Optimization Of Network Intrusion Detection Using Deep Reinforcement Learning: A Controlled Study of DQN And Double-DQN

Jayshree Ghorpade-Aher[1], Manthan Jikadara[2], Nimai Singhal[3]
[1,2,3]*DCET, SOCSE, Dr. Vishwanath Karad MITWPU, Pune, India*

***Abstract*—Network intrusion detection systems face an operational tension: detecting more attacks can also increase the number of benign network flows incorrectly escalated. Prior work has explored deep Q-networks (DQN) for intrusion classification, but the practical value of reinforcement learning should be judged against strong supervised models and explicitly cost-sensitive decision rules. This paper studies a two-stage framework in which a frozen binary LightGBM classifier provides traffic probabilities to a DQN or Double-DQN decision policy. A simulated security operations center (SOC) contributes queue and alert-budget state, allowing actions to influence subsequent operational conditions. CIC-IDS2017 traffic is relabeled as normal or attack and partitioned to separate supervised fitting, reinforcement learning, validation, and a reserved test set. On 434,298 common validation flows, default LightGBM yields a false-positive rate (FPR) of 0.0410% and attack recall of 99.9045%. DQN improves recall to 99.9706% but raises FPR to 0.0702%. Double-DQN reaches 99.9790% recall at 0.0814% FPR. Thus, these initial policies favor sensitivity rather than demonstrating false-positive reduction. We report exact decision counts, training traces, cost-sensitive baselines, and methodological constraints. The work provides a reproducible starting point for investigating when sequential decision policies offer advantages beyond calibrated, fixed-threshold classifiers.**

***Index Terms*—Network intrusion detection, deep reinforcement learning, DQN, Double-DQN, LightGBM, false-positive rate, CIC-IDS2017, cost-sensitive classification, SOC simulation**

I. INTRODUCTION

A network intrusion detection system (NIDS) examines network activity for signs of malicious behavior. Modern machine-learning approaches usually convert packet exchanges into statistical flows and classify each flow as normal traffic or a known attack. Benchmark performance can be impressive, yet operating a detector introduces a different question: how should a security operations center (SOC) respond when an alert may be wrong? A false negative can leave an attack unexamined. A false positive consumes analyst time and can hide genuinely urgent events within a busy queue. Consequently, accuracy alone does not capture the decision quality of an intrusion detector.

Tree ensembles are effective on structured traffic measurements. LightGBM, for example, combines histogram-based training with efficient gradient-boosted decision trees [1]. But a probability estimate is normally converted into an alert by a static threshold. Such a rule cannot directly take account of operational resources unless those considerations are encoded separately. Reinforcement learning (RL) offers one possible way to represent this decision process: an agent observes the flow and resource state, takes an action, and receives a reward reflecting detection outcomes and workload. DQN learns action values with a neural network [2], while Double-DQN separates greedy action selection from target-value evaluation to reduce certain overestimation effects [3]. Tang *et al.* [4] recently used DQN for a binary normal/intrusion decision and reported improved accuracy, recall, F1 score and FPR over a traditional baseline. Their experiment employed KDD Cup 99 and is not directly comparable with CIC-IDS2017. More broadly, a survey of DRL-based NIDS research identifies unresolved issues surrounding computational efficiency, minority attacks and credible deployment evaluation [5].

These studies motivate a narrower and testable question: *Does a cost-sensitive deep-RL layer improve*

network-alert decisions once a strong supervised probability model and tuned thresholds have already been considered?

We extend an existing CIC-IDS2017 LightGBM pipeline into a binary decision study. The frozen classifier supplies probabilities and selected normalized traffic measurements to a DQN or Double-DQN. A simulated SOC supplies queue and budget information. The agent chooses whether to accept the flow as normal or raise an attack alert.

The present work makes three contributions:

1. A two-stage supervised/DRL experimental formulation with separated fitting and policy-training data, explicit reward terms, and an action-dependent simulated workload state.
2. A controlled comparison against default, tuned, and cost-sensitive LightGBM decision rules on the same 434,298 validation observations.
3. An evidence-based analysis of recall–FPR trade-offs, policy learning traces, validation-selected operating points, and limitations that constrain claims of real-world generalization.

We intentionally do not claim that the reinforcement-learning models outperform the supervised baseline: the measured results do not support that conclusion.

II. RELATED WORK AND RESEARCH MOTIVATION

A. Flow-Based Intrusion Detection

Sharafaldin *et al.* introduced CIC-IDS2017 to provide more varied network behavior and attack scenarios than older intrusion benchmarks [6]. Its flow-level measurements include counts, packet-length statistics, flags, rates and inter-arrival times. Although these features are useful for supervised learning, the dataset is strongly imbalanced across individual attack categories, and handling invalid values and repeated observations matters for reliable evaluation. The binary task in this paper retains all malicious categories as one positive class. The original category labels are kept outside the policy input for diagnostic analysis.

LightGBM is a useful benchmark for this type of numerical data [1]. SHAP offers an explanation framework for model predictions [7]. However, the current DRL experiment uses an existing set of traffic features and uncertainty signals, not signed SHAP values as policy inputs. The earlier multiclass SHAP analysis therefore motivates future experiments but is not itself evidence that explainability improves the measured RL decisions.

B. Deep Reinforcement Learning for NIDS

The foundational DQN algorithm combines replay memory and a delayed target network to stabilize value learning [2]. Double-DQN modifies the temporal-difference target to reduce maximization bias [3]. Tang *et al.* describe traffic attributes as a DQN state and normal/intrusion classification as its two possible actions [4]. Their reported comparison is 84.3% versus 92.1% accuracy, 76.5% versus 87.8% recall, 80.2% versus 89.1% F1, and 15.1% versus 7.9% FPR for their conventional and DRL systems, respectively. Those numbers belong to that study and are not presented here as results on our dataset.

Simply treating independent labeled rows as successive “states” can make a nominal RL experiment difficult to distinguish from ordinary cost-sensitive classification. In response, our simulator includes an action-dependent SOC queue and budget. This does not make the experiment a verified temporal replay of real traffic, but it gives the decisions a measurable sequential operational consequence. A key comparison remains whether that additional complexity beats a much simpler cost-sensitive LightGBM threshold.

III. DATASET AND EXPERIMENTAL DESIGN

A. Binary Labels and Cleaning

We use the CIC-IDS2017 parquet collection containing 2,830,743 flow records. BENIGN is encoded as 0 (normal). Every other original label is encoded as 1 (attack). The initial source contains 78 candidate numerical feature columns and a target label. A conflict audit quarantined 7,020 observations across 698 feature-identical groups carrying inconsistent binary labels (736 normal and 6,284 attack), leaving 2,823,723 observations. This operation makes the training target unambiguous for those groups but may introduce selection bias. It is therefore treated as a methodological choice rather than an objective improvement to the benchmark.

The underlying source also contains many exact repeats: 308,381 original duplicate rows were recorded in the preceding audit. Instead of regarding

repeated flows as independent evidence, partition construction prevented group overlap. Exact duplicates could remain *within* a partition. The feature preprocessing was fitted only on the supervised-training partition and uses median-based treatment of missing or non-finite values and scale normalization based on its training distribution.

Eight constant columns were among the removed predictors, yielding 70 supervised model features. The resulting model, transformation parameters and feature ordering remain frozen during policy training.

TABLE I. Disjoint experimental partitions (binary CIC-IDS2017)

Partition	Normal	Attack	Total
Supervised fit	794,742	184,938	979,680
Supervised validation	169,145	38,280	207,425
Calibration	172,872	39,006	211,878
RL training	453,620	116,810	570,430
RL validation	338,977	95,321	434,298
Reserved final test	343,005	77,007	420,012
Total	2,272,361	551,362	2,823,723

Counts include duplicates retained within each partition. The final-test observations have not been evaluated in the available experiment record.

B. Partition Protocol and Frozen Supervised Baseline

Table I describes six non-overlapping partitions. Supervised fitting trains LightGBM. Supervised validation selects the boosting iteration and candidate decision rules. A separate partition supports candidate probability calibration. RL training fits the agent. RL validation selects checkpoints and operating offsets. The final-test split is reserved and excluded from every result reported here.

The initial model used 485 selected boosting iterations, a learning rate of 0.05, 31 leaves, and a minimum of 50 training observations per leaf. Candidate sigmoid calibration was evaluated but not adopted, so the model's probability outputs should not be described as demonstrably calibrated. Fig. 1 documents the supervised stage before RL: both training and validation binary log loss decrease rapidly and then flatten, and the selected boosting iteration is marked at 485.

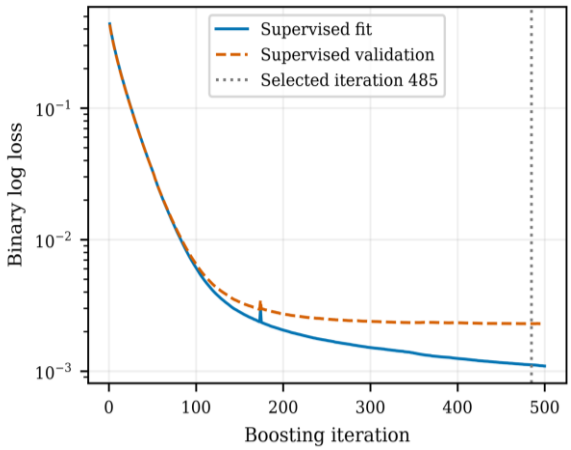


Fig. 1. Supervised LightGBM training and validation binary log loss. The selected iteration at 485 provides the frozen probability model used by the later DQN and Double-DQN stages.

In addition to LightGBM with a 0.5 threshold, we evaluate a validation-tuned recall-oriented threshold of approximately 0.9678 and a cost-sensitive threshold of $1/(1+20) \approx 0.0476$.

The latter reflects an illustrative 20:1 missed-attack to false-alarm penalty, not a measured economic cost. Comparing these rules against RL is essential: a policy cannot be credited with improved decision-making if the gain is equally available by moving a static threshold.

IV. PROPOSED METHODOLOGY

A. Hybrid Perception and Decision Layers

Fig. 2 shows the implemented two-stage design. Frozen LightGBM receives the 70 aligned model features and produces an attack score. A 16-dimensional agent observation contains one score, two uncertainty-related signals, eight standardized traffic features and five operational-state variables representing queue, budget and episode phase/horizon.

The action set in the completed experiment is

$$\mathcal{A} = \{0: \text{Normal}, \quad 1: \text{Alert}\}. \quad (1)$$

Neither the ground-truth label nor later flow records enter the policy state. A separate simulator uses labels only to calculate training reward and evaluation metrics. SHAP-state augmentation and an abstain/review action are future variants, not measured results in this paper.

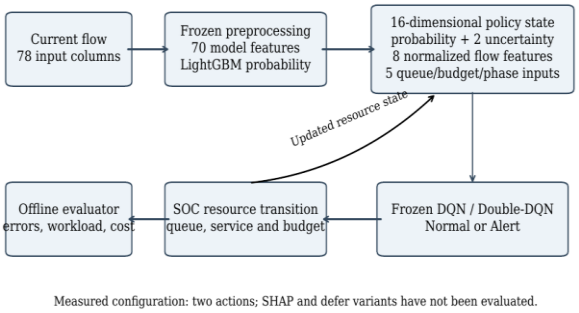


Fig. 2. Implemented hybrid LightGBM–DRL pipeline. SHAP-input and defer-action extensions were not trained in the recorded experiment.

B. Simulated Sequential SOC Environment

The environment processes flow in a fixed, simulated shuffle. Each episode spans up to 512 observations. An alert requests work from a queue with capacity 32. A budget of 48 work units is renewed every 128 flows, and one unit of queue work is served every four flows. Admission happens before service. These operating constraints create a transition in the resource state when an alert is emitted. Episodes reset resource counters. Terminal queue contents are recorded rather than cleared for free. The state is an operational approximation rather than a formally proven sufficient Markov representation.

The sequence is *not* a chronological packet capture. The original feature table lacks verified timestamps and endpoint/flow identifiers, so neither source-level attack progression nor time-to-detect performance can be inferred. Fig. 3 displays the simulated interaction.

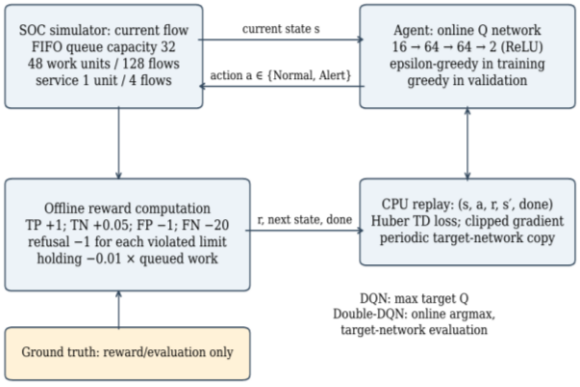


Fig. 3. Offline SOC decision loop. The selected action affects workload, while labels are available only to reward computation and post-hoc evaluation.

C. Cost-Sensitive Reward and Evaluation Cost

Let $y_t \in \{0,1\}$ be the true class and $a_t \in \{0,1\}$ the effective decision. The classification component of reward uses the values implemented in our reference environment:

$$r_{cls}(y_t, a_t) = \begin{cases} +0.05, & (y_t, a_t) = (0,0), \\ +1, & (y_t, a_t) = (1,1), \\ -1, & (y_t, a_t) = (0,1), \\ -20, & (y_t, a_t) = (1,0). \end{cases} \quad (2)$$

The environment additionally charges 0.01 per queued work unit per step and 1 for queue or budget violations. A deferred-review penalty of 2 is configured for a later unmeasured extension. Accordingly, the effective step reward can be expressed as

$$r_t = r_{cls}(y_t, a_t) - 0.01 q_t - \mathbb{I}_{overflow} - \mathbb{I}_{budget}. \quad (3)$$

All compared policies are replayed through the same resource model. A common simulated cost incorporates false alarms, missed attacks and queue-related charges. It is an accounting mechanism used for comparison, not verified SOC labor expenditure. Crucially, phase-one costs and phase-four common costs use different accounting definitions and are not mixed.

D. DQN and Double-DQN Updates

An online network $Q_\theta(s, a)$ estimates the discounted return for each decision. A periodically synchronized target network $Q_{\bar{\theta}}$ supports bootstrapping. For a replay transition $(s_t, a_t, r_t, s_{t+1}, d_t)$, the vanilla DQN target is

$$y_t^{DQN} = r_t + \gamma(1 - d_t) \max_{a'} Q_{\bar{\theta}}(s_{t+1}, a'). \quad (4)$$

Double-DQN uses the online network to choose the next greedy action and the target network to evaluate it [3]:

$$y_t^{DDQN} = r_t + \gamma(1 - d_t) Q_{\bar{\theta}} \left(s_{t+1}, \operatorname{argmax}_{a'} Q_\theta(s_{t+1}, a') \right). \quad (5)$$

Both agents optimize a Huber temporal-difference loss over minibatches drawn from replay memory. A linearly annealed ϵ-greedy exploration schedule selects random actions early in training and favors the learned greedy decision later. The trained action value is *not* itself a calibrated intrusion probability. The probability comes from frozen LightGBM.

E. Training Protocol and Checkpointing

Both agents use a 16–64–64–2 multilayer perceptron with ReLU activations, Adam learning rate 10^{-3}, discount factor $\gamma = 0.99$, replay capacity 100,000 and minibatch size 128. Optimization begins after 2,048

transitions, occurs once every four environment steps, and synchronizes the target after every 1,000 optimizer updates. Exploration falls from 1.0 to 0.05 over 200,000 steps. Gradient norm is clipped at 10. The training budget is 500 episodes (256,000 environment steps), which is less than one full pass over the 570,430-row RL-training partition. Checkpoints were evaluated on the fixed validation episodes. DQN was selected at episode 350 and Double-DQN at episode 400. Only random seed 42 has completed research-scale policy training.

V. EVALUATION CRITERIA AND EVIDENCE PROVENANCE

For a binary decision, a true positive (TP) is an alerted attack and a false positive (FP) is benign traffic incorrectly alerted. The primary outcomes are

$$\text{FPR} = \frac{FP}{FP+TN}, \qquad \text{Recall} = \frac{TP}{TP+FN}, \quad (6)$$

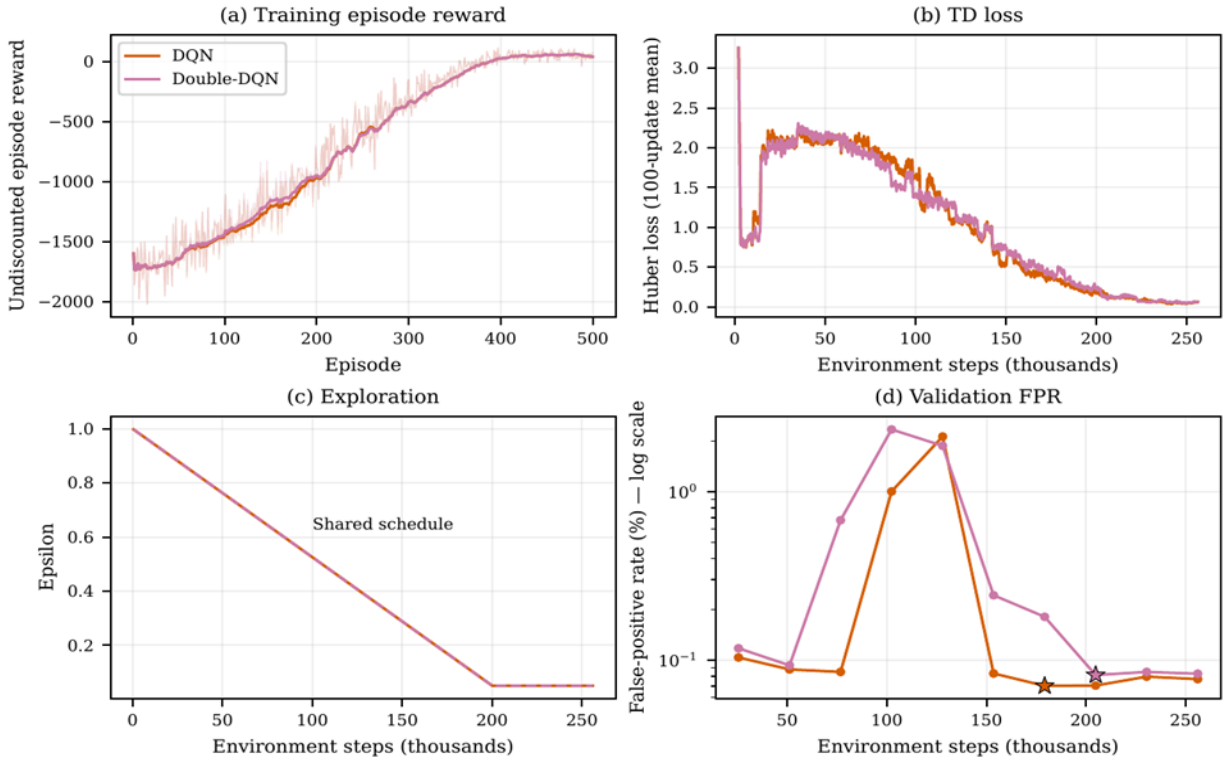
with precision $TP/(TP+FP)$ and attack-class F_1 as supporting measures. Each operational rule is evaluated on exactly 434,298 RL-validation flows (338,977 normal, 95,321 attack) in the same simulated order and with identical resource settings. We report flow counts as well as percentages because small absolute changes in benign alerts matter at scale.

We distinguish *original* trained decisions from *adjusted* decisions obtained after a validation-selected additive offset. The preregistered matched-recall region was 99.00–99.05%, but no measured operating point fell inside it. The two RL offset searches ended at their configured upper boundary. Therefore, the adjusted results cannot be interpreted as FPR performance at genuinely matched recall. No independent final-test evaluation or multiple-seed uncertainty estimate is available in this manuscript.

VI. RESULTS AND DISCUSSION

A. Learning Behavior

Fig. 4 combines episode reward, Huber TD loss, exploration schedule, and checkpoint-validation outcomes. Reward generally improves during training as ϵ decreases. Loss changes as the agent's policy and replay distribution evolve. These observations establish that both implementations learned nonconstant decision policies within the allocated budget. They do not, by themselves, prove convergence to an optimal policy or demonstrate better detection than LightGBM. The validation traces show substantial fluctuations at intermediate checkpoints, reinforcing the importance of retaining checkpoint provenance.



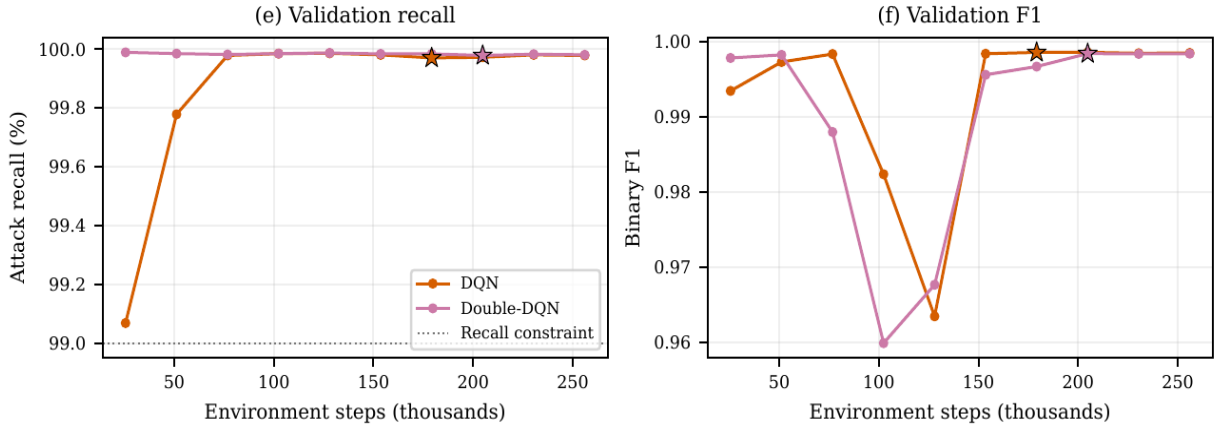


Fig. 4. DQN and Double-DQN learning curves (seed 42). Reward and TD-loss lines include smoothing. Validation results cover 849 fixed episodes. Stars mark selected checkpoints (DQN 350, Double-DQN 400). One seed does not establish training stability across independent runs.

B. Detection Decisions on a Common Validation Set

Table II reports the five original operating rules. Default LightGBM produces 139 false positives and 91 false negatives, corresponding to 0.0410% FPR and 99.9045% attack recall. DQN yields fewer missed attacks (28) and higher recall (99.9706%), but produces 238 false positives, raising FPR to 0.0702%. Double-DQN misses only 20 attacks (99.9790% recall), while the 276 benign alerts give 0.0814% FPR. Both RL policies therefore move toward greater sensitivity at a measurable false-alert cost. Neither improves FPR over the default classifier.

The tuned LightGBM threshold reduces false positives to 79 (0.0233% FPR), but misses 756 attacks, decreasing recall to 99.2069%. The cost-sensitive threshold shows the converse pattern: only seven missed attacks (99.9927% recall), alongside 354 false positives (0.1044% FPR). This broad range from simple thresholds illustrates why a single accuracy figure is not enough and why a reinforcement-learning policy must be compared at an appropriate operating point rather than only at its native greedy decision.

TABLE II. Original decisions on the same 434,298 RL-validation flows (one RL seed)

Method	TP	TN	FP	FN	FPR (%)	Attack recall (%)	Attack F1 (%)
LightGBM (0.5 threshold)	95,230	338,838	139	91	0.0410	99.9045	99.8794
LightGBM (tuned threshold)	94,565	338,898	79	756	0.0233	99.2069	99.5604
LightGBM (cost-sensitive rule)	95,314	338,623	354	7	0.1044	99.9927	99.8110
DQN (seed 42)	95,293	338,739	238	28	0.0702	99.9706	99.8606
Double-DQN (seed 42)	95,301	338,701	276	20	0.0814	99.9790	99.8449

Positive class is Attack. All counts come from the same replay. Lower FPR and higher recall are competing objectives. Bold entries are individual-column extrema, not a claim of overall superiority.

The FPR–recall view in Fig. 5 makes the outcome clearer than accuracy: the cost-sensitive baseline and the learned policies occupy different portions of the detection trade-off. In a high-consequence setting an additional false positive might be acceptable in exchange for substantially fewer misses, but that conclusion depends on an independently justified cost model. Here, the cost ratio is experimental, not derived from a SOC deployment.

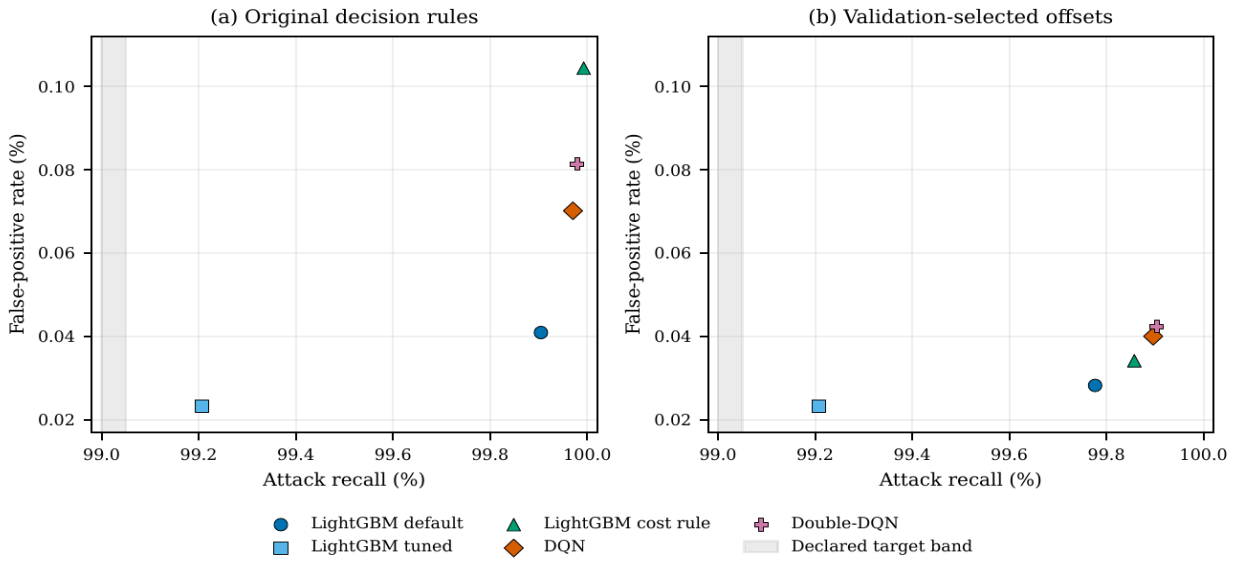


Fig. 5. False-positive rate versus attack recall for original and validation-adjusted operating points. No point satisfies the declared 99.00–99.05% matching band. Points are not interpolated, and the offsets were selected on validation data

C. Validation-Selected Offsets and Operational Cost

The offset-adjusted DQN produces 136 FP and 99 FN, compared with 139 FP and 91 FN for default LightGBM. It saves three benign alerts but misses eight additional attacks. Adjusted Double-DQN produces 144 FP and 92 FN. Neither result demonstrates an overall improvement at the declared target recall. Moreover, both RL offsets reached the search-grid maximum, indicating that the specified search range is insufficient to describe the trade-off fully.

Under the common Phase-4 simulator accounting, the original LightGBM default incurs 11,825.69 cost units, versus 10,754.77 for DQN and 10,659.59 for Double-DQN. However, the simple *cost-sensitive LightGBM rule* reaches 10,527.73, lower than either learned policy. These values are simulated and conditional on the common queue model. They show that the objective can favor attack-sensitive policies while still failing to establish a unique advantage from RL.

TABLE III. Adjusted validation decisions and common simulator cost

Method	FP	FN	FPR (%)	Cost units
LightGBM default	139	91	0.0410	11,825.69
LightGBM cost rule	354	7	0.1044	10,527.73
DQN, original	238	28	0.0702	10,754.77
Double-DQN, original	276	20	0.0814	10,659.59
DQN, adjusted	136	99	0.0401	11,974.48
Double-DQN, adjusted	144	92	0.0425	11,850.83

Adjusted offsets were chosen on the same validation partition. Costs include common workload accounting and are not monetary estimates.

D. Confusion-Matrix and Category Analysis

Fig. 6 presents absolute and row-normalized confusion matrices for all original rules. Large normal-class support makes FPR numerically small even where the count of benign alerts changes substantially. For example, going from 139 to 276 false positives is an additional 137 unnecessary alerts over the same validation period. Conversely, Double-DQN misses 71 fewer attacks than default LightGBM. These changes describe a trade-off, not a uniformly better classifier.

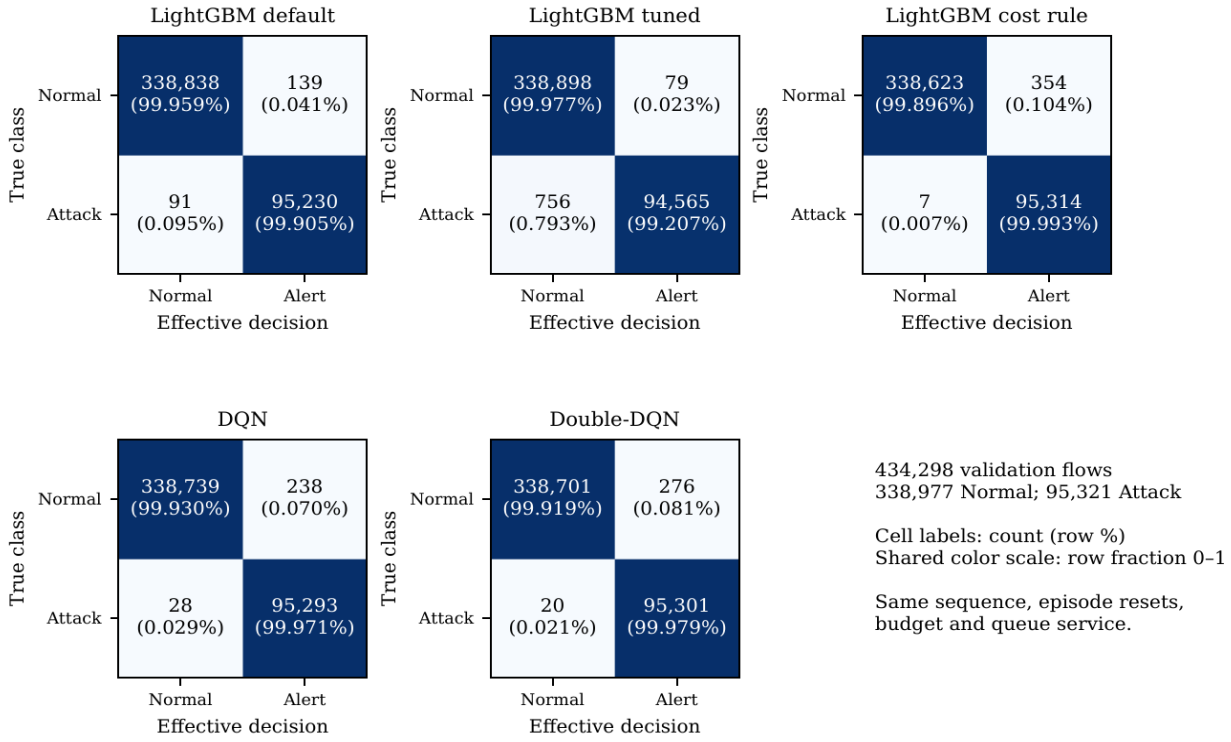


Fig. 6. Original-policy confusion matrices on 434,298 identical validation observations. Entries show flow counts and within-true-class percentages.

Although the decision target is binary, preserving the original attack category labels matters. In the archived category-level validation results, only two Heartbleed, five Infiltration, and three SQL-injection flows remain in the validation split. Reliable claims about rare-category improvement cannot be made from such small supports, even when aggregate binary recall approaches 100%. Furthermore, a higher global attack recall need not mean better coverage of each rare attack family.

TABLE IV. Exploratory recall by original attack category (validation only)

Attack category	Support	LGB	DQN	DDQN
Bot	293	78.84	95.56	97.95
Web brute force	224	99.55	100.00	100.00
Infiltration	5	60.00	80.00	80.00
SQL injection	3	66.67	100.00	100.00
Heartbleed	2	100.00	100.00	100.00

Recall is a percentage, for original policies only. LGB denotes default LightGBM. DDQN denotes Double-DQN. Category labels are evaluation metadata, not RL inputs. Extremely small supports preclude dependable statistical claims.

E. What the Policy Does and Does Not Learn

The category breakdown in Table IV supplies useful diagnostic context. The Bot class, with 293 validation flows, has substantially lower LightGBM recall than the aggregate attack class, while both RL agents identify more of those observations. This is suggestive, but it is not evidence that the policy has learned an attack-family-specific response: category labels were not provided to it, and false-positive changes must be assessed at the same time. Heartbleed, with only two validation observations, illustrates a more fundamental limitation. Perfect recall on two flows has little evidential value for unseen Heartbleed behavior.

The simulator also exposes a distinction between *learning an alert threshold* and *learning operational adaptation*. In principle, a DQN receiving a queue and a budget could change its alert policy when resources become scarce. In these recorded evaluations, however, no selected rule caused a queue or budget rejection. Consequently, the present results cannot establish that the resource variables improved action selection. A simpler nonsequential explanation of the observed decisions remains plausible. An appropriate

next experiment is to vary service capacity and alert budget while holding traffic and detection costs constant, and to test whether a learned policy adapts in a way that static thresholds cannot.

F. Interpreting the Detection Trade-off

A false-positive reduction target should always be conditional on an explicit attack-recall requirement. Choosing a much higher threshold can trivially suppress benign alerts, but it may also discard attacks that a SOC would have considered important. The tuned LightGBM result in Table II demonstrates this effect: its 79 false positives are attractive in isolation, yet the corresponding 756 false negatives are substantially worse than the 91 misses of the default rule. Conversely, the cost-sensitive threshold misses only seven attacks, but raises a larger number of benign alerts. The selected DQN and Double-DQN policies lie between these endpoints, rather than dominating them.

For operational interpretation, a useful future comparison would express decisions per fixed traffic volume at several predeclared recall levels, with real investigation effort assigned to the alert stream. Statistical testing must also account for repeated or correlated flows, rather than treating every network record as an independent incident. Such a protocol would distinguish meaningful gains in analyst workload from changes that merely shift the classifier's operating point.

VII. LIMITATIONS AND PLANNED EXTENSIONS

The present evidence has several boundaries. First, *all numerical performance claims are validation-only*. Checkpoints and offsets were chosen using these observations, while the 420,012-row final-test partition remains untouched in the available record. Reporting the selected validation figures as final generalization results would be optimistic. Second, only seed 42 was trained for each RL algorithm. Smooth aggregate plots and fixed-sequence replay do not replace independent-seed variation, confidence intervals, or a held-out replication.

Third, the environment is an offline SOC simulation: it lacks verified traffic chronology, authenticated flow/session independence, and real analyst feedback. Within-partition duplicate groups further limit the independence of sampled observations. All selected operating points incurred zero resource rejections, so the benefit of learning explicit queue or budget management has not been demonstrated in a binding-resource regime. The 16-variable state is only an approximate operational state representation.

Fourth, the reward design uses stipulated values (FP penalty 1, FN penalty 20) instead of externally measured incident-response costs. Model rankings may change if the ratio or resource capacity changes. The archived experiment contains no completed *isolated-factor* ablations for uncertainty, queue context, reward symmetry, signed SHAP inputs, or an abstain/review action. Therefore, the effect of these features is not estimated and cannot be asserted as a result. Finally, differences in timing benchmarks do not permit trustworthy inference-speed ranking without repeated controlled tests.

Future research will address these limitations in a controlled order: (i) extend the search grid and predefine a genuinely comparable recall operating band, (ii) train three or more independent seeds and run the proposed reward and state ablations, (iii) freeze the model selection protocol and evaluate the reserved final-test partition once, (iv) reconstruct verified chronological flow streams and incorporate realistic analyst queue dynamics, and (v) evaluate cross-dataset transfer to CIC-UNSW-NB15 after strict feature-semantic validation. An abstain/review action and SHAP-derived policy features should be introduced only with complete accounting for review effort and inference overhead. Together, these extensions will test whether the temporal decision machinery offers something unattainable through simpler supervised thresholds.

VIII. CONCLUSION

This paper investigated whether a cost-sensitive DQN or Double-DQN could act as a decision layer above a strong binary LightGBM intrusion classifier. On a common CIC-IDS2017 validation set, the trained policies detected a higher fraction of attacks than the default supervised rule, but both also increased false positives. Double-DQN reduced the number of missed attacks from 91 to 20, while benign alerts rose from 139 to 276. A simple cost-sensitive LightGBM rule achieved a lower simulated total cost than either learned policy under the recorded settings. Thus, the

initial experiment demonstrates an implementable sequential decision framework, not a verified superiority of DRL for false-positive reduction. The principal contribution at this stage is a traceable comparison protocol and a clear account of where the DRL hypothesis remains unproven. Stronger claims require independent test evaluation, repeated training seeds, real ablations, and a more representative sequential operating environment.

REFERENCES

[1] G. Ke *et al*., "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 3146–3154.

[2] V. Mnih *et al*., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015, doi: 10.1038/nature14236.

[3] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with Double Q-learning," in *Proc. 30th AAAI Conf. Artificial Intelligence*, 2016, pp. 2094–2100, doi: 10.1609/aaai.v30i1.10295.

[4] J. Tang, H. Cao, and Q. Zhu, "Research on optimization of network intrusion detection system based on deep reinforcement learning," in *Proc. 2025 Int. Conf. Big Data, Communication Technology and Computer Applications (BDCTA)*, 2025, doi: 10.1145/3727505.3727523.

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

[6] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Information Systems Security and Privacy (ICISSP)*, 2018, pp. 108–116, doi: 10.5220/0006639801080116.

[7] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[8] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6, doi: 10.1109/CISDA.2009.5356528.

[9] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," in *Proc. Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6, doi: 10.1109/MilCIS.2015.7348942.

[10] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016, doi: 10.1109/COMST.2015.2494502.

[11] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symposium on Security and Privacy*, 2010, pp. 305–316, doi: 10.1109/SP.2010.25.

[12] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, Art. no. 20, pp. 1–22, 2019, doi: 10.1186/s42400-019-0038-7.

[13] M. A. Ferrag, L. Maglaras, H. Janicke, J. Jiang, and L. Shu, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, Art. no. 102419, 2020, doi: 10.1016/j.jisa.2019.102419.

[14] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Proc. Network and Distributed System Security Symposium (NDSS)*, 2018, doi: 10.14722/ndss.2018.23204.

[15] M. Ring, D. Schlör, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019, doi: 10.1016/j.cose.2019.06.005.

[16] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying deep learning approaches for network traffic prediction and intrusion detection," in *Advances in Big Data and Cloud Computing*. Singapore: Springer, 2019, pp. 229–243, doi: 10.1007/978-981-13-1882-5_15.

[17] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proc. 9th EAI Int. Conf. Bio-*

inspired Information and Communications Technologies, 2016, pp. 21–26, doi: 10.4108/eai.3-12-2015.2262516.

[18] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017, doi: 10.1109/ACCESS.2017.2762418.

[19] M. Aloqaily, S. Otoum, I. A. Ridhawi, and Y. Jararweh, "An intrusion detection system for connected vehicles using deep reinforcement learning," *Computer Networks*, vol. 210, Art. no. 108979, 2022, doi: 10.1016/j.comnet.2022.108979.

[20] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis, and R. Atkinson, "Shallow and deep networks intrusion detection system: A taxonomy and survey," arXiv:1701.02145, 2017.

[21] M. Tavallaee, N. Stakhanova, and A. A. Ghorbani, "Toward credible evaluation of anomaly-based intrusion-detection methods," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 40, no. 5, pp. 516–524, 2010, doi: 10.1109/TSMCC.2010.2048428.

[22] S. Huang, K. Lei, Y. Fan, Y. Liu, and Z. Yang, "Survey of network intrusion detection methods from the perspective of feature representation," *Sensors*, vol. 20, no. 22, Art. no. 6425, 2020, doi: 10.3390/s20226425.

[23] J. Zhang, Y. Wang, and M. P. Fanti, "A deep reinforcement learning approach for dynamic cyber-attack detection in industrial control systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, pp. 4107–4116, 2021, doi: 10.1109/TII.2020.3027312.

[24] M. Louk and S. Tama, "Explainable artificial intelligence for intrusion detection systems: A survey," *IEEE Access*, vol. 11, pp. 115947–115972, 2023, doi: 10.1109/ACCESS.2023.3322723.

[25] Y. Zhang and S. Wang, "Deep reinforcement learning for network intrusion detection: A survey," *Journal of Network and Computer Applications*, vol. 135, pp. 123–135, 2019.