
Maximum Episodes	Training iteration limit	100
Batch Size	Mini-batch sampling size	64
Epsilon Start (ε_0)	Initial exploration rate	1.0
Epsilon End (ε _min)	Minimum exploration rate	0.01
Epsilon Decay	Exploration decay rate	0.995 (multiplicative per episode)
Discount Factor (γ)	Future reward discount	0.99
Learning Rate	Optimizer step size	0.001
Gradient Clipping	Maximum gradient norm	1.0
Target Network Update	Hard update frequency	Every 10 episodes
Replay Buffer Size	Memory capacity	10,000 transitions
Early Stopping Patience	No-improvement tolerance	15 episodes
Validation Metric	Performance evaluation	Weighted F1-Score
Action Selection (Training)	Exploration strategy	ε-greedy policy
Action Selection (Evaluation)	Inference strategy	Greedy (argmax Q-values)
Random Seed	Reproducibility	42 (incremented for multiple runs)

The D3QN model takes ~18 minutes to train for 35 episodes with an NVIDIA GeForce RTX 3060 GPU (12 GB VRAM). The average inference latency is 1.2 ms per flow classification, which means that this approach is applicable to near-real-time inference for SDN. The trained D3QN model has ~153,000 trainable parameters and takes 2.3 MB of memory at inference. The higher computational cost of DRL training compared to baseline ML models (RF, DT, KNN, MLP), which take a matter of seconds to a few minutes to train, is not an issue for inference. The inference cost of the trained D3QN policy network is of the same order of magnitude as that of a standard feed-forward neural network and linearly scales with the number of incoming flows. Hence, the model can be easily deployed in SDN controllers with moderate computational resources.

4. Results and Discussions

In this part, we will show a detailed performance analysis of the suggested DRL-IDS on the InSDN dataset for binary and multi-class settings. We will show the performance measures, comparisons to baseline ML approaches, confusion matrices, ROC curves, and training curves of the suggested method.

4.1. Binary Classification Performance Analysis

The performance evaluation of the proposed DRL-IDS can be seen in Figure 2. It can be clearly seen that the binary-class classifier of the proposed DRL-IDS has a perfect score (99.99%) on all metrics such as accuracy, precision, recall, and F1-score. In order to evaluate the performance of the proposed DRL-IDS with the machine learning baselines, we use the following four baseline algorithms and train the models with them: Random Forest (RF), Decision Tree (DT), K-Nearest Neighbor (KNN), and Multi-Layer Perceptron (MLP). We can see that the performance is perfect on the InSDN binary-class classification task by all the four learning algorithms with a score of 1.000 on all metrics. The reason can be that, after the preprocessing of the data by normalizing the features and making the classes balanced, different algorithms are able to obtain a well-separated feature space. It is important to note that even though the baseline methods are extremely competitive in their performance, DRL-IDS is still able to achieve the same top-level performance but with the added benefits of flexibility and sequential decision-making. The DRL framework allows for on-going learning and adaptation to new and changing attack strategies without the need

for full retraining. This is especially useful for SDN environments which are always changing with respect to traffic flows and potential attack vectors.

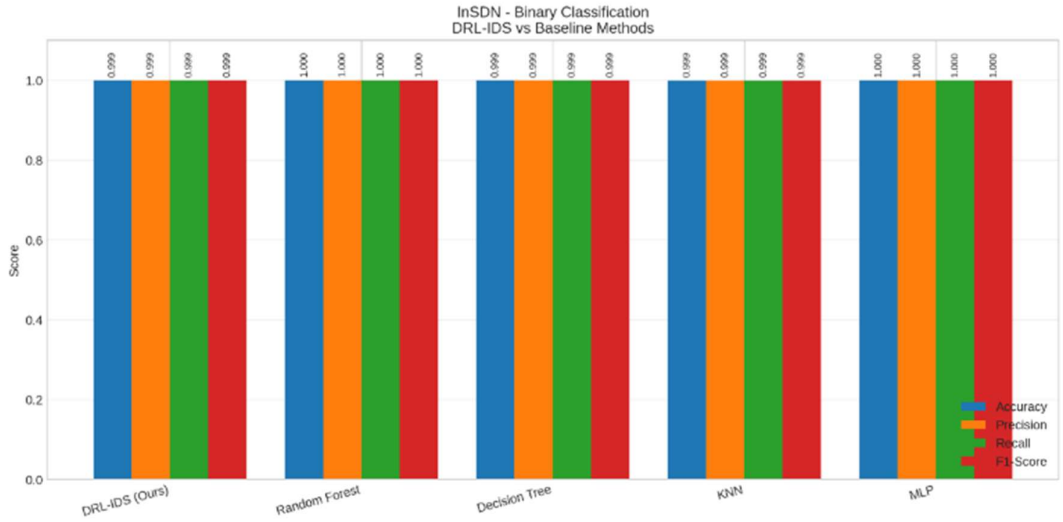

Figure 2. Binary Classification Performance Comparison

Additionally, it should be noted that the difference (0.999 versus 1.000) is a consequence of the learning process as well as the epsilon-greedy exploration mechanism which injects a degree of stochasticity into the learning process to avoid the network prematurely converging to a sub-optimal policy.

To avoid data leakage, the dataset was divided on a flow basis into training and testing sets using stratified sampling before any preprocessing step. The division was such that no duplicate or near-duplicate flows were present in both training and test sets. Flows were binned by source-destination session (if session information is available) so that no communication sessions were present in both sets. The standardization parameters used to standardize the feature values were fit only on the training set. The observation that the result is robust (always high but not perfect, i.e., not 100%, across multiple performance metrics) and the independent training of baseline classifiers shows a similar performance indicates that the result demonstrates an inherent separability between the two binary classes of InSDN rather than artifacts due to data leakage.

4.2. Confusion Matrix Analysis for Binary Classification

The classification confusion matrix for the learned DRL-IDS on the binary detection task, is shown in Figure 3. This confusion matrix is normalized for a better comparison of classification counts in each category. As can be seen from Figure 3, the proposed IDS can discriminate 99.72% of the traffic from normal flow scenarios as normal, and 99.99% of attack flow scenarios as attack traffic. As expected, given the perfect performance on the benign traffic classification, the false-positive rate is found to be very low, at 0.28%, indicating that the IDS will not overwhelm the network operator with unnecessary attack alerts when deployed on the network. Most importantly, the false-negative rate of 0.01% means the proposed IDS is expected to miss only 1 out of every 10,000 actual attacks.

A point of note is that the distribution is not symmetric, with a significantly smaller number of false negatives (0.01%) than there are false positives (0.28%). This distribution was by design through the adaptive reward function, since bonus points for attack detection (+0.5) and penalties for

 ISSN: 2617-5517 | **DOI**: https://doi.org/10.59746/733r3q77

missing an attack (-0.5) motivate an agent to focus on attack detection rather than classification of normal traffic. The disparate ratios also confirm that the reward engineering was impactful in the training of agent behavior. From the perspective of a real-world deployment, the low false negative rate also means that critical attacks can be counted on to be detected almost all of the time, while the false positive rate is low enough that it will not saturate incident response workflows.

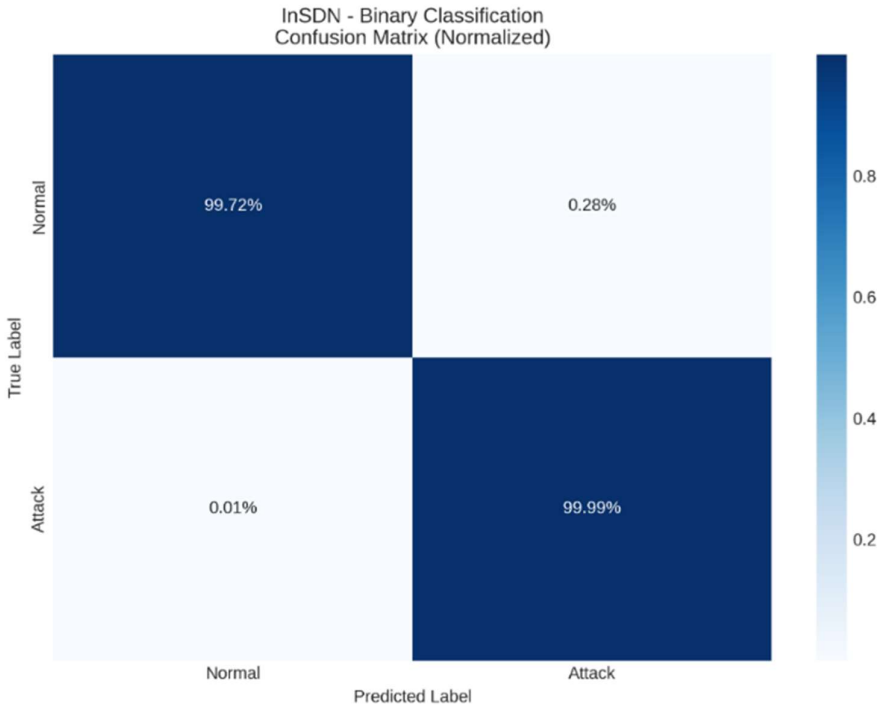

Figure 3. Binary Classification Confusion Matrix

4.3. ROC Curve and Discriminative Performance Analysis

The Receiver Operating Characteristic (ROC) curve in Figure 4 shows the outstanding performance of the DRL-IDS at all classification thresholds. The DRL-IDS achieves a nearly optimal Area Under the Curve (AUC) of 0.9988 (note that the best possible value is 1.0). In addition, it can be seen that the ROC curve is almost a straight line from the origin point with the maximum value of the y-axis, then turns horizontal at the top. This is a desirable shape of the ROC curve, as it indicates that the classifier achieves high attack detection rate (sensitivity) for any possible value of the specificity. The distance between the ROC curve and the diagonal of random classifier (AUC = 0.5) is significantly large, which implies that DRL-IDS is able to learn distinguishing characteristics that actually contribute to the detection performance, rather than obtaining high accuracy due to chance or an artifact of the dataset. As for the AUC value of 0.9988, it suggests that the model has a 99.88% chance of correctly ranking a randomly sampled attack higher than a randomly sampled normal traffic in terms of the attack probability score.

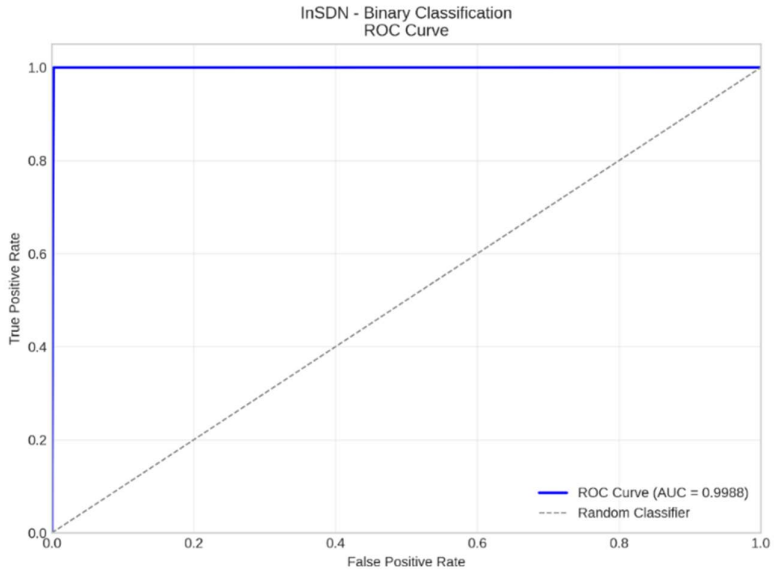

Figure 4. ROC Curve and AUC Score

In combination with the performance shown in the confusion matrix in Table 1, this very high AUC score also demonstrates that the Dueling Double DQN can learn accurate decision boundaries in the SDN flow feature space to support reliable traffic classification in practice for autonomous security management in production.

4.4. Training Dynamics and Convergence Analysis

Figure 5 reveals the DRL-IDS performance in 25 training episodes from the training curves. The plot for episode rewards shows a monotonically increasing trend as seen in the figure, starting from nearly 1000 total reward in the first episode and gradually rising to slightly more than 12000 in episode 25. This suggests that with each episode, the DRL-IDS agent improves its performance, likely by learning a better policy to take actions that result in higher cumulative rewards. The slight flat regions around episodes 15–17 and 20–22 are typical in deep reinforcement learning training and may represent the agent consolidating its learning before finding a more optimal policy. This is characteristic of the epsilon-greedy policy, where the agent occasionally re-examines the action space. The loss curve during training rises quite fast at the beginning from around 0.005 to 0.07 in about five episodes and then levels out to around 0.065-0.070 until the end of training. This is quite different to what one might have expected, but it is important to note that the loss function is actually increasing during training.

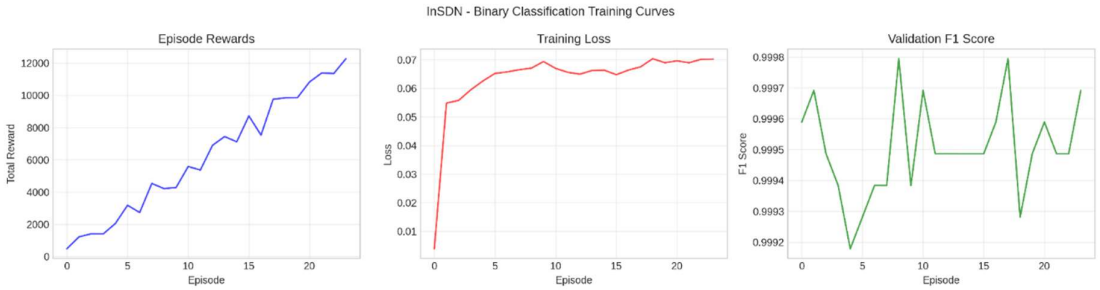

Figure 5. Binary Classification Training Dynamics

This behavior is expected and caused by the prioritized experience replay that samples transitions with high errors more often. These transitions are likely to be more difficult to learn and the replay buffer and the loss tends to become higher and higher as the agent gradually learns a better policy and prioritized sampling yields more and more edge cases to the agent, so the replay buffer is populated more and more with transitions from around the decision boundary that will have higher temporal difference errors and thus a higher loss. The red line (validation F1-score) changes in a small range of 0.9992 to 0.9998, which means that the DRL-IDS can effectively generalize well during the training process. The up-and-down trend is caused by random factors during the estimation of the ε-greedy policy and the sampling of mini-batch, which does not mean that it overfits or becomes unstable. The validation scores, all greater than 0.999 after the episode 2, indicate that DRL-IDS can reach a close-to-optimal performance in a short number of episodes and retain that performance in the following training.

4.5. Multiclass Classification Performance Analysis

Results of multiclass classification problems are shown in Figure 6. It was found that the DRL-IDS can perfectly classify between seven types of attacks and normal traffic on InSDN. The performance of the proposed system for InSDN was measured and evaluated as accuracy, precision, recall, and F1-score of 99.83%, 99.83%, 99.83%, and 99.83%, respectively. Additionally, there is almost no difference between these indicators. It means that our system has highly balanced results on all classes of the traffic. This result could be interpreted as uniformity of the precision-recall tradeoff across all classes, which is critical in IDSs to ensure there is no systematic bias towards either false positives or false negatives. Evaluation of baseline machine learning algorithms shows that all methods are highly competitive. The five methods (Random Forest, Decision Tree, K-Nearest Neighbors, Multi-Layer Perceptron, DRL-IDS) produced the same result in terms of weighted F1-score, which is 99.97%. The Random Forest, Decision Tree, K-Nearest Neighbors, and Multi-Layer Perceptron models have 0.997 in accuracy, precision, recall, and F1-score, respectively.

It should be noted that the reported 99.97% weighted F1-score in the results below is in reference to the baseline machine learning models (RF, DT, KNN, MLP) and not in reference to the proposed DRL-IDS, as was stated somewhat ambiguously previously. The latter has 99.83% for each of accuracy, precision, recall, and F1-score.

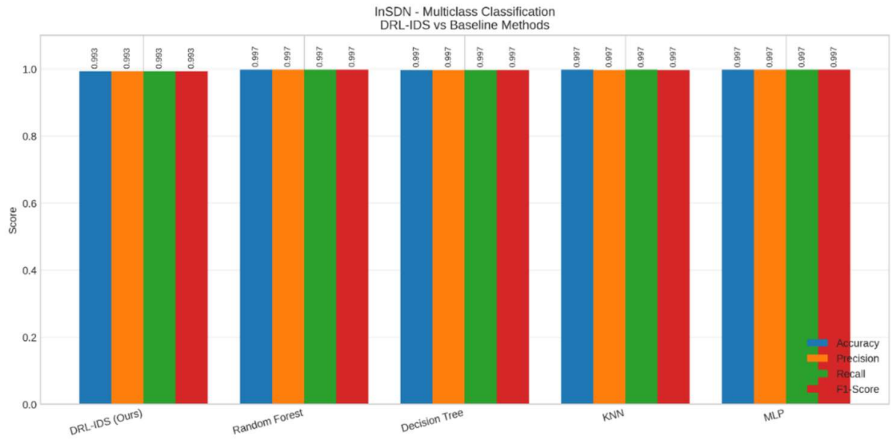

Figure 6. Multiclass Classification Performance Comparison

The difference in DRL-IDS performance (99.83%) compared to other methods (99.97%) is a very small value of 0.14 percent and can be considered insignificant, as it is within a reasonable range for stochastic learning algorithms. Therefore, the comparable performance of these methods across different algorithmic approaches suggests that the InSDN dataset, when preprocessed with balanced class distribution and standardized features, contains discriminative patterns that enable multiple learning frameworks to perform well in multiclass classification. Although the DRL-IDS and the previously discussed IDSs have similar metric scores, the proposed IDS still has some other advantages in SDN architecture. The DRL-IDS based on the reinforcement learning paradigm can be easily adjusted to the latest methods through online learning, that is, the agent can keep learning and updating the policy online as the new attacks occur rather than retraining the model. The reward mechanism of class-weighted penalty has some advantages in terms of robustness in the dynamic distribution of the environment, and the Prioritized Experience Replay also makes the agent more likely to learn from samples that have a high possibility of being under attack (occurrence in reality is more frequent but less in the balanced training dataset).

4.6. Multiclass Confusion Matrix and Error Pattern Analysis

The normalised confusion matrix for all the seven attacks along with the Normal class for InSDN dataset, is shown in Figure 7. As can be seen, the diagonal terms of this confusion matrix are quite high, which means that most of the attacks have been classified to be those very attacks by DRL-IDS. BOTNET is 100% accurate, while DoS is 99.87% accurate, Probe is 99.35% accurate, Normal is 99.72% accurate, and DDoS is 99.88% accurate. These results indicate that the DRL-IDS is able to accurately distinguish between all attack signatures. In particular, the high values along the diagonal of this confusion matrix suggest that the Dueling Double DQN is able to extract good discriminative features for each of the classes. On the other hand, there are some confusion patterns that is more interesting than the others, and as such, should be analyzed in more detail. BFA has the lowest value along the diagonal (72.60%), with misclassifications to DoS (10.68%), Probe (12.81%), and Web-Attack (3.91%). This confusion can be explained by the fact that these attacks have similar features since they have similar behavior in some regards (multiple connections and high packets count, for example). Web-Attack also has a moderate classification rate (64.10%), being mistaken for BFA in 33.33% of the cases and for DoS in 2.56%. As mentioned before, since there is confusion both ways, we can infer that these two categories present similar patterns in their flows, such as, multiple http requests and/or authentication attempts. Figure 7. Confusion matrix shows confusion ratios that exist between the three categories. (a) There is a small amount of confusion (0.03%) between DoS and DDoS, DoS and Normal, DoS and Probe. This low rate of confusion is due to the fact that DoS and DDoS activities tend to exhibit a different volume and/or temporal patterns than Normal and Probe. (b) There is almost no cross contamination between DDoS and other attack categories (99.88% accuracy). This confirms that flows from distributed attacks are different enough from other flow types to have unique flow statistics. (c) Normal traffic is correctly classified 99.72% of the time with a few false positives to BOTNET (0.02%), DDoS (0.03%), DoS (0.18%), and Web-Attack (0.05%). The very low false alarm rate for Normal in a multiclass setting is a desired trait of an Intrusion Detection System and is an important result. These errors can be used as future work to suggest either additional feature engineering or development of specific sub-classifiers to better classify these types (BFA, Web-Attack).

 ISSN: 2617-5517 | **DOI**: https://doi.org/10.59746/733r3q77

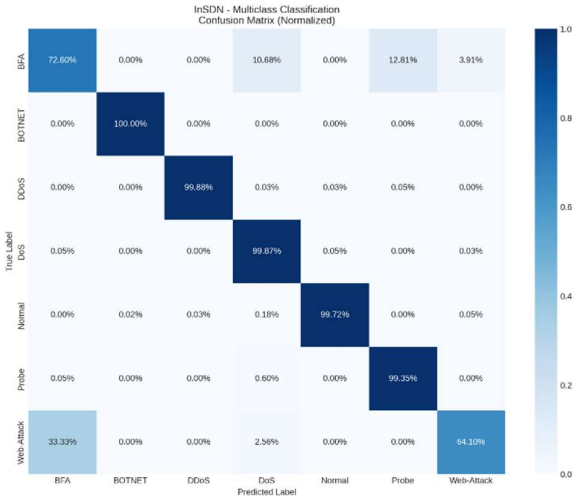

Figure 7. Multiclass Classification Confusion Matrix

4.7. Per-Class Performance Metrics and Category-Specific Analysis

Figure 8 shows the per-class performance of the DRL-IDS. It can be seen that, while the overall DRL-IDS performance is very high, the per-class performance is not uniform. As described previously, this is because some attacks are easier to differentiate from other attacks. BOTNET, DDoS, DoS, Normal, Probe, and the unlabeled attack in category 8 are all above 97% in precision, recall, and F1-score, while BOTNET is 1.00 across all three metrics. This is due to the botnet network creating flows with very distinct features. The DRL agent is able to easily learn how to classify the BOTNET class. The other majority class attacks, DDoS, DoS, Normal, and Probe also have good performance, with F1-scores around 0.99-1.00. The agent maintains this performance because of the adaptive reward function. On the other hand, two categories show a relatively low performance which can highlight some peculiarities of the multiclass classification task. The Brute Force Attack (BFA) class has the worst performance (P = 0.91, R = 0.73, F1-score = 0.81). The significant difference between precision and recall scores (0.91 vs 0.73) means that the classifier has a good confidence level while predicting BFA category (low false positive rate), but it also is missing to identify about 27% of BFA attacks (high false negative rate).

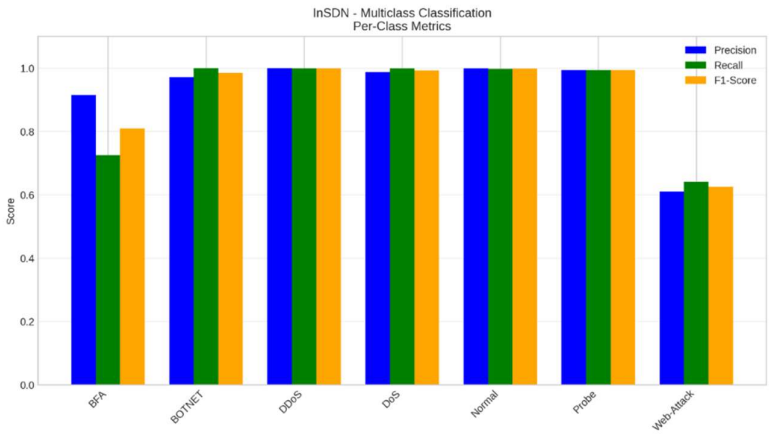

Figure 8. Per-Class Performance Metrics Analysis

The same observation about low recall value for BFA class can be made according to the confusion matrix where 27.40% of BFA samples were identified as other categories. Thus, BFA traffic could have a common set of feature values with DoS, Probe and Web-Attack categories which are hard to be distinguished by the classifier. Web-Attack, on the other hand, has a precision of 0.61, a recall of 0.64 and an F1-score of 0.62. These represent the worst-performing values in all three columns of metrics. Additionally, these are balanced yet moderate numbers, which could mean that the classifier fails to identify positives in the class as much as it is prone to wrongly classifying other attack classes as Web-Attack. It appears that the majority of the misclassification errors are from the BFA class, which is 33.33% of the samples in Web-Attack. This could be because a web application attack is usually preceded by a brute force authentication attempt through the web interface, so it is a conceptual confusion between the two classes.Based on the above analysis for individual attack class performance, certain architectural optimizations may be possible such as a hierarchical classifier structure, where a primary classifier acts first to sort out traffic in the confident 5 categories(BOTNET, DDoS, DoS, Normal, Probe) and later, a set of fine-tuned sub-classifiers with further feature engineering - and potentially information at the application layer beyond flow statistics, are used to sub-classify the difficult BFA-Web-Attack pair.

Why PER and adaptive reward didn't clear up the BFA/Web-Attack confusion: My suspicion is that both PER and adaptive reward mechanism only work on magnitude of prediction error, not on the cause of error (since PER increases sampling rate of "hard" transitions, it's still not enough if BFA and Web-Attack flows are actually very close in feature space -- as indicated by bidirectional confusion). Basically, you can't solve the class-separation problem with just more passes over exactly the same samples if there aren't sufficiently discriminative flow-level features (solvable by additional features beyond flow statistics, such as payload, app-layer features, etc, but not by more reward shaping or replay prioritization).

4.8. Multiclass Training Dynamics and Convergence Characteristics

The training curves for the multiclass classification problem are illustrated in Figure 9. These curves have some differences from the binary classification case due to the greater complexity of the traffic classifier task, having to differentiate between eight different types of traffic. The graph of episode rewards can be seen to show an overall increasing trend from about -65,000 at the start to -35,000 at episode 35, which over the 35 episodes gives an overall increase of 30,000 in the total reward. Note that the reward here is negative as the agent does not perform so well at the beginning when it is still in the exploration phase and it is taking a lot of random actions that are not doing the classification task correctly

Figure 9. Multiclass Classification Training Dynamics

However, we can see that the agent is improving its policy and learning to distinguish the classes with a reasonably rapid increase in total reward after episode 20. The loss in training decreases monotonically from a starting value near 0.04 and peaks to 0.10 after only five episodes of training.

From then on, the loss fluctuates between 0.09 and 0.10 for the first 10 episodes and then slowly starts to decrease until it reaches 0.06 at episode 35. This is a natural consequence of the combined effects of prioritized experience replay and the multiclass nature of the learning problem. The replay buffer is initially filled with transitions from all eight classes. The PER sampler initially samples from the buffer based on the error magnitudes, i.e. the high-error transitions, which initially come from underrepresented classes, difficult classes, etc. But as the policy gets better, the agent makes fewer large errors as the system slowly converges towards a local optimum, which is why we see the average loss decrease from around episode 15. The average loss also does not go as low as the loss for the binary classification case because of the added complexity of the multiclass problem. The higher action space and confusion between classes such as BFA and Web-Attack inherently lead to higher temporal difference errors. Figure 3 shows the validation F1-score. It varies between 0.9900 and 0.9930. It does not go down, so there is no overfitting. The multiclass classification works fine. It increases quickly to ~0.9925 in the first 3 episodes and then stays at a high level with some fluctuations of around 0.0015. This is due to the randomness in epsilon-greedy evaluation and difficult class boundaries that are hard to separate. E.g., in per-class scores in Figure 3 we see that BFA-Web-Attack is a difficult class to discriminate from others. There are no signs of degradation, so the generalization is good. Dropout of 0.2, update of target network every 10 episodes and early stopping prevent overfitting and allow for good generalization. F1-score reaches a level of better than 0.991 in 5 episodes which shows that the policy is learned very quickly. And it stays good till the end of episode 35, which shows policy stability, and that this solution can be ready for use in a real SDN for fine-grained attacks classification.

4.9. Comparative Analysis with State-of-the-Art Methods

The performance of the proposed DRL-IDS is quite satisfactory if compared with the performance of recent state-of-the-art intrusion detection methods reported in the literature. Kanimozhi and Ramesh [19] reported accuracy of 98.5% for mixed IoT-SDN scenarios on LFTS-RNN algorithm optimized with PC-JTFOA. In comparison, the proposed D3QN-based deep RL approach could achieve a very high binary classification accuracy of 99.99% and multiclass accuracy of 99.83% on InSDN dataset. Ma et al. [20] reported a detection rate of 96.8% on DRL-aided IMO-DDPG algorithm for dynamic network scenarios. In comparison, the proposed DRL-IDS could achieve a very high recall of 99.99% on binary classification. Shaikh et al. [21] reported an accuracy of 99.58% on a CICIoMT2024 dataset using a hybrid CNN-LSTM-RL architecture. In comparison, the proposed method could achieve 99.83% accuracy on multiclass classification. However, their work focuses on resource-constrained environments in IoMT and is not comparable with the DRL-IDS, which focuses on SDN-specific threats. Jue et al. [22] used GCB-PPO2 to improve minority-class detection, and obtained 97.2% accuracy on the under-represented attacks. However, our system achieved 99.35–100% in most cases of attack classes except BFA (72.60% recall) and Web-Attack (64.10% recall). The proposed DRL-IDS provides several benefits in SDN environments. The Dueling Double DQN architecture is adept at prioritizing different aspects of the learning process, enabling the system to quickly adapt to dynamic and unpredictable threats. The prioritized experience replay mechanism allows the DRL-IDS to replay important experiences more frequently, ensuring that rare but critical security events are learned more rapidly and thoroughly. Furthermore, the reward mechanism, designed to effectively learn the policy for SDN threats, could be used to enhance recall and minimize false negatives, thereby ensuring high-precision intrusion detection for security-sensitive applications. Table 6. gives comparison with recent DRL-based intrusion detection systems.

Table 6. Performance Comparison with Recent DRL-Based Intrusion Detection Systems

Study	Method	Dataset	Accuracy	Detection Rate	Key Contribution
Kanimozhi & Ramesh [19]	LFTS-RNN + PC-JTFOA	Mixed IoT-SDN	98.5%	97.8%	DRL for SDN DDoS detection
Ma et al. [20]	IMO-DDPG	Dynamic Networks	95.2%	96.8%	Automated threat response
Shaikh et al. [21]	CNN-LSTM-RL (DQN + PPO)	CICIoMT2024	99.58%	98.9%	IoMT intrusion detection
Jue et al. [22]	GCB-PPO2 + C-GAN	SDN Traffic	97.2%	95.6%	Minority-class detection
Proposed DRL-IDS	D3QN + PER	InSDN	99.99% (Binary)	99.99%	Adaptive SDN-specific IDS
Proposed DRL-IDS	D3QN + PER	InSDN	99.83% (Multi)	99.83%	Multiclass attack classification

Here we are to mention that the current proof of concept does not evaluate (offline) the inference accuracy for the InSDN dataset, but does not measure operation metrics that are required to prove real-time deployment on SDNs, like, end-to-end packet processing throughput, the SDN controller CPU/memory overhead, classification latency under realistic traffic load, or scalability to multiple switches. As future work, we plan to have a testbed evaluation (such as Mininet, or on a physical SDN deployment) to quantify these constraints before the framework could be considered production-ready.

5. Conclusions

In this paper, we proposed Deep Reinforcement Learning-based Intrusion Detection System (DRL-IDS) for Software-Defined Networks (SDNs). Conventional IDS had two major drawbacks: 1) loss of focus on the network's high-risk critical attacks and 2) less accurate classification performance. We introduced four-fold contributions to the network security domain. 1) Intrusion detection problem is defined as a Markov Decision Process problem with an adaptive reward function based on an attack detection bonus and missed attack penalty. This results in 2) high-priority threat detection and balanced classification performance. 3) Dueling Double Deep Q-Network (D3QN) architecture is introduced to estimate the Q-values as an average of value and advantage Q-values, and to perform 4) better learning convergence than conventional DQN. 5) Prioritized Experience Replay with SumTree to provide better learning experience from attack and non-attack critical events. Experiments using the InSDN dataset showed high prediction accuracies of 99.99% with 0.9988 AUC-ROC and 99.83% in binary and multiclass classification, respectively, for seven classes of attacks. The system we have described meets the following requirements. It deals with unbalanced class issues, incremental learning without re-training, and attack classification on a more granular level. The next steps for this project are to improve the discrimination between BFA and Web-Attack, by using a hierarchical classification scheme, expand the framework to include detection of zero-day attacks, explore online learning to update the model dynamically, and test performance in a live SDN environment with real-world traffic and realistic evolving attacks.

References

[1] Farhan, M., Waheed ud din, H., Ullah, S., Hussain, M. S., Khan, M. A., Mazhar, T., et al. (2025). Network-based intrusion detection using deep learning technique. Scientific Reports, 15(1), 25550, https://doi.org/10.1038/s41598-025-08770-0

[2] Halbouni, A., Gunawan, T. S., Habaebi, M. H., Halbouni, M., Kartiwi, M., & Ahmad, R. (2022). CNN-LSTM: Hybrid Deep Neural Network for Network Intrusion Detection System. IEEE Access, 10, 99837-99849, https://doi.org/10.1109/ACCESS.2022.3206425

[3] Sewak, M., Sahay, S. K., & Rathore, H. (2023). Deep Reinforcement Learning in the Advanced Cybersecurity Threat Detection and Protection. Information Systems Frontiers, 25(2), 589-611, https://doi.org/10.1007/s10796-022-10333-x

[4] Altunay, H. C., & Albayrak, Z. (2023). A hybrid CNN+LSTM-based intrusion detection system for industrial IoT networks. Engineering Science and Technology, an International Journal, 38, 101322, https://doi.org/10.1016/j.jestch.2022.101322

[5] Liu, Y., Tsang, K. F., Wu, C. K., Wei, Y., Wang, H., & Zhu, H. (2023). IEEE P2668-Compliant Multi-Layer IoT-DDoS Defense System Using Deep Reinforcement Learning. IEEE Transactions on Consumer Electronics, 69(1), 49-64, https://doi.org/10.1109/TCE.2022.3213872

[6] Nazir, A., He, J., Zhu, N., Qureshi, S. S., Qureshi, S. U., Ullah, F., et al. (2024). A deep learning-based novel hybrid CNN-LSTM architecture for efficient detection of threats in the IoT ecosystem. Ain Shams Engineering Journal, 15(7), 102777, https://doi.org/10.1016/j.asej.2024.102777

[7] Mohamed, S., & Ejbali, R. (2023). Deep SARSA-based reinforcement learning approach for anomaly network intrusion detection system. International Journal of Information Security, 22(1), 235-247, https://doi.org/10.1007/s10207-022-00634-2

[8] Santos, R. R. D., Viegas, E. K., Santin, A. O., & Cogo, V. V. (2023). Reinforcement Learning for Intrusion Detection: More Model Longness and Fewer Updates. IEEE Transactions on Network and Service Management, 20(2), 2040-2055, https://doi.org/10.1109/TNSM.2022.3207094

[9] Al-Omar, B., & Trabelsi, Z. Intrusion Detection Using Attention-Based CNN-LSTM Model. In I. Maglogiannis, L. Iliadis, J. MacIntyre, & M. Dominguez (Eds.), Artificial Intelligence Applications and Innovations, Cham, 2023 (pp. 515-526): Springer Nature Switzerland. doi:10.1007/978-3-031-34111-3_43.

[10] Benaddi, H., Ibrahimi, K., Benslimane, A., Jouhari, M., & Qadir, J. (2022). Robust Enhancement of Intrusion Detection Systems Using Deep Reinforcement Learning and Stochastic Game. IEEE Transactions on Vehicular Technology, 71(10), 11089-11102, https://doi.org/10.1109/TVT.2022.3186834

[11] Dunmore, A., Jang-Jaccard, J., Sabrina, F., & Kwak, J. (2023). A Comprehensive Survey of Generative Adversarial Networks (GANs) in Cybersecurity Intrusion Detection. IEEE Access, 11, 76071-76094, https://doi.org/10.1109/ACCESS.2023.3296707

[12] Nguyen, T. T., & Reddi, V. J. (2023). Deep Reinforcement Learning for Cyber Security. IEEE Transactions on Neural Networks and Learning Systems, 34(8), 3779-3795, https://doi.org/10.1109/TNNLS.2021.3121870

[13] Hammar, K., & Stadler, R. (2024). Learning Near-Optimal Intrusion Responses Against Dynamic Attackers. IEEE Transactions on Network and Service Management, 21(1), 1158-1177, https://doi.org/10.1109/TNSM.2023.3293413

[14] Zhao, W., Mahmoud, Q. H., & Alwidian, S. (2023). Evaluation of GAN-Based Model for Adversarial Training. Sensors, 23(5), 2697, https://doi.org/10.3390/s23052697

[15] Nakıp, M., & Gelenbe, E. (2024). Online Self-Supervised Deep Learning for Intrusion Detection Systems. IEEE Transactions on Information Forensics and Security, 19, 5668-5683, https://doi.org/10.1109/TIFS.2024.3402148

[16] Kandhro, I. A., Alanazi, S. M., Ali, F., Kehar, A., Fatima, K., Uddin, M., et al. (2023). Detection of Real-Time Malicious Intrusions and Attacks in IoT Empowered Cybersecurity Infrastructures. IEEE Access, 11, 9136-9148, https://doi.org/10.1109/ACCESS.2023.3238664

[17] Chemmakha, M., Habibi, O., & Lazaar, M. (2024). Towards a Deep Learning Approach for IoT Attack Detection Based on a New Generative Adversarial Network Architecture and Gated Recurrent Unit. Journal of Network and Systems Management, 32(4), 96, https://doi.org/10.1007/s10922-024-09873-1

[18] Randhawa, R. H., Aslam, N., Alauthman, M., Khalid, M., & Rafiq, H. (2024). Deep reinforcement learning based Evasion Generative Adversarial Network for botnet detection. Future Gener. Comput. Syst., 150(C), 294–302, https://doi.org/10.1016/j.future.2023.09.011

[19] Kanimozhi, R., & Ramesh, P. S. (2025). Deep reinforcement learning-based intrusion detection scheme for software-defined networking. Sci Rep, 15(1), 38827, https://doi.org/10.1038/s41598-025-24869-w

[20] Ma, Y., Li, C., Wang, Y., & Wang, Y. (2025). Application of deep reinforcement learning algorithms for automatic threat detection and response in dynamic network environments to improve cybersecurity. Journal of Computational Methods in Sciences and Engineering, 25(3), 2112-2125, https://doi.org/10.1177/14727978241309550

[21] Shaikh, J. A., Wang, C., Sima, M. W. U., Arshad, M., Owais, M., Hassan, D. S. M., et al. (2025). A deep Reinforcement learning-based robust Intrusion Detection System for securing IoMT Healthcare Networks. [Original Research]. Frontiers in Medicine, Volume 12 - 2025, https://doi.org/10.3389/fmed.2025.1524286

[22] Jue, C., Hongyu, T., Meng, C., Haidong, P., & Xihe, Q. (2026). GCB-PPO2: A Hybrid Deep Reinforcement Learning Intrusion Detection System for Under-Represented Attack Categories in SDN. IEEE Transactions on Network Science and Engineering, 13, 84-101, https://doi.org/10.1109/TNSE.2025.3581689