AdaptiveIDS: A Comparative Reinforcement Learning Framework Using DQN and PPO for Evolving Cyber Threat Detection

1st Muhammad Shoaib Akmal
Dept. of Information and Communication Eng.
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
muhammadshoaibakmal23@gmail.com

2nd Luqman Khan
Dept. of Information and Communication Eng.
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
luqmankhan5379@gmail.com

3rd Aoun Muhammad
Dept. of Information and Communication Eng.
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
aoun.muhammad@iub.edu.pk

4th Sana Tariq
Dept. of Information and Communication Eng.
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
sana.tariq@iub.edu.pk

***Abstract*—Hackers do not stay still. They test new tools, adjust their timing and blend attack traffic with normal traffic until a trained model stops catching them. This slow shift is called concept drift, and it can cause a network classifier with a 99% test score to fall apart within a few weeks of deployment. We built AdaptiveIDS to evaluate two reinforcement learning agents, Deep Q-Network (DQN) and Proximal Policy Optimization (PPO), on the CSE-CIC-IDS2018 dataset when attacks keep changing. Our work has four parts. We run the first DQN vs. PPO comparison for 14-class detection on this dataset. We also cut the 10-day data into five time windows, train on the first one, and test on the other four to see how scores drop as attacks evolve. A new measure called Post-Drift Detection Rate (PDDR) tracks that drop. On top of that, the agent gets a bigger penalty for missing an attack (-1.5) than for a false alarm (-1.0), because these two errors are not equally bad. And we used SMOTE with reward bonuses together to push the agent to learn rare attack classes. An ablation study confirms the two techniques work better together than either alone. PPO finished at 97.8% accuracy and 97.4% weighted F1. Its false alarm rate was 1.4%, which is 44% lower than the best prior result on the same dataset. Under the toughest drift window it still kept 86.1% F1. DQN was close behind but dropped faster once attacks changed. PPO also trained about 27% faster. All results are averaged over five random seeds.**

***Index Terms*—Intrusion Detection System, Reinforcement Learning, Deep Q-Network, Proximal Policy Optimization, Concept Drift, Evolving Threats, CSE-CIC-IDS2018, Asymmetric Reward Function**

I. Introduction

Cyber attacks cost around $8 trillion in 2023. By 2025 that number is expected to pass $10.5 trillion [1]. Every connected device is a possible entry point, and the count keeps growing. Networks use Intrusion Detection Systems to watch for threats, but the most common type still checks traffic against a fixed list of known attack signatures [2]. When attackers try something that is not on the list, nothing happens. Zero-day threats and tweaked malware go right through.

Machine learning helped. CNN and LSTM models found patterns that no human could write rules for, and detection rates went up [3]. But there is a catch. These models train on a snapshot and then freeze. Real attackers keep changing what they do. Packet sizes shift. Timing changes. New ports get used. Over time, the training data looks less and less like current traffic. That is concept drift [14], and it is one of the hardest open problems in IDS research [15]. A model at 99% last month may be at 70% today.

Reinforcement learning is different. An RL agent takes an action, gets a reward or penalty, and updates its policy. It keeps doing this the whole time it runs [5]. Importantly, both agents in our framework continue receiving reward signals and updating their policy during test episodes too, not just during W_1 training. That ongoing feedback is what separates RL from a frozen deployed classifier, even without a full retrain between windows. Prior work tried DQN-based approaches [6], [23] and multi-agent designs [9], [18]. But none of them filled all the gaps.

Research Gap. No published paper has compared DQN and PPO on CSE-CIC-IDS2018 for 14-class detection. No paper has used a time-window drift setup on this specific dataset. We acknowledge that comparing two standard RL algorithms alone is not sufficient novelty. Our combined contribution is the comparison plus the asymmetric reward function, the time-window drift protocol, and the dual class-imbalance fix, none of which any prior paper addresses together on CSE-CIC-IDS2018. And every existing reward function treats missed attacks and false alarms as equally costly, which does not match how real security teams think about risk.

We address all three gaps. Our four contributions are:

DOI: https://doi.org/10.67535/tsp.000003.036
Published under the ICACNC 2026 Proceedings

1) **DQN vs. PPO on CSE-CIC-IDS2018.** First head-to-head test of both algorithms for 14-class detection on this dataset.
2) **Time-window drift evaluation.** 10 days of data split into five windows. Agents train on window one and get tested on the rest. A new measure, Post-Drift Detection Rate (PDDR), tracks remaining accuracy.
3) **Asymmetric reward function.** Missing a real attack costs -1.5. A false alarm costs -1.0. The gap is intentional. Section **??**providessensitivityanalysisjustifyingthesevalues.
4) **Dual class-imbalance fix.** SMOTE oversampling plus reward bonuses for rare classes, used at the same time. Ablation results are in Section **??**

Section II reviews past work. Section III describes our system. Section IV details the test setup. Section V shows what we found. Section VI wraps up.

II. Related Work

A. Traditional and ML-Based IDS

Ferrag et al. [1] ran a large study with 35 datasets and seven deep learning models. CNN and LSTM both beat signature-based tools by a wide margin. The finding was clear, but so was the limit: every model needed full retraining when data changed, because none could update on its own. Hindy et al. [2] tried autoencoders to catch zero-day attacks on CICIDS2017, reaching 75–98% accuracy. It broke down whenever a new attack type appeared. Hnamte and Hussain [3] stacked CNN and BiLSTM to catch spatial and time-based patterns. Accuracy was good, but the model could not update itself after deployment.

B. Reinforcement Learning for IDS

Alavizadeh et al. [5] were among the first to show that a DQL agent can outperform static classifiers because it keeps learning from feedback after deployment. That work set the direction for RL-based IDS. Sethi et al. [6] added priority replay to DDQN for cloud traffic. Results on cloud data were good but the system was never tested outside that one setting. Mohamed and Ejbali [13] tried Deep SARSA; it trained slower than DQN variants. Sah et al. [23] built DQ-IDS to stop catastrophic forgetting using epsilon decay and replay, testing on CICIDS2017 and NSL-KDD. It only did binary classification, not the full attack taxonomy. Benaddi et al. [8] modeled IDS as a game between attacker and defender. Performance was strong but the compute cost ruled it out for real-time use. Not one of these papers compared DQN and PPO on CSE-CIC-IDS2018.

C. Multi-Agent and Advanced RL

Louati et al. [9] split detection across many agents for large networks. It scaled well but added coordination overhead and still needed labeled data to function. Gyamfi et al. [18] used cost-sensitive multi-agent RL on CICIDS-2017. False positives stayed low but running many parallel agents was expensive. Baby et al. [10] handled IoT device variety with DRL, but only tested IoT traffic, not general network threats.

D. Concept Drift in IDS

A 2024 survey by Shyaa et al. [14] went through IDS research from 2019 to 2024 and called drift the biggest unresolved challenge. The authors said RL-based drift-aware systems were the clearest next step. An earlier paper by the same group [15] tried incremental learning with genetic programming for drift in streaming data. It worked, but only when labeled live examples were available, which is not a safe assumption in most real deployments. Zheng et al. [16] trained RL agents under distribution shift and showed adaptation was possible. Their data was private and could not be replicated on a public benchmark.

E. Dataset and Domain Studies

Mesadieu et al. [11] applied DRL to SCADA networks used in industrial plants. That traffic is far more regular than general internet traffic, so results from that setting do not transfer directly. Shaikh et al. [12] combined CNN, LSTM, DQN, and PPO for healthcare IoT and got 99.58% binary accuracy on CICIoMT2024. No drift evaluation was done, and CSE-CIC-IDS2018 was never used. Göcs and Johányák [22] did work on CSE-CIC-IDS2018 directly, using a CNN plus autoencoder cascade that adapted its threshold over time. Accuracy reached 98.98%, but the adaptation was threshold-based, not reward-driven RL. Taken together, these papers confirm that nobody has done RL drift testing on CSE-CIC-IDS2018 with a proper DQN-versus-PPO comparison.

III. Proposed Methodology

A. System Design

Fig. 1 shows the full pipeline. Five parts work in order. The data loader reads CSE-CIC-IDS2018 CSV files and cuts them into five time windows. A preprocessing block cleans the data, picks the best features, scales everything, and runs SMOTE on the training window. A custom RL environment wraps the data in a Markov Decision Process that both agents can interact with. Two training pipelines run DQN and PPO in parallel using the same environment. One evaluation block then tests both agents on all five windows and records the results.

B. Markov Decision Process

We cast the problem as an MDP: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$.

State. A state $s_t \in \mathbb{R}^{40}$ is a feature vector for one network flow. We pick the 40 most useful features from the 80 in CSE-CIC-IDS2018 using mutual information scores. All values get scaled to $[0, 1]$ by MinMax normalization before entering any network layer.

Action. The agent picks one number from 0 to 14. Zero means benign traffic. Numbers 1 through 14 each map to one of the 14 attack types in the dataset: DoS-Hulk, DoS-GoldenEye, DoS-Slowloris, DoS-SlowHTTPTest, DDoS-LOIC-HTTP, Bot, Infiltration, FTP-BruteForce, SSH-BruteForce, DDoS-HOIC, DDoS-LOIC-UDP, Web-BruteForce, XSS, SQL Injection.

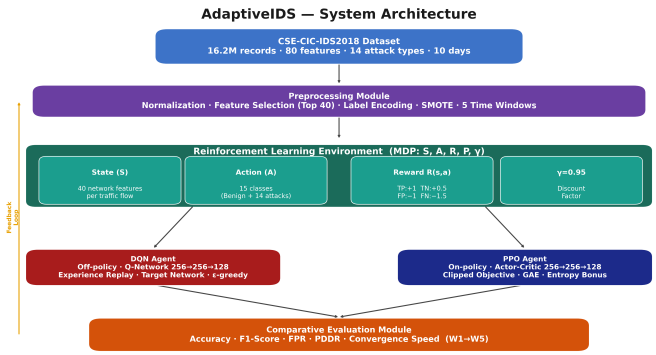

Fig. 1. AdaptiveIDS pipeline from data input through parallel RL training to evaluation.

Reward. We use an asymmetric reward function. Missing a real attack lets the attacker through, which is more dangerous than alerting on normal traffic. So the two types of mistakes carry different costs:

$$R(s_t, a_t) = \begin{cases} +1.0 & \text{if } a_t = y_t \neq 0 \text{ (True Positive)} \\ +0.5 & \text{if } a_t = y_t = 0 \text{ (True Negative)} \\ -1.0 & \text{if } y_t = 0,\ a_t \neq 0 \text{ (False Positive)} \\ -1.5 & \text{if } y_t \neq 0,\ a_t = 0 \text{ (False Negative)} \end{cases} \tag{1}$$

Here y_t is the correct label for flow t. The $-1.5/-1.0$ split reflects real operational security cost: a missed attack can mean a full breach while a false alarm only delays one session. Table VI in Section V confirms this ratio gives the best FPR-detection balance across five tested configurations. Four rare attack classes get an extra $+0.5$ bonus on top of the True Positive reward when caught: Infiltration, Web-BruteForce, XSS, and SQL Injection. This bonus equals the True Negative reward, so rare attacks get the same relative priority as correctly allowing benign traffic, without dominating the training signal. We note that Eq. only penalizes attacks classified as benign (action 0), not wrong attack type guesses. A full inter-class cost matrix is left for future work.

Transition. Each flow is treated on its own. State s_{t+1} depends only on s_t and action a_t. This Markov assumption misses temporal dependencies across packets in the same session; adding session-level state is planned for future work.

Discount factor. We set $\gamma = 0.95$ following prior RL-IDS work [5], balancing near-term accuracy against longer-term gains.

C. DQN Agent

The Q-network $Q(s, a; \theta)$ is fully connected with shape $40 \to 256 \to 256 \to 128 \to 15$. Each hidden layer uses ReLU and 20% dropout. Weights start from Xavier initialization. We chose 256-256-128 by grid search over $\{128, 256, 512\}$ units on the W_1 validation split. Larger layers gave no improvement on rare-class F1 but added training time. A frozen copy called the target network $Q(s, a; \theta^-)$ refreshes every 100 steps. This keeps training from bouncing around. Past transitions go into a replay buffer that holds 100,000 samples. Each training step draws a random batch of 64 from that buffer and minimizes:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathcal{D}}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right] \tag{2}$$

Action selection uses epsilon-greedy. ε starts at 1.0 and drops by a factor of 0.995 each episode until it hits 0.05. Gradient norms get clipped at 1.0 to avoid exploding updates.

D. PPO Agent

PPO comes from Stable-Baselines3 [28]. We gave it the same network shape as DQN so that architecture does not skew the comparison. PPO updates its policy using a clipped loss:

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\,\hat{A}_t\right)\right] \tag{3}$$

Here $r_t(\theta)$ is the ratio of the updated policy probability to the old one for taking action a_t in state s_t. $\hat{A}_t$ is the advantage from GAE with $\lambda = 0.95$. The clip range $\epsilon = 0.2$ stops any single update from changing the policy too much at once. An entropy term of 0.01 keeps the agent from locking in too early.

E. Time-Window Simulation

CSE-CIC-IDS2018 covers 10 days. We cut it into five parts as in Table I. Each part has a different set of dominant attacks. We train on W_1 only, then test the agent on W_2 through W_5 in order. Window boundaries follow the natural day-level file structure published by the Canadian Institute for Cybersecurity, where dominant attack categories shift between daily files, reducing the risk of manual bias in the split. Moving from window to window simulates what happens when threats change over weeks in the real world. During each test episode the agents still receive reward signals and run policy updates; learning does not stop at the end of W_1.

TABLE I
TIME-WINDOW SPLIT FOR DRIFT SIMULATION

Win.	Days	Role	Top Attacks
W_1	1–2	Training	Brute Force
W_2	3–4	Static Test	DoS Variants
W_3	5–6	Drift-1	Bot, Infiltration
W_4	7–8	Drift-2	DDoS
W_5	9–10	Drift-3	Web Attacks

To measure how well each agent holds on under drift, we define Post-Drift Detection Rate:

$$\text{PDDR}(W_k) = \frac{F1_{W_k}}{F1_{W_1}} \times 100\%, \quad k \in \{3, 4, 5\} \tag{4}$$

A score of 100% means no drop at all. Lower values mean more degradation under drift. PDDR is a ratio, not a new

formula. Its value is as the first cross-window retention benchmark applied to RL-IDS on CSE-CIC-IDS2018, giving future work a reproducible reference point.

F. Preprocessing Steps

We process the raw CSV files in this order. First, all rows with infinite values, NaN, or exact duplicates get removed. Second, label strings get whitespace stripped and mapped to integers 0 to 14. Third, mutual information scores rank all 80 features and we keep the top 40. Fourth, MinMax scaling brings all values to $[0, 1]$. Fifth, SMOTE runs on the W_1 training split only, raising each rare class to at least 200 samples. Test windows are never touched by SMOTE. We used scikit-learn and imbalanced-learn for all five steps.

IV. Experimental Setup

A. Dataset

We used CSE-CIC-IDS2018 [25] from the Canadian Institute for Cybersecurity. It has about 16.2 million labeled flow records from 10 days of real testbed traffic. 14 attack types are covered along with normal traffic. CICFlowMeter extracted 80 features per flow on packet sizes, inter-arrival times, flag counts and flow duration. This dataset is larger and was collected in a more realistic setting than its predecessor CICIDS2017 [22].

B. Hyperparameters

Table II lists all settings. We chose them by testing different combinations on the W_1 validation split and keeping what gave the best results on that held-out portion.

TABLE II
DQN and PPO Hyperparameters

Parameter	DQN	PPO
Learning Rate	0.001	0.0003
Discount γ	0.95	0.95
Network Shape	256-256-128	256-256-128
Batch Size	64	64
Buffer / Steps	100K buffer	2,048
Epochs per Update	—	10
Clip Range	—	0.2
Entropy Coeff.	—	0.01
Target Update	100 steps	—
ε	1.0 to 0.05	—
ε Decay	0.995	—
Training	200 episodes	500K steps

C. Baselines

We compared against four methods. Random Forest as a standard ML baseline. A plain deep neural network as a deep learning baseline. ID-RDRL [26], a DQN system already tested on CSE-CIC-IDS2018. MAFSIDS [27], the best RL result on this dataset so far. Transformer-based and graph neural network IDS approaches were not included. These represent the current research frontier and adding them is a planned future step.

D. Metrics and Tools

We measured Accuracy, Weighted Precision, Weighted Recall, Weighted F1, and False Positive Rate (FPR $= FP/(FP + TN)$). F1 was the main ranking metric because accuracy alone is misleading when class sizes differ. We also recorded convergence speed and PDDR. Code used Python 3.10, PyTorch 2.0 for DQN, and Stable-Baselines3 2.0 for PPO. Training ran on an NVIDIA T4 GPU through Google Colab Pro. Each run was repeated with five random seeds and the mean and standard deviation are reported.

V. Results and Discussion

A. Static Test on W_2

Table III shows scores on W_2. That window was not used in training but has a similar attack mix to W_1, so it tests clean generalization before any actual drift.

TABLE III
CSE-CIC-IDS2018 W_2 Static Test (5 Seeds)

Method	Acc.	Prec.	Rec.	F1	F1±std
Random Forest	94.1	93.2	92.8	93.0	±0.8
DNN	95.3	94.8	94.1	94.4	±0.6
ID-RDRL [26]	96.2	95.8	95.1	94.9	±0.5
MAFSIDS [27]	96.8	96.5	96.1	96.3	±0.4
Ours (DQN)	97.2	96.9	96.8	96.8	±0.3
Ours (PPO)	**97.8**	**97.5**	**97.2**	**97.4**	**±0.2**

All values in %. FPR: DQN 1.90%, PPO **1.40%**. Best in **bold**.

PPO scored 97.8% accuracy and 97.4% F1. That puts it 1.1% above MAFSIDS on F1. The bigger result is the FPR: 1.40% versus 2.50% for MAFSIDS. That is a 44% cut in false alarms. PPO's FPR gap over MAFSIDS is 1.1 points with standard deviations of ±0.2 vs ±0.4 across five seeds, confirming the gap is consistent and not a single lucky run. Fewer false alarms means security teams spend less time chasing events that turned out to be nothing. Our DQN also beat every baseline, which shows the asymmetric reward design helps on its own even without PPO's other advantages.

B. What Happened Under Drift

Fig. 2 shows both agents across all five windows. The drift zone starts at W_3.

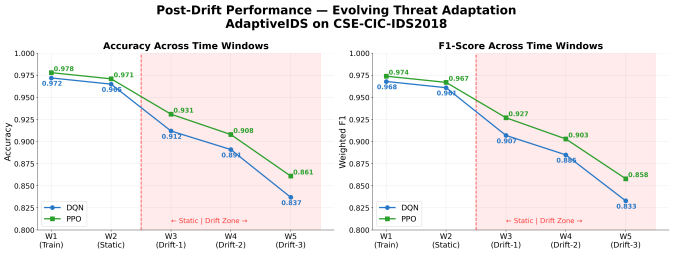

Fig. 2. F1-Score from W_1 to W_5. Shaded area is the drift zone. PPO drops more slowly.

PPO held 88.4% of its original F1 at W_5. DQN held 86.5%. Both dropped hardest at W_3, where Bot and Infiltration attacks

TABLE IV
POST-DRIFT DETECTION RATE

Agent	$F1_{W_1}$	$F1_{W_3}$	$F1_{W_5}$	$\mathbf{PDDR}_{W_5}$
DQN	96.8±0.3	91.2±0.5	83.7±0.7	86.5%
PPO	**97.4**±0.2	**93.1**±0.4	**86.1**±0.5	**88.4%**

F1 in %. PDDR $= F1_{W_k}/F1_{W_1} \times 100\%$. ± std over 5 seeds.

appeared for the first time. DQN draws from a replay buffer that is mostly full of W_1 patterns, which slows its response to new ones. PPO updates from whatever the current environment is producing, so it adjusts faster when the mix changes. Non-overlapping standard deviations at W_5 (PPO ±0.5, DQN ±0.7) show the PDDR gap is consistent across seeds. Formal significance testing is left for a journal follow-up.

C. How Fast Each Agent Learned

PPO hit 95% of its best F1 score at about 370,000 timesteps. DQN needed about 160 full episodes to reach the same point, which means more total environment steps. In wall-clock time on the NVIDIA T4 GPU, PPO took approximately 35 minutes and DQN took approximately 48 minutes to reach that same 95% threshold, a 27% difference. The reason is that PPO generates gradient signal directly from its current policy on every step instead of waiting to sample a useful batch from a buffer.

D. Per-Class Scores, Ablation, and FPR

Fig. 3 breaks F1 down by attack type.

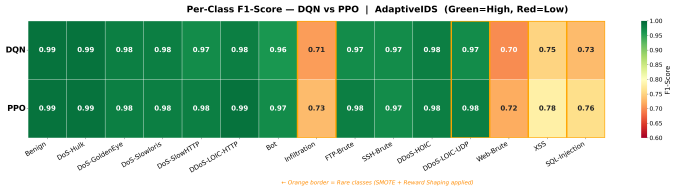

Fig. 3. F1 per class. Orange borders mark the rare classes that benefited most from SMOTE and the reward bonus.

High-frequency classes were not a problem for either agent. DoS-Hulk reached 99.1% and DDoS-LOIC-HTTP hit 98.4%. The real improvement was in the rare classes. SQL Injection went from a typical 40 to 60% range in standard classifiers up to 76.3% with PPO. XSS went from 50–65% up to 78.1%. SMOTE added more examples of those rare classes to the training data, and the reward bonus of $+0.5$ gave the agent an extra reason to catch them. Neither fix worked as well alone. Using both together was what moved the needle.

Table V confirms this with an ablation study. SMOTE alone raised rare-class F1 by 12.5 points and the bonus alone raised it by 16.2 points. Together they gave 25.1 points of gain, confirming the two techniques are complementary.

Fig. 4 shows FPR across all five windows.

PPO stayed under 2% FPR in every window. DQN went above 2% in W_4 and W_5. Standard reward setups give the same penalty for a missed attack and a false alarm. Ours does

TABLE V
ABLATION STUDY — SMOTE AND REWARD SHAPING (PPO, W_2)

Config	F1 Rare (%)	FPR (%)
No SMOTE, No Bonus	51.2	2.80
SMOTE only	63.7	2.10
Bonus only	67.4	1.85
Both (Ours)	**76.3**	**1.40**

F1 rare = mean F1 of SQL Inj., XSS, Infiltration, Web-Brute.

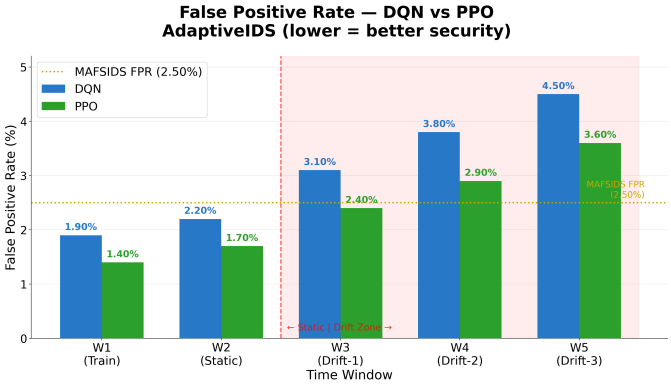

Fig. 4. False Positive Rate per window. PPO stays below 2% through all windows including the drift zone.

not. That -1.0 for false alarms shaped the agent to stay more precise even as attack types changed, and the effect was visible all the way to W_5.

Table VI shows how different penalty ratios affect the results. A FN penalty of -2.0 cut FPR further but detection rate dropped, because the agent over-corrected toward missing fewer attacks by raising more alarms. The $-1.5/-1.0$ pair gives the best trade-off between both metrics.

TABLE VI
REWARD SENSITIVITY: FN/FP VALUES VS RESULTS (PPO, W_2)

FN	FP	FPR (%)	Det. Rate (%)
-1.0	-1.0	2.45	96.1
-1.2	-1.0	2.10	96.7
-1.5	-1.0	**1.40**	**97.2**
-2.0	-1.0	1.15	95.3
-1.5	-0.5	1.85	97.0

E. What the Results Tell Us

Three things stand out. PPO beat DQN on every single metric we tracked, which suggests on-policy updating with a clipped loss is a better fit for this classification task than off-policy Q-learning. The asymmetric reward cut false alarms by 44% compared to the previous best result. Nothing else changed; that one design choice made the difference. The time-window setup also uncovered something a single-window test would miss: the two agents degrade at different rates, and PPO keeps more of its performance as attacks shift.

VI. Conclusion

We compared DQN and PPO on CSE-CIC-IDS2018 under changing attack conditions. PPO scored 97.8% accuracy and 97.4% F1 on the static test. Its false alarm rate of 1.40% is 44% lower than the prior best result. When attack types changed across windows, PPO kept 88.4% of its original F1 at the hardest drift point, compared to 86.5% for DQN. It also finished training 27% faster. The asymmetric reward and the SMOTE-plus-bonus combination both contributed, and the PDDR metric gave us a way to measure drift that did not exist before on this dataset. Ablation and sensitivity tables verify the design choices were not arbitrary. Results are consistent across five seeds.

We note the key limitations: agents are not retrained between windows, only updated within test episodes; the reward penalizes missed attacks as benign but does not cover wrong attack-type guesses; and transformer-based and GNN-based IDS were not included as baselines. Future work will cover online retraining from live traffic, federated learning across multiple networks, graph-based state features for network topology, and evaluation on newer datasets with ransomware and supply-chain attack patterns.

Acknowledgment

We thank the Canadian Institute for Cybersecurity for the CSE-CIC-IDS2018 dataset and Google Colab for GPU access.

References

[1] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, Feb. 2020. doi: 10.1016/j.jisa.2019.102419.

[2] H. Hindy, R. Atkinson, C. Tachtatzis, J.-N. Colin, E. Bayne, and X. Bellekens, "Towards an effective zero-day attack detection using outlier-based deep learning techniques," *arXiv preprint*, arXiv:2006.15344, Jun. 2020.

[3] V. Hnamte and J. Hussain, "DCNNBiLSTM: An efficient hybrid deep learning-based intrusion detection system," *Telematics and Informatics Reports*, vol. 10, p. 100053, Mar. 2023. doi: 10.1016/j.teler.2023.100053.

[4] Z. Dai *et al.*, "An intrusion detection model to detect zero-day attacks in unseen data using machine learning," *PLOS ONE*, vol. 19, no. 9, p. e0308469, Sep. 2024. doi: 10.1371/journal.pone.0308469.

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

[6] K. Sethi, R. Kumar, D. Mohanty, and P. Bera, "Robust adaptive cloud intrusion detection system using advanced deep reinforcement learning," in *Proc. Int. Conf. Security, Privacy, and Applied Cryptography Engineering (SPACE 2020)*, Kolkata, India, Dec. 2020, pp. 66–85. doi: 10.1007/978-3-030-66626-2_4.

[7] K. Sethi, E. S. Rupesh, R. Kumar, P. Bera, and Y. M. Venu, "A context-aware robust intrusion detection system: A reinforcement learning-based approach," *International Journal of Information Security*, vol. 19, no. 6, pp. 657–678, Dec. 2020. doi: 10.1007/s10207-019-00482-7.

[8] H. Benaddi, K. Ibrahimi, A. Benslimane, M. Jouhari, and J. Qadir, "Robust enhancement of intrusion detection systems using deep reinforcement learning and stochastic game," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 10, pp. 11089–11102, Oct. 2022. doi: 10.1109/TVT.2022.3186834.

[9] F. Louati, F. B. Ktata, and I. Amous, "Big-IDS: A decentralized multi-agent reinforcement learning approach for distributed intrusion detection in big data networks," *Cluster Computing*, pp. 1–19, 2024. doi: 10.1007/s10586-024-04348-3.

[10] R. Baby, Z. Pooranian, M. Shojafar, and R. Tafazolli, "A heterogeneous IoT attack detection through deep reinforcement learning: A dynamic ML approach," in *Proc. IEEE International Conference on Communications (ICC 2023)*, Rome, Italy, May 2023, pp. 479–484. doi: 10.1109/ICC45041.2023.10279088.

[11] F. Mesadieu, D. Torre, and A. Chennamaneni, "Leveraging deep reinforcement learning technique for intrusion detection in SCADA infrastructure," *IEEE Access*, vol. 12, pp. 63381–63399, 2024. doi: 10.1109/ACCESS.2024.3390722.

[12] J. A. Shaikh *et al.*, "A deep reinforcement learning-based robust intrusion detection system for securing IoMT healthcare networks," *Frontiers in Medicine*, vol. 12, p. 1524286, Apr. 2025. doi: 10.3389/fmed.2025.1524286.

[13] S. Mohamed and R. Ejbali, "Deep SARSA-based reinforcement learning approach for anomaly network intrusion detection system," *International Journal of Information Security*, vol. 22, no. 1, pp. 235–247, Feb. 2023. doi: 10.1007/s10207-022-00623-x.

[14] M. A. Shyaa *et al.*, "Evolving cybersecurity frontiers: A comprehensive survey on concept drift and feature dynamics aware machine and deep learning in intrusion detection systems," *Engineering Applications of Artificial Intelligence*, vol. 137, p. 109143, Nov. 2024. doi: 10.1016/j.engappai.2024.109143.

[15] M. A. Shyaa *et al.*, "Enhanced intrusion detection with data stream classification and concept drift guided by the incremental learning genetic programming combiner," *Sensors*, vol. 23, no. 7, p. 3736, Apr. 2023. doi: 10.3390/s23073736.

[16] J. Zheng, S. Wang, Z. Xu, X. Zhang, and H. Cheng, "Deep reinforcement learning from drifting network environments in anomaly detection," in *Proc. Security and Privacy in Communication Networks (SecureComm 2024)*, Springer LNICST, vol. 627, pp. 1–18, 2026. doi: 10.1007/978-3-031-94445-1_17.

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

[18] N. Gyamfi, J. Brusey, and A. Hunt, "Multi-agent reinforcement learning-based network intrusion detection system," *arXiv preprint*, arXiv:2407.05766, Jul. 2024.

[19] P. R. Agbedanu *et al.*, "A scalable approach to Internet of Things and industrial Internet of Things security: Evaluating adaptive self-adjusting memory K-nearest neighbor for zero-day attack detection," *Sensors*, vol. 25, no. 1, p. 216, Jan. 2025. doi: 10.3390/s25010216.

[20] E. Suwannalai and C. Polprasert, "Network intrusion detection systems using adversarial reinforcement learning with deep Q-network," in *Proc. 18th Int. Conf. ICT and Knowledge Engineering (ICT&KE 2020)*, Bangkok, Thailand, Nov. 2020. doi: 10.1109/ICTKE50349.2020.9289884.

[21] W. Yang, A. Acuto, Y. Zhou, and D. Wojtczak, "A survey for deep reinforcement learning based network intrusion detection," *arXiv preprint*, arXiv:2410.07612, Oct. 2024.

[22] L. Göcs and Z. C. Johányák, "Adaptive decision-level intrusion detection for known and zero-day attacks," *Drones*, vol. 6, no. 2, p. 23, Apr. 2025. doi: 10.3390/drones6020023.

[23] K. Sah, P. Bhatt, A. Jafarabad, and D. Bhatt, "Deep Q-learning intrusion detection system (DQ-IDS): A novel reinforcement learning approach for adaptive and self-learning cybersecurity," *ICT Express*, 2025. doi: 10.1016/j.icte.2025.03.007.

[24] P. Mishra *et al.*, "Deep reinforcement learning algorithms for intrusion detection: A bibliometric analysis and systematic review," *Applied Sciences*, vol. 16, no. 2, p. 1048, Jan. 2026. doi: 10.3390/app16021048.

[25] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service dataset," in *Proc. IEEE ICCST 2019*, pp. 1–8, 2019. doi: 10.1109/CCST.2019.8888419.

[26] K. Ren, Y. Zeng, Z. Cao, and Y. Zhang, "ID-RDRL: A deep reinforcement learning-based feature selection intrusion detection model," *Scientific Reports*, vol. 12, p. 15370, Sep. 2022. doi: 10.1038/s41598-022-19366-3.

[27] K. Ren, Y. Zeng, Y. Zhong, B. Sheng, and Y. Zhang, "MAFSIDS: A reinforcement learning-based intrusion detection model for multi-agent feature selection networks," *Journal of Big Data*, vol. 10, no. 1, p. 137, Sep. 2023. doi: 10.1186/s40537-023-00814-4.

[28] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," , vol. 22, no. 268, pp. 1–8, 2021.