DEVELOPING A HYBRID INTRUSION DETECTION AND PREVENTION SYSTEM USING A COMBINATION OF SUPERVISED LEARNING AND REINFORCEMENT LEARNING

By

Olzhas Raiganiyev

A THESIS

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Computer Science

LAWRENCE TECHNOLOGICAL UNIVERSITY

2025

This thesis has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Computer Science.

Department of Math and Computer Science

Thesis Advisor:	*Dr. Fan Li*
Committee Member:	*Dr. Tao Liu*
Committee Member:	*Dr. George Pappas*
Committee Member:	*Dr. Fan Li*
Department Chair:	*Dr. Eric Martinson*

Contents

List of Figures

List of Tables

Abstract

The increasing complexity of cyberattacks has exposed the limitations of traditional intrusion detection systems, particularly in handling zero-day threats and minimizing false alarms. This study proposes a hybrid Intrusion Detection and Prevention System (IDPS) that integrates Supervised Learning (SL) for fast high-confidence classification with Deep Reinforcement Learning (DRL) for adaptively handling uncertain cases. The system is evaluated on two benchmark datasets: UNSW-NB15 and NSL-KDD. The hybrid model achieved 93.95% accuracy on UNSW-NB15 and 99.45% accuracy on NSL-KDD, significantly reducing false positives while maintaining balanced precision and recall. The computational time for model training and inference remained within practical limits, with end-to-end training completed in approximately 2–5 minutes. These results demonstrate the system's potential for scalable, real-time deployment in dynamic cybersecurity environments.

Chapter 1

Introduction

Fast development of digital technologies and vast computerization has greatly expanded the scope of potential attacks by adversaries. As the number of cyber threats are growing in sophistication and number a secure mechanism is needed to protect the sensitive data and to maintain the integrity of the computing systems. Intrusion detection and prevention systems (IDPS) is part of an overarching security strategy. These are meant to observe network or the system behaviour fore suspicious or policy breaches. IDPS can report such activities to administrators and even in certain configurations be programmed to respond in a proactive mode to stop or reduce threats [1]. IDPS technologies are generally classified into two: signature-based and anomaly-based. Signature-based IDPS use patterns that are pre-defined and its signature list will help detecting suspicious activities with high level of accuracy but it lacks the ability to detect zero day attacks. The anomaly based IDPS follow and establish the normal behavioral to detect breaches from this normal behavior, detect new attack types but suffer from usually higher false positive rates [2]. There has been a development from IDPS which has also integrated various sophisticated techniques, such as Machine Learning (ML) and Artificial Intelligence (AI) in order to improve detection and overcome new generation threats. These innovations are to overcome

the challenges of the classical IDPS, which are related to accuracy, false positive rate, and the requirement for real-time response [3].

1.1 Machine Learning in Cybersecurity

The increasing complexity and escalation of cyber threats require the adoption of sophisticated cybersecurity technologies. ML as an AI subfield has become a important tool in cybersecurity because it enables systems to detect, prevent and respond to cyber incidents more efficiently. ML algorithms can process extensive datasets to find patterns and irregularities which represent potential security risks. The proactive strategy enables threat detection that conventional signature-based systems cannot identify. Unsupervised learning methods can identify unusual patterns within network traffic which may indicate security breaches or malware attacks [4]. Researchers use deep learning as a ML technique to address various cybersecurity problems including spam detection, malware classification, and intrusion detection systems (IDS). These models are capable of independently extracting features from raw data which boosts the accuracy and capability of threat detection systems [5]. The integration of ML into cybersecurity systems makes real-time threat intelligence capabilities which allow defenses to adapt dynamically to new attack patterns. The ability to adapt to new security challenges plays a crucial role in defending against zero-day vulnerabilities and advanced persistent threats (APTs) that manage to evade traditional security measures [6]. Despite the fact, its potential advantages, the application of ML in cybersecurity entails specific obstacles. Effective implementation of ML models in cybersecurity needs ongoing research because data quality issues and adversarial attack vulnerabilities along with model transparency problems persist. The ongoing maintenance of ML-based security solutions remains a crucial priority [7]. ML delivers significant advantages to cybersecurity systems by employing intelligent methods that adapt and scale to detect and defend against emerging threats.

1.2 Limitations of Traditional IDPS

Traditional IDPS have been instrumental in safeguarding digital infrastructures. However, with the increasing sophistication of cyber threats, these systems exhibit several limitations that hinder their effectiveness.

One significant challenge is the high rate of false positives and negatives. Signature-based IDPS rely on known threat patterns, making them ineffective against obfuscated attacks. This limitation often results in genuine threats going undetected while benign activities trigger alerts, leading to alert fatigue among security personnel [8].

Traditional IDPS struggle with encrypted traffic. As more network communications adopt encryption, these systems find it challenging to inspect packet contents, thereby reducing their visibility into potential threats [9]. This blind spot can be exploited by attackers to bypass security measures. Another concern is the scalability of traditional IDPS. With the exponential growth of network traffic, these systems face difficulties in processing and analyzing vast amounts of data in real-time. This limitation can lead to delays in threat detection and response, compromising the security posture of organizations [9].

Additionally, the static nature of traditional IDPS makes them ill-equipped to adapt to the dynamic threat landscape. They often require manual updates and lack the capability to learn from new attack patterns autonomously, rendering them less effective against zero-day exploits and advanced persistent threats [8].

In summary, while traditional IDPS have served as foundational security tools, their limitations necessitate the integration of advanced technologies, such as ML and artificial intelligence, to enhance threat detection and response capabilities in modern cybersecurity frameworks.

1.3 Motivation

The evolving complexity and volume of cyberattacks—such as zero-day exploits and advanced persistent threats—continue to outpace the capabilities of traditional IDPS. Signature-based systems are limited to known threats and require frequent manual updates. Anomaly-based systems, while capable of identifying previously unseen attacks, often suffer from high false positive rates because uncommon but benign behaviors may be flagged as malicious [10].

ML methods have been increasingly applied to address these limitations. Supervised Learning (SL) algorithms, such as XGBoost, offer efficient classification and interpretability but are constrained by their reliance on labeled data and lack of adaptability to evolving threats [4]. Deep Reinforcement Learning (DRL), on the other hand, learns policies through interaction with an environment and has shown promise in modeling complex sequential behaviors, particularly in dynamic or adversarial contexts [11]. However, DRL models often require extensive training time, are computationally intensive, and may suffer from convergence instability.

This research is motivated by the opportunity to combine the strengths of SL and DRL in a hybrid IDPS framework. The system is designed to perform initial classification using XGBoost, a gradient-boosted decision tree algorithm known for its robustness and high accuracy on structured network traffic data. Each prediction is accompanied by a confidence score, and if this score falls below a predefined threshold, the instance is routed to a Deep Q-Networks (DQN) agent for further evaluation.

This layered architecture allows the system to process the majority of routine or clearly identifiable traffic efficiently using the SL model, while leveraging the adaptive decision-making capabilities of DRL for ambiguous or low-confidence cases. The DQN

agent is trained to recognize subtle patterns and improve classification decisions on complex or minority-class attacks that are often misclassified by traditional classifiers. This selective delegation reduces the computational burden on the DRL component while enhancing detection performance where SL alone may be insufficient. The hybrid pipeline, therefore, provides a practical balance between speed, accuracy, and adaptability—crucial properties for real-time intrusion detection in modern, dynamic network environments.

1.4 Research Objectives

The primary objective of this research is to investigate whether a hybrid IDPS, combining SL and DRL, can improve detection accuracy and adaptability in dynamic and heterogeneous network environments. To achieve this, the research aims to:

- Assess the effectiveness of a confidence-aware hybrid architecture in reducing false positives and improving detection of rare or ambiguous cyberattacks.
- Determine whether confidence-based routing improves classification performance compared to standalone SL or DRL approaches on benchmark intrusion datasets.
- Evaluate the hybrid model's ability to generalize across multiple attack types, including low-frequency classes such as R2L and U2R, using NSL-KDD and UNSW-NB15 datasets.
- Analyze the trade-off between detection accuracy and computational efficiency to assess feasibility for real-time deployment in practical network security scenarios.

These objectives guide the investigation into how hybrid learning techniques can be systematically applied to enhance the performance, reliability, and scalability of modern IDPS solutions.

1.5 Contributions

This thesis presents a novel hybrid IDPS that synergistically combines SL and DRL techniques to enhance cybersecurity measures. The primary contributions of this research are as follows:

1. *Hybrid SL-DRL Architecture:* Development of a two-layer hybrid IDPS architecture that integrates XGBoost for high-confidence classification and DQN for adaptive handling of uncertain cases. This architecture aims to leverage the strengths of both SL and DRL to improve detection accuracy and adaptability.

2. *Confidence-Based Thresholding Strategy:* Implementation of a static, confidence-based thresholding strategy to route ambiguous traffic instances from the SL layer to the DRL layer. This approach enhances the system's ability to handle uncertain predictions and improves detection of complex or low-frequency attack patterns while minimizing false positives.

3. *Comprehensive Evaluation:* Rigorous evaluation of the proposed hybrid model on benchmark datasets, namely UNSW-NB15 and NSL-KDD, demonstrating superior performance in terms of accuracy, precision, recall, and computational efficiency compared to traditional IDPS models.

Chapter 2

Related Work

The evolution of IDPS has been extensively studied across multiple domains, from rule-based designs to advanced data-driven methods. This section outlines the foundational categories of traditional IDS, highlighting their core mechanisms and known limitations. It then explores the role of ML and DRL in enhancing intrusion detection, particularly in dynamic and high-dimensional threat environments. Emphasis is placed on hybrid models that integrate SL and DRL to balance accuracy and adaptability. Finally, recent contributions from 2020 to 2025 are reviewed, and key limitations in current approaches are identified to motivate the proposed framework.

2.1 Traditional IDS

Traditional IDS are foundational components in cybersecurity, designed to monitor and analyze network or system activities to detect malicious behaviors. These systems are primarily categorized into two types: Signature-based Intrusion Detection Systems (SIDS) and Anomaly-based Intrusion Detection Systems (AIDS).

2.1.1 Signature-based Intrusion Detection Systems (SIDS)

SIDS operate by comparing observed activities against a database of known attack signatures. When a match is found, an alert is generated. This method is highly effective for identifying known threats with minimal false positives. However, it struggles to detect novel or obfuscated attacks, as it relies on predefined signatures. The effectiveness of SIDS is heavily dependent on the quality and comprehensiveness of its signature database. Poorly designed signatures can lead to increased false alarms, making the system less practical for deployment [12].

2.1.2 Anomaly-based Intrusion Detection Systems (AIDS)

AIDS establish a baseline of normal behavior and flag deviations from this norm as potential threats. This approach is advantageous for detecting previously unknown attacks. Nevertheless, it often suffers from high false-positive rates, as benign activities that deviate from the established norm can be misclassified as malicious. The challenge lies in accurately defining 'normal' behavior in dynamic network environments.

2.1.3 Limitations of Traditional IDS

While traditional IDS have been instrumental in early threat detection, they face several limitations:

- *Evasion Techniques*: Attackers can employ techniques to evade detection, such as fragmenting malicious payloads or mimicking legitimate traffic patterns.
- *Scalability Issues*: As network speeds and volumes increase, traditional IDS may struggle to process and analyze data in real-time.

- *Maintenance Overhead*: Regular updates to signature databases are required to maintain effectiveness, which can be resource-intensive.

To address these challenges, researchers have explored integrating ML techniques into IDS to enhance their adaptability and accuracy [10].

2.2 ML for Intrusion Detection

The integration of ML into IDS has significantly enhanced the detection of cyber threats. Unlike traditional systems that rely heavily on static rules or predefined signatures, ML-based IDS can learn from network traffic patterns and dynamically adapt to evolving attack strategies.

SL techniques, such as Decision Trees, Support Vector Machines (SVM), and k-Nearest Neighbors (k-NN), have been widely used for classifying network behaviors as normal or malicious [13]. These models are trained on labeled datasets and can achieve high accuracy in recognizing known attack patterns. Unsupervised learning approaches, such as clustering and anomaly detection algorithms, are utilized when labeled data is scarce. They aim to detect outliers in network traffic without prior knowledge of specific attacks, offering an effective means for identifying zero-day threats.

More recently, ensemble methods like Random Forest and XGBoost have shown improved performance by combining the predictions of multiple base classifiers, increasing both robustness and generalization capability [14]. Despite promising results, challenges remain. ML models may suffer from overfitting, bias due to imbalanced datasets, and vulnerabilities to adversarial attacks. Therefore, continuous research is focused on developing more resilient and interpretable ML-based IDS for practical deployment in complex network environments.

2.3 DRL for Adaptive Security

DRL has recently gained attention as a powerful approach for enhancing the adaptability of IDPS. Unlike SL models, which require extensive labeled datasets, DRL enables agents to learn optimal strategies through interaction with an environment, receiving feedback via rewards or penalties.

In cybersecurity contexts, RL and DRL-based systems can dynamically adjust detection policies in response to evolving attack patterns. Agents can be trained to identify subtle attack behaviors and make decisions that maximize long-term network security objectives [15]. This capability is particularly useful for defending against zero-day exploits, adversarial attacks, and other dynamic threats where static detection rules are insufficient.

One promising development is the application of DRL, which combines RL with DNN to handle high-dimensional state spaces typically found in complex network traffic datasets. Recent research demonstrates that DRL-based approaches outperform traditional ML models in environments characterized by highly dynamic and adversarial behaviors [16].

However, challenges remain, including ensuring the stability of DRL training, preventing overfitting to specific attack types, and managing exploration-exploitation trade-offs effectively. Future work continues to focus on making DRL-driven IDPS more robust, interpretable, and efficient for real-world deployment.

2.4 Hybrid Models for IDPS

Hybrid IDPS aims to take advantage of the strengths of multiple detection approaches, such as combining signature-based, anomaly-based, and ML techniques, to improve overall performance, accuracy, and adaptability in cybersecurity defense mechanisms.

Hybrid models typically integrate a fast and accurate classifier for known threats (e.g., decision trees, SVMs) with a more adaptive, flexible method such as clustering or DRL for unknown or evolving threats [17]. This combination allows hybrid IDPS to achieve lower false positive rates while maintaining high detection coverage across a wide range of attack types.

Recent work has shown that hybrid deep learning models, such as combining Convolutional Neural Networks (CNN) with Recurrent Neural Networks (RNN), can effectively capture both spatial and temporal features of network traffic for enhanced intrusion detection [18]. These hybrid deep models are particularly well-suited to handle complex, high-dimensional datasets like UNSW-NB15 and CICIDS-2017.

Moreover, ensemble hybrid approaches, where multiple classifiers vote or collaborate to make a final decision, have demonstrated improved robustness and generalization in intrusion detection tasks [19]. However, the main challenges for hybrid IDPS remain the increased computational overhead and difficulty in tuning multiple subsystems simultaneously. The growing adoption of hybrid models underlines their potential as a promising pathway for developing scalable, adaptive, and accurate intrusion detection solutions capable of coping with modern dynamic threat environments.

2.5 State of the Art

2.5.1 Recent Machine Learning Approaches (2020–2025)

In this subsection, we evaluate the performance of DRL models applied to IDS between 2020 and 2025. Two key datasets are analyzed: NSL-KDD and UNSW-NB15. The results are summarized in two tables, with Table 2.1 presenting the performance on the NSL-KDD dataset, and Table 2.2 detailing the performance on the UNSW-NB15 dataset.

Performance Trends on NSL-KDD Dataset: Table 2.1 presents comparative performance metrics of several hybrid and DRL models applied to the NSL-KDD dataset between 2020 and 2025. The table shows that accuracy scores have climbed progressively, with hybrid methods achieving notable improvements.

LS-SVM, reported in 2025, also displayed impressive performance with 99.3% accuracy and 99% across all other metrics. Despite relying on a classical support vector machine framework, its effectiveness stems from kernel optimization and precise decision boundary calibration. DRL - based methods also performed strongly. DQN in 2022 achieved 99.36% accuracy and 99.21% F1-score, showing its adaptability in real-time environments. Similarly, A-DQN (2021) recorded 97.20% accuracy and 97.80% F1-score, benefiting from action selection mechanisms tailored to network behavior patterns. MAFSID (2022), which uses multiagent RL for feature selection, reached 99. 10% accuracy and F1 score, illustrating the power of distributed learning in security contexts. Deep neural models like CNN (2024) and DNN (2021) also showed competitive results. CNN attained 98.91% accuracy and 91.41% F1-score, with its spatial feature learning ability aiding detection. However, DNN yielded a

lower F1-score of 77%, indicating challenges in generalizing across diverse attack categories. ELSTM-RNN with LPPSO, proposed in 2023, achieved a strong 96.89% accuracy, though other metrics were not disclosed. Nevertheless, its integration of long short-term memory with particle swarm optimization signals the ongoing emphasis on time-aware learning and dynamic tuning of hyperparameters. In general, the results in Table 2.1 emphasize the growing effectiveness of hybrid systems in detecting complex intrusion patterns in the NSL-KDD dataset.

Table 2.1
Comparative Performance of ML and DRL Models on the NSL-KDD Dataset (2020–2025)

Year	Method	ACC(%)	PR(%)	RC(%)	F1(%)
2023	ELSTM-RNN with LPPSO [20]	96.89	/	/	/
2025	LS-SVM [21]	99.3	99	99	99
2021	DNN [22]	94	91	92	77
2024	CNN [23]	98.91	96.75	91.96	91.41
2022	DQN [24]	99.36	99.07	99.36	99.21
2022	MAFSID [25]	99.10	/	/	99.10
2021	A-DQN [26]	97.20	96.50	99.10	97.80
2020	DQN [27]	98.71	97.35	98.71	98.30
2020	DON [28]	91.40	92.80	90.20	91.48

2.5.2 Performance on UNSW-NB15 Dataset

Table 2.2 focuses on recent model performance using the UNSW-NB15 dataset, which is known for its realistic simulation of modern attacks. While the benchmark is more complex than NSL-KDD due to a broader feature set and attack diversity, recent hybrid methods have made notable progress. The DQN model from 2020 stands out with a solid performance, achieving 91.80% accuracy, 93.20% precision, 91.70% recall, and 92.44% F1-score. These results highlight DQN's suitability for dynamically changing threat landscapes and its ability to learn effective policies over time through reward feedback.

In 2022, the DRL-RBFNN method reached 82.62% accuracy and similar values for precision, recall, and F1-score (around 82.5%). By combining DRL with radial basis function neural networks, the model attempts to generalize better across traffic patterns, but its performance suggests possible limitations in learning from imbalanced data or real-time traffic. Deep SARSA, also from 2022, achieved an accuracy of 85.09%. Although precision, recall, and F1-score were not specified, the use of SARSA over Q-learning reflects a shift toward on-policy methods, which can offer more stable convergence in safety-critical environments. Despite slightly lower accuracy metrics compared to NSL-KDD results, these findings in Table 2.2 confirm the feasibility of applying DRL techniques to more demanding real-world scenarios.

Table 2.2
Deep Reinforcement Learning Model Performance on UNSW-NB15 Dataset (2020–2022)

Year	Method	ACC(%)	PR(%)	RC(%)	F1(%)
2022	Deep SARSA [24]	85.09	/	/	/
2022	DRL-RBFNN [29]	82.62	82.40	82.60	82.49
2020	DQN [28]	91.80	93.20	91.70	92.44

2.6 Observed Gaps

Despite significant advancements in ML and hybrid intrusion detection approaches, several critical gaps remain in the current research landscape:

- *Scalability to Large-Scale Networks:* Many recent hybrid models are evaluated on relatively small or curated datasets such as NSL-KDD or CICIDS-2017. Their ability to handle large-scale, real-world traffic with millions of concurrent flows remains underexplored [30].
- *High Computational Cost:* Deep hybrid architectures often require substantial

computational resources for training and inference, limiting their practical deployment in resource-constrained environments like IoT devices [31].

- *Generalization to Unseen Attacks:* Although hybrid systems improve detection rates for known attacks, their performance against zero-day or highly obfuscated threats is inconsistent. There is a pressing need for methods that better generalize beyond the training distribution [32].

- *Adversarial Robustness:* Emerging adversarial attacks designed to fool ML-based IDS models have revealed vulnerabilities in many hybrid systems, indicating a lack of robustness against sophisticated evasion tactics.

Addressing these gaps remains essential for the broader adoption of hybrid IDPS models in real-world cybersecurity infrastructure.

Chapter 3

Methodology

This section presents the foundational components and technical design of the proposed hybrid IDPS. The architecture is centered on two complementary learning paradigms: gradient-boosted decision trees (XGBoost) for fast and confident classification, and DRL (specifically DQN) for adaptive handling of uncertain cases. Section 3.1 provides an in-depth overview of XGBoost, covering its theoretical basis, internal mechanics, training workflow, and advantages for structured intrusion detection datasets. It also explores its integration into hybrid pipelines, discusses parameter tuning strategies, and highlights implementation challenges. Section 3.2 introduces RL and DQN, outlining its ability to generalize across dynamic security environments by leveraging policy optimization and state-action representations. Various improvements and practical applications of DQN in cybersecurity are discussed, along with architectural refinements. Section 3.3 articulates the rationale behind the hybrid design, detailing why specific components were chosen and how they align with the goals of scalability, interpretability, and responsiveness to ambiguous threats.

3.1 XGBoost Algorithm

Extreme Gradient Boosting (XGBoost) is a highly scalable and efficient ML algorithm widely adopted for classification, regression, and ranking tasks [33]. Designed to optimize both computational speed and model accuracy, XGBoost utilizes a sophisticated ensemble approach that builds multiple decision trees iteratively, with each new tree correcting errors made by previous ones. Its ability to handle sparse data, prevent overfitting through regularization, and exploit parallel computation has made it a popular choice for cybersecurity applications, including IDS [34].

3.1.1 Core Concepts

XGBoost is based on the principle of gradient boosting, where weak learners (typically shallow decision trees) are trained sequentially. At each iteration, a new tree is fit to the residuals (errors) of previous trees, aiming to minimize a specified loss function. This gradual improvement leads to a strong overall model capable of capturing complex patterns in the data [35].

The objective function of XGBoost is composed of two main components: a loss term that measures model prediction error and a regularization term that penalizes model complexity. It is formally defined as:

$$\mathcal{L}(\theta) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k), \tag{3.1}$$

where:

- $\mathcal{L}(\theta)$: The overall objective (cost) function to be minimized, parameterized by θ,

which denotes the set of all model parameters (e.g., split conditions, leaf weights).

- i: Index over the training instances, from 1 to n.
- $\hat{y}_i$: The predicted value for the i-th training instance.
- y_i: The ground truth label for the i-th instance.
- $l(\hat{y}_i, y_i)$: The differentiable convex loss function (e.g., logistic loss or mean squared error), which quantifies the difference between the predicted value and the actual label.
- k: Index over the individual trees in the ensemble.
- f_k: The k-th decision tree (function) in the model.
- $\Omega(f_k)$: The regularization term applied to tree f_k, defined to penalize model complexity (e.g., number of leaves or magnitude of leaf weights). It helps control overfitting by discouraging overly complex trees.

3.1.2 Workflow of XGBoost

The overall workflow of XGBoost is visually illustrated in Figure 3.1. The process involves the following steps:

1. *Data Preparation:* Initial data are collected and processed into a format suitable for training. In cybersecurity applications, features are often extracted from network traffic or system logs [36].
2. *Model Initialization:* The algorithm starts with an initial model that could simply predict the mean of the target variable.

3. *Fitting Weak Learners:* A decision tree is fit to the residual errors from the previous model. These errors are computed by taking the negative gradient of the loss function.

4. *Prediction Residuals:* After each tree is trained, it produces predictions on the training data. The residuals are updated accordingly, reflecting the difference between actual and predicted labels.

5. *Weighted Data Handling:* Subsequent trees focus more on the examples that previous trees misclassified by assigning higher weights to such samples [37].

6. *Ensemble Aggregation:* Predictions from all the weak learners are aggregated to make the final decision. For classification tasks, the aggregation typically involves summing the outputs and applying a threshold.

7. *Final Prediction:* The ensemble output is then used to predict the class labels for unseen data.

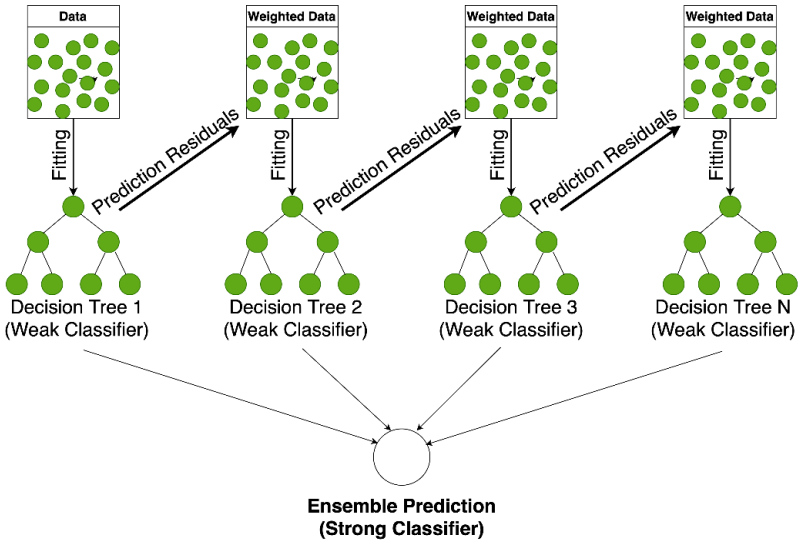

Figure 3.1: XGBoost Workflow for Ensemble Boosting Decision Trees.

3.1.3 Advantages of XGBoost in Intrusion Detection

XGBoost offers several benefits particularly suited to the cybersecurity domain:

- *Efficiency:* Its sparse-aware algorithm and block structure allow for faster computation, crucial when processing large-scale network datasets like UNSW-NB15 or NSL-KDD [38].
- *Handling Imbalanced Data:* XGBoost effectively handles class imbalance through scale_pos_weight hyperparameter tuning, a key challenge in intrusion detection where attack samples are often rare compared to normal traffic.
- *Feature Importance:* The algorithm provides feature importance scores, which help to understand the behavior of the model and identify critical security features [39].

3.1.4 XGBoost in Hybrid IDPS Frameworks

In this study, XGBoost serves as the first-layer classifier in the hybrid IDPS architecture. The model initially predicts whether the traffic instance is benign or malicious, accompanied by a confidence score derived from the prediction probability distribution. High-confidence predictions are accepted immediately, while low-confidence samples are passed to the second-layer DRL module for refined decision-making.

Such a cascading hybrid design addresses the trade-off between detection speed and accuracy. XGBoost filters clear-cut cases quickly while deferring uncertain samples to a more adaptive but slower secondary model [40].

3.1.5 Hyperparameter Configuration of XGBoost

The performance of the XGBoost classifier is highly sensitive to several hyperparameters. In this study, the following settings were used and fine-tuned based on validation performance:

- Number of Trees (n_estimators = 400): Determines how many boosting rounds are performed. More trees can improve learning but also increase computation time.
- Maximum Tree Depth (max_depth = 7): Limits the depth of each decision tree to avoid overfitting and reduce complexity.
- Learning Rate ($\eta = 0.01$): Controls the contribution of each tree in the ensemble. A smaller value leads to slower but more robust learning.
- Subsample Ratio (subsample = 0.8): Randomly samples 80% of the training data for each tree to prevent overfitting.
- Column Sampling (colsample_bytree = 0.8): Samples 80% of the features for each tree to reduce inter-tree correlation.

These parameters were selected after empirical evaluation using validation data to balance between generalization and computation time. The final configuration ensured both detection accuracy and training efficiency in large-scale network traffic classification tasks.

3.1.6 Challenges and Considerations

While XGBoost excels in many domains, it does come with certain challenges that must be carefully considered in the context of intrusion detection. One major limitation is its memory usage; training XGBoost on very large datasets can consume

substantial computational resources, particularly memory, due to the ensemble of numerous decision trees. Additionally, interpretability remains a concern. Although XGBoost provides feature importance scores, the aggregated decision logic across hundreds of trees can still be difficult to decipher, making it harder to extract intuitive explanations behind specific predictions. Another challenge lies in the model's sensitivity to hyperparameter configurations. If hyperparameters are not tuned appropriately, the model may suffer from overfitting or underperformance, especially in imbalanced or noisy datasets.

Nonetheless, with rigorous cross-validation, proper feature selection, and thoughtful parameter tuning, XGBoost maintains its status as one of the most effective supervised learning methods for cybersecurity tasks. In summary, XGBoost offers a scalable, fast, and highly effective foundation for high-confidence classification in hybrid intrusion detection and prevention system (IDPS) frameworks. Its role in the proposed model capitalizes on these strengths, ensuring both efficiency and robustness when dealing with complex and evolving threat patterns.

3.2 DRL and DQN

RL has emerged as a foundational framework in ML where agents learn optimal behaviors through interaction with an environment [41]. Unlike SL, which relies on explicit labeled data, RL allows the agent to discover strategies by maximizing cumulative rewards. This approach makes RL especially suitable for cybersecurity scenarios where attack patterns continuously evolve, and labeled instances of new threats are scarce.

3.2.1 Foundations of Reinforcement Learning

In RL, the environment is typically modeled as a Markov Decision Process (MDP) defined by a tuple (S, A, P, R, γ), where S represents the set of states, A the set of possible actions, P the transition probability matrix, R the reward function, and γ the discount factor [42]. The agent learns a policy $\pi(a|s)$ that maximizes the expected cumulative discounted reward:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right], \tag{3.2}$$

where:

- π: A policy that maps states to actions.
- π^*: The optimal policy that maximizes the expected cumulative reward.
- $\mathbb{E}[]$: Expectation over possible state-action trajectories under policy π.
- t: The discrete time step index.
- $\gamma \in [0, 1]$: The discount factor that determines the present value of future rewards.
- $R(s_t, a_t)$: The immediate reward received after taking action a_t in state s_t.
- s_t: The state of the environment at time step t.
- a_t: The action taken by the agent at time step t.

A major goal in Reinforcement Learning (RL) is to estimate the optimal action-value function $Q^*(s, a)$, which represents the maximum expected cumulative reward achievable from state s by taking action a and following the optimal policy thereafter.

3.2.2 Challenges in Classical Reinforcement Learning

Despite its theoretical appeal, applying classical RL algorithms, such as tabular Q-learning or State-Action-Reward-State-Action, to real-world cybersecurity problems is highly challenging. One significant issue is the high dimensionality of the state space, as intrusion detection systems often rely on hundreds of features to characterize network activity. This complexity renders tabular methods impractical, as they cannot scale efficiently with such large input spaces. Another obstacle is the sparsity of rewards, which arises from the fact that malicious activities represent a small fraction of overall traffic. This imbalance makes it difficult for the agent to receive frequent and informative feedback during training, slowing down the learning process and biasing it toward the majority class. Furthermore, when function approximators like neural networks are introduced to handle complex environments, the learning process can become unstable. Q-values may oscillate or diverge, particularly when inappropriate update strategies or network architectures are used [11]. These limitations have motivated the transition toward DQN, which combine RL with the representational power of deep learning to overcome the shortcomings of traditional approaches.

3.2.3 Introduction to DQN

DQN, introduced by Mnih et al. [11], combine the representational power of deep learning with the sequential decision-making capabilities of RL. DQN approximates the action-value function $Q(s, a; \theta)$ using a DNN, where θ represents the network's parameters. This formulation allows the agent to estimate the expected cumulative reward for taking an action a in a given state s, which is essential for making informed decisions in high-dimensional environments such as network intrusion detection.

Two critical innovations make DQN effective and stable during training. The first is the use of experience replay, a mechanism in which the agent's experiences—each consisting of a state, action, reward, and next state tuple (s, a, r, s')—are stored in a memory buffer. During training, mini-batches of these experiences are sampled uniformly at random to update the network. This process breaks the strong temporal correlations present in sequential data and promotes more stable and efficient learning. The second innovation involves the use of target networks. In this technique, the target Q-value is computed using a target network denoted as $Q(s', a'; \theta^-)$. In this expression, s' refers to the next state the agent transitions to after executing an action, a' represents the next action under consideration in that state, and θ^- corresponds to the parameters (weights) of the target Q-network. This target network is a periodically updated replica of the main Q-network and is used to compute stable target values during training. The use of $Q(s', a'; \theta^-)$ helps mitigate training instabilities by decoupling the target calculation from the rapidly fluctuating parameters of the primary network. The overall training objective is to minimize the difference between the predicted Q-value and the target value derived from the Bellman equation. The loss function used to train the network is given by:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \right)^2 \right], \tag{3.3}$$

where:

- $L(\theta)$: The loss function used to train the Q-network, parameterized by θ.
- $\mathbb{E}_{(s,a,r,s')}$: Expectation over experience tuples (s, a, r, s') sampled from the replay buffer.
- s: Current state.
- a: Action taken in state s.
- r: Reward received after taking action a in state s.

- s': Next state resulting from action a.
- $\gamma \in [0, 1]$: Discount factor for future rewards.
- $Q(s, a; \theta)$: The predicted Q-value for state s and action a using the current Q-network with parameters θ.
- $Q(s', a'; \theta^-)$: The target Q-value for the next state s' and action a', computed using the target network with fixed parameters θ^-.
- $\max_{a'} Q(s', a'; \theta^-)$: The maximum estimated Q-value over all possible actions a' in the next state s', representing the best expected future reward.

Here, γ denotes the discount factor, balancing immediate and future rewards. By optimizing this loss function through stochastic gradient descent, the network learns to approximate optimal action-value estimates, ultimately enabling it to make effective decisions even in complex and uncertain environments such as intrusion detection systems.

3.2.4 DQN Workflow Overview

The workflow of DQN is illustrated in Figure 3.2, which shows the interaction loop between the agent and the environment in a reinforcement learning setting.

At each time step, the *agent* observes the current *state* s from the *environment*. In the context of intrusion detection, this state represents a feature vector that encodes characteristics of a network session, such as protocol type, byte counts, or connection duration.

The state s is passed into a DNN, which approximates the action-value function $Q(s, a; \theta)$, where θ denotes the network parameters. The network outputs Q-values

for all possible actions, estimating the expected future reward for taking each action in the given state.

An action a is selected based on a policy derived from these Q-values, typically using an ε-greedy strategy. This policy balances exploitation (choosing the action with the highest Q-value) with exploration (selecting a random action with some probability). In this application, actions correspond to decisions such as classifying traffic as benign or malicious.

Once the action is taken, the environment transitions to a new *next state* s' and returns a *reward* r, which reflects the quality of the action. The transition tuple (s, a, r, s') is stored in a replay memory buffer.

During training, batches of these stored experiences are sampled from memory to update the network. The DNN is trained to minimize the temporal difference error between the predicted Q-value and the target Q-value, which is computed using the Bellman equation and a fixed target network with parameters θ^{-}.

Through this iterative interaction and training loop, the agent improves its decision-making policy over time, allowing it to handle ambiguous or previously unseen patterns in the data more effectively.

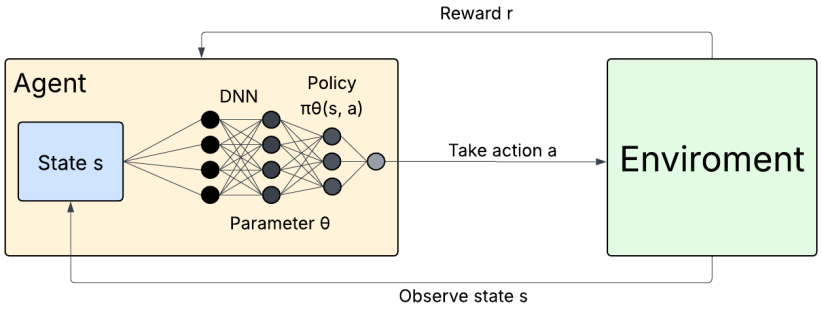

Figure 3.2: Deep Q-Network Workflow.

3.2.5 Hyperparameter Configuration of DQN

The DQN model used in this study is configured with a feedforward neural network consisting of two hidden layers of sizes 512 and 256, respectively. To enhance convergence and reduce overfitting, Leaky ReLU activations are combined with batch normalization and a dropout rate of 0.3. A small learning rate of $\alpha = 0.0002$ is selected to allow stable updates during training.

To address class imbalance, focal loss is utilized instead of cross-entropy, configured with parameters $\alpha = 0.9$ and $\gamma = 2.0$, which gives more focus to hard-to-classify instances. Focal loss modifies the standard cross-entropy loss by down-weighting easy examples and emphasizing those with low predicted probability. The focal loss for a multi-class classification problem is defined as:

$$\mathcal{L}_{\text{focal}} = -\alpha_t (1 - p_t)^{\gamma} \log(p_t), \tag{3.4}$$

where p_t is the predicted probability for the true class, α_t is a weighting factor for class imbalance, and γ is the focusing parameter that adjusts the rate at which easy examples are down-weighted.

Training is conducted in mini-batches of 512 samples, balancing gradient stability with efficiency. The replay memory stores up to 55,000 transitions (s, a, r, s'), which are randomly sampled to break temporal correlation and stabilize learning. Additionally, the target network is updated every 700 steps to prevent instability in Q-value estimation. Gradient clipping is applied with a norm threshold of 1.5 to avoid exploding gradients. Finally, the DQN is trained for 30 epochs, providing sufficient interaction to learn effective decision policies without overfitting.

These hyperparameter values are selected based on best practices in DRL literature

and empirically tuned for optimal classification performance on ambiguous and low-confidence instances in the NSL-KDD and UNSW-NB15 datasets.

3.2.6 Advantages of DQN in Cybersecurity

Recent research highlights the advantages of DQN in addressing the challenges posed by dynamic and evolving cybersecurity environments [30, 31]. One key strength of DQN lies in its adaptability. Unlike static models, DQN learns through continuous interaction with the environment, allowing it to detect previously unseen threats by refining its policies over time based on trial-and-error learning. This makes it particularly useful in settings where threat patterns frequently change. Additionally, DQN exhibits strong scalability due to its integration with DNN. This capability enables the model to effectively process high-dimensional feature spaces that are common in network traffic data, without requiring an exhaustive enumeration of possible states. Another notable benefit is its ability to refine decision-making in uncertain scenarios. In cases where a SL model yields ambiguous or low-confidence predictions, DQN can leverage temporal information and state transitions to make more informed and context-aware decisions. These properties collectively make DQN a compelling component in the construction of robust IDPS.

3.2.7 Applications of DQN in IDS

DQN has been increasingly adopted for various tasks within IDS, including network anomaly detection, malware classification, and automated intrusion response. One of its primary applications is in real-time anomaly detection, where network traffic is modeled as sequences of state transitions, allowing the agent to dynamically identify deviations from normal behavior that may signify malicious activity. Another

emerging use case is in the deployment of adaptive honeypots, where DQN agents are used to dynamically adjust deception strategies based on attacker behavior and policy learning. Furthermore, DQN has been explored for adaptive feature selection during model training. This enables the system to prioritize and learn from the most informative features, thereby enhancing generalization performance, especially in high-dimensional datasets [43]. These applications underscore the growing utility of DQN in building intelligent, responsive, and efficient security systems.

3.2.8 Design Enhancements for DQN

Given the complexity of cybersecurity environments, several enhancements to the original DQN architecture have been proposed to improve learning stability and detection performance. One such enhancement is Double DQN, introduced to address the issue of overestimation bias by decoupling action selection from action evaluation during Q-value updates [44]. Another significant advancement is Dueling DQN, which improves value estimation by separately modeling the state-value and advantage functions within the neural network architecture [16]. Additionally, Prioritized Experience Replay has been employed to focus training on the most informative or surprising transitions by assigning higher sampling probabilities to experiences with larger temporal-difference errors [45]. These modifications are especially beneficial for handling rare attack patterns and highly imbalanced data, where traditional uniform sampling or naïve value estimation may fall short.

3.3 Design Rationale

The design of the proposed hybrid IDPS integrates structured learning and DRL to leverage the strengths of both paradigms while mitigating their respective limitations.

This section outlines the motivations, assumptions, and decisions that guided the architectural and algorithmic choices behind the two-layer system using XGBoost for initial classification and DQN for adaptive decision refinement.

3.3.1 Motivation for Hybrid Architecture

Modern cybersecurity threats are diverse, rapidly evolving, and often ambiguous. Supervised models such as decision trees, support vector machines, or ensemble learners like XGBoost perform well when trained on clean, labeled datasets. However, their performance degrades in the presence of distribution drift, class imbalance, or novel attack patterns. On the other hand, DRL offers adaptability through learning from interaction, making it suitable for refining decisions under uncertainty. Combining both allows the system to use XGBoost for efficient high-confidence classification, while DQN handles complex or ambiguous instances where confidence is low.

3.3.2 Choice of XGBoost for Layer 1

XGBoost (Extreme Gradient Boosting) was selected as the first-layer classifier due to its performance in tabular cybersecurity datasets, robustness to noise, and interpretability. It excels in handling high-dimensional data, supports parallel computation, and provides probability scores that facilitate confidence-based routing of uncertain instances. Additionally, XGBoost's built-in feature importance evaluation supports our preprocessing stage by guiding the feature selection process. This contributes to reducing overfitting and computational load.

In preliminary experiments, multiple classifiers were evaluated on the preprocessed NSL-KDD dataset without using the two-layer hybrid architecture. The classification results are summarized in Table 3.1.

Table 3.1
Standalone Classifier Accuracy on NSL-KDD (Preliminary Evaluation)

Model	**Accuracy (%)**
Decision Tree	75.02
Random Forest	78.13
XGBoost	81.10

These results demonstrate that XGBoost offered the best standalone performance among the tested models. In addition to its accuracy, XGBoost provides calibrated confidence scores, supports efficient training on high-dimensional structured data, and enables fast inference—making it an ideal candidate for the first-layer classifier in the hybrid IDPS pipeline.

3.3.3 Threshold-Based Routing Strategy

A soft decision boundary is established using confidence thresholds derived from XGBoost's prediction probabilities. Instances with confidence exceeding a pre-defined threshold (e.g., 0.97) are considered reliable and accepted directly. Conversely, samples falling below this threshold are flagged as uncertain and passed to the second-layer DQN for deeper analysis. This thresholding mechanism not only improves system interpretability but also optimizes resource usage by limiting the DRL agent's workload to complex cases only.

3.3.4 Motivation for DQN in Layer 2

The DRL component, DQN, serves as the decision refinement agent. RL is advantageous in cybersecurity scenarios where reward signals (correct detections) are sparse, and decisions must be learned dynamically based on feedback. By employing DQN, the system generalizes over large state spaces via DNN while optimizing long-term detection accuracy through trial-and-error learning. The DQN is trained on instances where XGBoost was uncertain or incorrect, allowing it to specialize in difficult or minority-class samples (e.g., R2L or U2R in NSL-KDD). To justify the use of reinforcement learning in the second layer, several standalone RL models were evaluated on the preprocessed NSL-KDD dataset without the integration of the first-layer XGBoost classifier. The classification results are summarized in Table 3.2.

Table 3.2
Standalone RL Model Accuracy on NSL-KDD (Preliminary Evaluation)

Model	**Accuracy (%)**
Q-Learning	75.20
Double DQN	80.13
Deep Q-Learning	84.86
DQN	88.65

Although each of these models underperformed compared to the full hybrid system, DQN demonstrated the best generalization capability and scalability through deep function approximation. Its ability to adapt to ambiguous or previously unseen traffic patterns, especially in cases flagged as uncertain by XGBoost – supports its selection as the second-layer model for refining low-confidence predictions through policy-based learning.

3.3.5 Replay Memory and Target Networks

Incorporating experience replay and target networks within DQN ensures training stability. Replay memory stores transitions (s, a, r, s'), where s is the current state (i.e., the input feature vector of a network traffic instance), a is the action taken (e.g., allow or block), r is the reward received based on the correctness of the action, and s' is the next state observed after the action. These transitions are sampled randomly during training to reduce correlation between sequential samples and improve convergence. The target network, updated periodically, provides fixed targets for Q-value computation to avoid oscillations during learning. These mechanisms are crucial when learning from highly imbalanced and noisy intrusion data.

3.3.6 Reward Function Design

The reward function in DQN is designed to penalize false negatives more than false positives, as undetected attacks have higher consequences in real-world settings. For example, a misclassification of an attack as benign incurs a higher penalty than misclassifying a benign request as an attack. This asymmetric reward shaping encourages the model to become more sensitive to subtle threat patterns, particularly those overlooked by the first-layer classifier.

3.3.7 Dataset Compatibility

The architecture is designed to be modular and compatible with benchmark intrusion detection datasets like NSL-KDD and UNSW-NB15. Both datasets offer labeled traffic samples with categorical and continuous features, which XGBoost handles naturally. DQN, in turn, receives normalized state vectors after feature selection,

ensuring consistency across datasets and experiments. This cross-dataset compatibility allows comparative evaluations and increases the model's applicability in varied cybersecurity contexts.

3.3.8 Scalability and Efficiency

To ensure computational feasibility, especially for real-time or streaming scenarios, the hybrid system restricts DQN's workload to a subset of instances. This selective activation strategy reduces the number of forward passes and backpropagation steps needed by the RL agent. Additionally, training phases for XGBoost and DQN are decoupled and executed in parallel pipelines, allowing scalability on multi-core or GPU-enabled systems.

3.3.9 Data Splitting Configuration for Experiments

For both the NSL-KDD and UNSW-NB15 datasets, two different data splitting strategies were explored. In the first approach, the training and testing subsets provided by the original datasets were used as-is: KDDTrain+ and KDDTest+ for NSL-KDD, and the official training and test CSV files for UNSW-NB15. These configurations represent realistic evaluation settings with no overlap between training and testing samples. In the second and primary configuration used for the majority of experiments in this thesis, the training and testing sets were merged into a single dataset. From this combined dataset, 80% of the samples were randomly selected for training and the remaining 20% for testing. This approach was adopted to provide a more comprehensive and balanced distribution of samples during training, which in turn led to improved performance metrics. All performance tables and analyses throughout this thesis—unless otherwise stated—are based on this 80/20 mixed-data configuration.

Chapter 4

Proposed Method

This section presents the architecture and operational workflow of the proposed two-layer hybrid IDPS. The design integrates a SL model (XGBoost) with a DRL model (DQN) to achieve both high-confidence detection and adaptive classification. It begins by outlining the entire architecture, detailing how input data progresses through various stages, from preprocessing and feature selection to the generation of confidence scores and final decision making. XGBoost serves as the first layer, performing efficient initial classification, while the second layer, based on DQN, is invoked for low-confidence cases to further refine ambiguous predictions. The section also discusses thresholding strategies, RFE techniques, and the rationale behind confidence-aware routing. A detailed analysis of the DQN framework follows, including its state representation, action space, reward function, and enhancements such as experience replay and target networks. Finally, the entire hybrid workflow is summarized in a clear and interpretable pseudocode format to guide practical implementation.

4.1 Overview of the Two-Layer Hybrid Architecture

The proposed hybrid IDPS is structured around a two-layer architecture that combines the strengths of SL and DRL. This design aims to achieve fast and confident classification on routine instances, while adaptively refining decisions on uncertain or complex cases. An overview of the entire workflow is depicted in Figure 4.1, capturing the essential stages from data preparation to action execution.

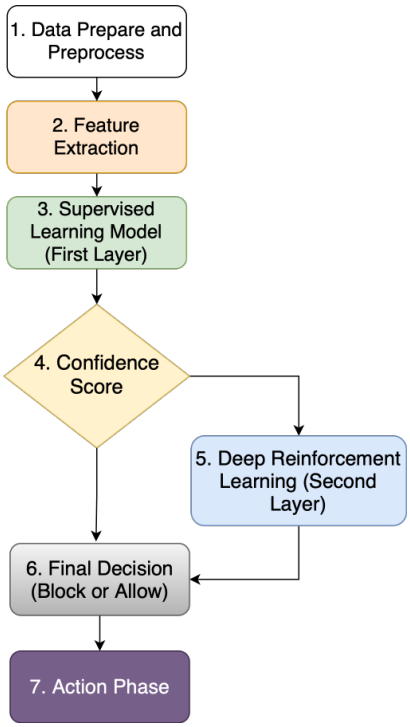

Figure 4.1: Overview of the Two-Layer Hybrid IDPS Architecture.

4.1.1 Data Preparation and Preprocessing

The pipeline begins with collecting, cleaning, and preprocessing the network traffic data. In this stage, features are normalized, categorical attributes are encoded,

and redundant or irrelevant fields are removed to improve learning efficiency. Benchmark datasets such as NSL-KDD [46] and UNSW-NB15 [47] are utilized to ensure standardization and comparability with prior work.

4.1.2 Feature Extraction

After preprocessing, relevant feature extraction is performed to retain informative characteristics that support discrimination between normal and malicious traffic. Feature selection techniques are employed to enhance generalization and reduce computational complexity, which is critical for ensuring real-time responsiveness in cybersecurity systems.

4.1.3 Supervised Learning Model (First Layer)

The first decision-making layer is powered by the XGBoost algorithm [33]. XGBoost is chosen for its superior performance on structured datasets, efficient handling of missing values, and interpretability. The model is trained on labeled traffic data to predict whether a given instance is benign or malicious. Alongside the prediction, XGBoost produces a probability confidence score for each decision.

4.1.4 Confidence Score Evaluation

A crucial element of the system is the evaluation of the model's confidence in its predictions. If the confidence score exceeds a predefined threshold (e.g., 97%), the instance is classified based on XGBoost's decision. Otherwise, instances falling below the threshold are flagged as uncertain and routed to the second-layer RL agent for further analysis. This dynamic routing ensures that only challenging cases are handled by the more computationally intensive RL component, optimizing resource usage.

4.1.5 Deep Reinforcement Learning Model (Second Layer)

The second layer consists of a DQN agent [11], designed to adaptively refine decisions on ambiguous instances. The DQN receives the processed features and state representation of uncertain samples and interacts with a simulated environment to maximize cumulative detection accuracy. The agent learns through trial and error, thereby continuously improving its ability to detect sophisticated or novel attack patterns.

4.1.6 Final Decision Making

After processing by either the first-layer or second-layer model, a final decision is made to either block or allow the network traffic. This decision is derived from the highest confidence prediction available, ensuring robustness against adversarial or stealthy attacks.

4.1.7 Action Phase

The system concludes by executing the determined action — either blocking the malicious traffic or allowing legitimate packets through the network. Alerts and logs are also generated for administrative analysis and system audit trails.

4.1.8 Design Benefits

This layered approach offers multiple benefits: (1) high efficiency by filtering out easy decisions quickly, (2) robustness through adaptive learning on difficult cases, and (3) scalability to large-scale network environments without overwhelming computational resources. Furthermore, by decoupling static SL from dynamic RL, the

system achieves a balance between detection performance and adaptability, a critical requirement for modern cybersecurity operations [9, 15].

Overall, the two-layer hybrid architecture represents a synergistic fusion of ML and DRL paradigms, engineered specifically to meet the demands of evolving and uncertain cyber threat landscapes.

4.2 Preprocessing, Feature Selection, and Thresholding

Effective data preprocessing and feature selection are critical stages for ensuring the performance, robustness, and generalization of any ML model, particularly in cybersecurity contexts. Given the complexity and high dimensionality of network traffic datasets like NSL-KDD [46] and UNSW-NB15 [47], careful design of these stages substantially impacts the success of the IDPS. This section outlines the sequential workflow applied to preprocess the data, select the most relevant features, and apply confidence-based thresholding for model routing. The overview of this preprocessing and feature selection methodology is illustrated in Figure 4.2.

4.2.1 Data Preparation and Preprocessing

Data preprocessing involves several key steps to ensure the quality and effectiveness of ML models. Label cleaning is performed to correct redundant, inconsistent, or missing labels, which helps preserve the integrity of class definitions. Features such as packet sizes, durations, or byte counts are normalized using techniques like Min-Max scaling or standardization, preventing learning algorithms from being biased toward features with larger numerical ranges. For datasets such as NSL-KDD, categorical

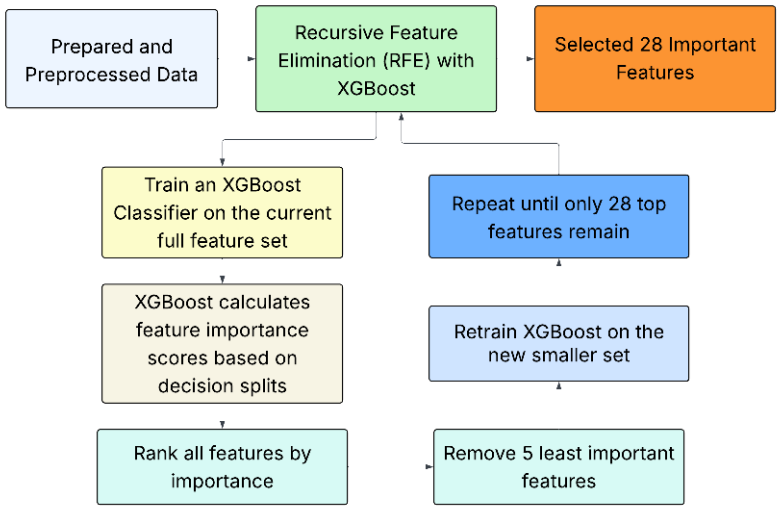

Figure 4.2: Preprocessing, Feature Selection, and Thresholding Workflow.

attributes—such as protocol type (e.g., TCP, UDP, ICMP)—are transformed into numerical representations using one-hot encoding or ordinal encoding. Additionally, to address class imbalance, particularly in cases where certain attack types have very few samples, techniques like SMOTE are used for oversampling, or the loss function is adjusted during training to give more weight to underrepresented classes.

By establishing a clean and standardized input format, preprocessing mitigates noise and inconsistencies that could otherwise degrade the model's learning quality.

4.2.2 Recursive Feature Elimination with XGBoost

Once preprocessing is complete, Recursive Feature Elimination (RFE) driven by XGBoost [33] is applied to perform feature selection. The motivation behind this choice lies in the observation that intrusion detection datasets often contain redundant, irrelevant, or weakly informative features [48]. Eliminating such features reduces computational overhead and enhances model interpretability. The process, as depicted in Figure 4.2, unfolds as below.

1. *Training an Initial XGBoost Classifier*: A full-featured XGBoost model is trained using all available features on the preprocessed data.

2. *Feature Importance Calculation*: XGBoost computes feature importance scores based on their contribution to decision tree splits, particularly using Gain, Cover, and Frequency metrics [36]. Gain, which is the default metric, measures the improvement in the loss function (e.g., log-loss) achieved by a feature when it is used in a split. The gain for a feature f is calculated as:

$$\text{Gain}_f = \sum_{t \in T_f} \Delta L_t, \tag{4.1}$$

   where T_f is the set of all nodes where feature f is used, and ΔL_t is the reduction in loss at node t. The final importance score is then normalized across all features:

$$\text{Importance}_f = \frac{\text{Gain}_f}{\sum_{j=1}^{K} \text{Gain}_j}. \tag{4.2}$$

3. *Ranking Features*: All features are ranked according to their calculated importance scores.

4. *Pruning the Least Important Features*: The five least important features are systematically removed.

5. *Retraining the Model*: The XGBoost model is retrained on the reduced feature set to reassess feature importances dynamically.

6. *Iteration*: Steps 2–5 are repeated until only the top 28 features remain, providing an optimal balance between model performance and feature set compactness.

The RFE approach offers several benefits. First, it enhances generalization by reducing model overfitting through the removal of noisy or irrelevant features. Second, it improves efficiency, as working with a lower-dimensional feature space speeds up

both training and inference processes. Lastly, RFE supports better interpretability, since the features that are retained can be analyzed to gain insights into underlying cyberattack patterns.

Empirical studies on NSL-KDD and UNSW-NB15 datasets demonstrate that using only the most important 28 features retains over 95% of the full model's predictive power, while halving training times [49, 50].

4.2.3 Feature Selection Strategy Justification

Choosing XGBoost for feature selection aligns with prior works where tree-based methods demonstrated superior capability in identifying non-linear and high-order feature interactions in cybersecurity datasets [47]. Moreover, XGBoost's in-built handling of missing values and automatic regularization prevents overfitting during feature ranking.

Alternative methods like Principal Component Analysis or mutual information scores were considered but discarded, because they either project features into latent space (losing semantic interpretability) or fail to capture complex feature interactions relevant for attack detection.

4.2.4 Thresholding on Confidence Score

Following feature selection, a trained XGBoost classifier is used to produce probabilistic confidence scores for each prediction. Instead of making hard decisions solely based on maximum probability, a confidence thresholding mechanism is applied to balance precision and recall. Thresholding mechanism:

- If the XGBoost classifier predicts a class with confidence above a certain threshold

(e.g., 97%), the decision is accepted immediately.

- If the confidence falls below the threshold, the instance is flagged as "uncertain" and forwarded to the second-layer DRL agent for deeper evaluation.

4.2.5 Advantages of the Combined Workflow

The integrated preprocessing, feature selection, and thresholding workflow yields several advantages:

- *Reduced Model Complexity*: Shrinking the feature space to the top 28 features without compromising detection performance.
- *Robustness Against Overfitting*: Feature pruning regularizes model complexity, improving generalization on unseen attacks.
- *Improved Threat Adaptability*: The thresholding mechanism dynamically routes hard-to-classify samples for specialized handling.

In summary, the preprocessing and feature selection stage forms the foundation for the hybrid IDPS architecture. By leveraging XGBoost-driven RFE and intelligent thresholding, the system is empowered to prioritize both efficiency and detection effectiveness, preparing the data pipeline for optimal performance across both SL and DRL layers.

4.3 First-Layer: XGBoost for Confident Classification

The first layer of the proposed hybrid IDPS leverages the power of eXtreme Gradient Boosting (XGBoost) for high-confidence classification of network traffic data. This layer is responsible for quickly and accurately classifying most benign and malicious traffic samples without the need for deeper inspection unless uncertainty is detected. The objective is to reduce computational load on the DRL layer while preserving strong generalization and detection performance.

XGBoost is a decision-tree-based ensemble algorithm that uses gradient boosting to combine the outputs of multiple weak learners (typically shallow trees) into a strong classifier. It has demonstrated exceptional performance in many structured-data tasks due to its ability to handle missing data, robustness to outliers, regularization capabilities, and scalability [33].

4.3.1 Workflow Explanation

Figure 4.3 presents the internal workflow of the XGBoost classification and confidence estimation process used in our hybrid model.

The pipeline demonstrated in Figure 4.3 begins with the input data set containing the selected set of important characteristics. These features are chosen using RFE, where XGBoost's internal feature importance scores guide the selection process, as detailed in 4.2. Each sample in the dataset is encoded into an input feature vector $\mathbf{x}$, such as:

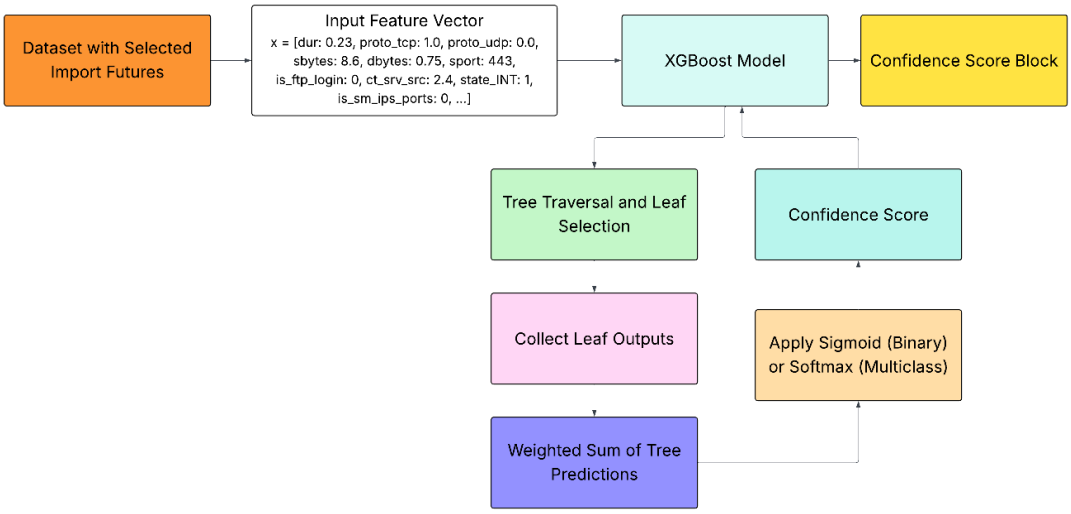

Figure 4.3: XGBoost Inference and Confidence Score Generation Workflow.

[dur: 0.23, proto_tcp: 1.0, sbytes: 8.6, ct_srv_src: 2.4,...].

This feature vector is then passed into the trained XGBoost model, which consists of an ensemble of decision trees. As the input propagates through the model, each tree independently evaluates the instance. Within each tree, internal nodes perform binary decisions based on feature thresholds (e.g., whether sbytes > 10), directing the instance down the tree until it reaches a terminal leaf node. The leaf node returns a real-valued output, often referred to as a logit or raw prediction.

Once all trees have been traversed, the leaf outputs are aggregated. This aggregation is achieved via a weighted summation, where each tree contributes a certain portion to the final score based on its optimization during gradient boosting. The cumulative output is then passed through a probability transformation layer. In binary classification settings, a sigmoid function is applied to scale the score into a probability between 0 and 1. For multi-class scenarios, a softmax function is employed to produce class-wise probability distributions.

The resulting value is treated as the model's confidence score regarding whether the given network traffic instance is benign or malicious. If this confidence score surpasses a predetermined threshold τ (for example, $\tau = 0.88$), the classification decision is accepted immediately. Otherwise, if the model is uncertain (i.e., the score falls below τ), the sample is deferred to the DRL layer for further evaluation. This architecture enables the system to make fast decisions when confident, while invoking deeper reasoning only for ambiguous cases.

4.3.2 Advantages of XGBoost in the First Layer

XGBoost is selected as the first-layer classifier due to several compelling advantages that align well with the goals of an efficient and reliable intrusion detection system. One of its primary strengths is its speed and scalability. XGBoost is engineered to handle large tabular datasets with ease by leveraging optimized data structures and multi-threaded parallelism, making it suitable for real-time or near-real-time applications. Furthermore, XGBoost offers a high degree of model interpretability. It exposes internal feature importance metrics which are instrumental in selecting and analyzing the most influential input features, thereby facilitating a more transparent ML workflow.

Another notable advantage is the reliability of its confidence scores. Since XGBoost outputs probabilistic predictions, these values can be directly interpreted as confidence estimates, which serve a critical role in our two-layer architecture by guiding the threshold-based routing to DRL. This property makes XGBoost particularly suitable for systems that rely on dynamic decision paths. In addition, XGBoost demonstrates robust generalization performance. Through the use of L1 and L2 regularization techniques, it mitigates the risk of overfitting, ensuring that the model retains its effectiveness when confronted with unseen attack patterns in the test data. Previous

studies have validated its consistency and adaptability in network security applications [38, 40, 51], making it a dependable choice for our hybrid detection system.

4.3.3 Prediction Decision Routing

Once XGBoost produces the confidence score c, the system evaluates whether to trust the prediction. A dynamic threshold τ is selected during validation (e.g., $\tau = 0.88$) to balance precision and recall. The decision logic is as follows:

- If $c \geq \tau$, classify the sample as either 'normal' or 'malicious' based on the label with the highest probability.
- If $c < \tau$, the sample is routed to the DQN layer for a second opinion.

This confidence-aware routing ensures that the DQN is only engaged on difficult or ambiguous samples, conserving computational resources while enhancing detection robustness.

Suppose that an input sample yields $p = 0.91$ from XGBoost. With $\tau = 0.88$, the model confidently classifies it as malicious. Conversely, a sample with $p = 0.67$ would be routed to DQN. The role of first layer is essential to offload most classification decisions from the computationally heavier DQN agent. As shown in prior studies, hybrid models employing XGBoost as a front-end achieve competitive results across multiple datasets, including NSL-KDD and UNSW-NB15 [36, 52]. By integrating the confidence-driven gating mechanism, our hybrid IDPS capitalizes on XGBoost's strengths for fast, high-certainty predictions while retaining DQN's adaptivity for ambiguous or novel threats.

4.4 Confidence Score

The confidence score plays a crucial role in the decision-making process of the proposed hybrid intrusion detection architecture. Acting as a bridge between the first-layer supervised model (XGBoost) and the second-layer DRL model (DQN), it quantifies predictive certainty and determines whether a given instance should be accepted or delegated for further evaluation. When a data instance passes through the XGBoost model, it traverses multiple decision trees trained during gradient boosting. Each tree contributes a numerical output, or logit, representing the model's internal estimate. These logits are aggregated through a weighted sum to produce a final score, which is then passed through a probability transformation function. In binary classification, a sigmoid function is applied to normalize the output to a range of 0 to 1. For multi-class problems, a softmax function distributes probabilities across classes.

The confidence score is derived by first calculating the *logit*, which is the sum of outputs from all individual decision trees:

$$\text{logit}(x) = \sum_{m=1}^{M} f_m(x), \tag{4.3}$$

where $f_m(x)$ is the output of the m^{th} decision tree, and M is the total number of trees. The logit is then passed through a sigmoid function to yield a confidence score between 0 and 1:

$$\text{Confidence Score} = \sigma(\text{logit}(x)) = \frac{1}{1 + e^{-\text{logit}(x)}}. \tag{4.4}$$

Here, $e \approx 2.71828$ is Euler's number. The resulting value represents the model's probabilistic certainty about its prediction.

The resulting probability value is interpreted as the model's confidence score, representing how likely it considers a data instance to belong to a particular class. In intrusion detection, this reflects how strongly the model believes a given session is benign or malicious. A value near 1 or 0 indicates high certainty, while values near 0.5 suggest ambiguity. To use this score in the proposed architecture, a threshold τ is defined. If the confidence exceeds τ (e.g., 0.88), the prediction is accepted and passed to the action phase. Otherwise, it is forwarded to the DQN layer for further inspection.

The threshold value of $\tau = 0.88$ was selected based on validation set experiments that maximized the F1-score, a metric that balances precision and recall. Multiple candidate thresholds were evaluated using a precision-recall curve, and 0.88 consistently provided the best trade-off between true positive rate and false positive control on the NSL-KDD dataset. This mechanism introduces adaptability, clearly classifiable cases are efficiently handled by XGBoost, minimizing latency. In contrast, ambiguous cases are routed to DQN, which can use contextual state information and policy-driven reasoning to improve classification. The confidence score thus serves as a decision gate between fast classification and deeper analysis.

Tuning τ controls the trade-off between performance and accuracy. A higher threshold increases sensitivity, forwarding more cases to DQN, potentially enhancing detection but raising computational cost. A lower threshold prioritizes speed but may overlook edge cases. Confidence scoring also offers diagnostic insight: low-confidence predictions across the dataset may signal weak feature representation or insufficient training data for specific classes. This feedback supports iterative improvements in feature engineering and reward shaping in the DRL layer.

4.5 Second-Layer: DRL (DQN) for Adaptive Classification

The second layer of the proposed hybrid architecture employs a DQN to refine the classification of uncertain instances that are routed from the first layer due to low confidence scores. While the XGBoost model is responsible for confidently classifying the majority of routine traffic, the DQN layer introduces adaptability and learning flexibility by leveraging the dynamics of DRL. This layer is activated only when the prediction confidence of XGBoost falls below a predefined threshold, thus optimizing computational resources while maintaining robust detection accuracy.

DRL is particularly well-suited for environments where the relationship between input and output is not always deterministic or where continuous adaptation is essential. In cybersecurity, especially in intrusion detection, new attack vectors and data patterns emerge frequently, necessitating an approach that can adapt over time. The DQN framework learns a policy that maps states to actions by interacting with the environment, observing transitions, and optimizing a reward-based objective. This interaction loop allows the agent to make increasingly refined predictions based on contextual and temporal information not easily captured by static supervised models.

In the proposed method, the DQN model receives as input a feature vector representing the network session. This vector includes a selected set of attributes such as connection duration, protocol types, byte counts, and flag indicators. These features are encoded as the agent's state and processed through a DRL that estimates the Q-values for each possible action. The Q-values represent the expected cumulative future reward of taking a certain action from a given state and following the current policy thereafter.

Figure 4.4 illustrates the end-to-end workflow of the DQN component in the proposed system. Starting from the left, a network session with its selected features is represented as a numerical vector. This vector becomes the agent's observable state, denoted as s, which is input into a DRL to estimate the Q-values associated with different actions. Based on these values, the agent selects an action a, which is then executed in the environment. The environment responds by transitioning to a new state and optionally providing a reward signal. This reward is used to refine the neural network during training but is not utilized during inference. The result of the environment interaction informs the final decision about whether the network session is malicious or benign. The DQN policy thereby learns not only from the current instance but also from the long-term impact of its actions on classification accuracy.

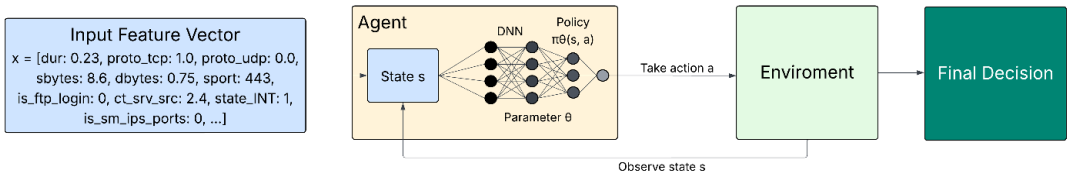

Figure 4.4: DQN-based Adaptive Classification Pipeline for Uncertain Predictions.

4.5.1 State Representation

In this model, the state space is defined by the same input feature vector used by the XGBoost model. However, unlike static classification, the DRL agent interprets these features within the context of temporal dependencies and potential downstream consequences of its actions. Each state s is a vector of normalized numeric values that encode protocol indicators, byte flow statistics, session timing, and flag attributes. Because DQN operates in a continuous state space, the neural network is capable of generalizing across similar inputs and learning nuanced distinctions between benign and malicious patterns. Importantly, the use of the same feature space between the

SL and DRL layers ensures compatibility and allows smooth transition of uncertain instances between layers.

4.5.2 Action Space

The action space in this system is discrete and consists of two possible outputs: allow or block. These actions correspond to the decision of whether the network session is benign (allow) or malicious (block). In DQN, the policy is derived by selecting the action with the highest Q-value at each state. During training, exploration is encouraged through the use of an epsilon-greedy policy, which balances exploration of new actions and exploitation of the current best-known actions. During inference, however, the model operates deterministically by choosing the highest-valued action.

4.5.3 Reward Function

The reward function is a critical component in shaping the learning behavior of the DQN agent. It incentivizes correct classification and penalizes incorrect actions, thus guiding the agent toward optimal policies. In the training phase, if the agent correctly classifies an uncertain instance as either benign or malicious, it receives a positive reward (e.g., +1). If it misclassifies the instance, it receives a negative reward (e.g., -1). This binary reward function is simple yet effective in stabilizing learning. Additionally, episodes can be defined for batch processing or sliding windows of network sessions, allowing the reward to accumulate and represent longer-term performance metrics such as detection rate and false positive rate.

4.5.4 Experience Replay and Target Networks

To ensure stable learning, the DQN architecture incorporates two widely-used techniques: experience replay and target networks. Experience replay stores past transitions in a buffer and samples them randomly to break temporal correlations during training. This technique prevents the agent from overfitting to recent samples and promotes more general learning. The use of a target network, which is a periodically updated copy of the Q-network, further stabilizes the learning process by decoupling the target Q-value computation from the current network's parameters.

4.5.5 Training and Testing Phases

It is important to note that the reward signal and environment interaction occur only during the training phase. During testing or deployment, the DQN model simply infers the best action based on the trained Q-values, without receiving any rewards or updating parameters. The decision is based solely on the policy learned during training. In deployment, the DQN operates as a deterministic classifier for uncertain samples passed from the XGBoost layer.

4.5.6 Model Adaptability and Advantages

The use of DQN in this context provides several strategic advantages. First, it enables dynamic decision-making, allowing the model to adjust based on observed states rather than relying solely on fixed thresholds. Second, it can learn complex patterns of attack behavior through repeated exposure and reward-guided refinement. This is particularly important in cybersecurity, where threat vectors evolve continuously and rigid classification boundaries may fail to capture emerging attack strategies. Third,

by limiting the activation of the DQN layer to only uncertain cases, the system avoids unnecessary computation and reduces latency for most routine traffic.

Another benefit of this architecture is its support for continual learning. The DQN agent can be periodically retrained using new labeled data to adapt to changing network conditions and updated attack signatures. This continuous learning loop enhances the long-term effectiveness and resilience of the IDS.

4.5.7 Confidence-Aware Decision Integration

The DQN layer plays a specialized role in resolving cases that fall below the confidence threshold of the XGBoost model. By handling only a fraction of the total traffic, it conserves computational resources while significantly improving classification precision in edge cases. The final decision is produced based on the policy output from the DQN, which is appended to the overall system pipeline and directly influences whether an instance is blocked or allowed.

In conclusion, the DQN-based second layer of the proposed hybrid architecture significantly enhances the system's ability to handle ambiguous or complex instances. Through policy-based refinement, reward-guided learning, and contextual state awareness, it introduces adaptive intelligence into the decision-making process. Its integration into the system enables a layered defense strategy that combines the efficiency of SL with the resilience of DRL, resulting in a robust and intelligent intrusion detection framework.

4.6 Pseudocode

To illustrate the operational logic of the proposed hybrid IDS, this section presents the pseudocode that governs the sequential decision-making process between the SL and DRL components. The two-layer architecture is structured such that the XGBoost classifier first evaluates the input instance and produces both a class prediction and a corresponding confidence score. Based on the magnitude of this confidence score and a predefined threshold value τ, the system determines whether the prediction should be accepted or delegated to the DQN module for further analysis.

The pseudocode in Algorithm 1 begins by iterating over all instances in the test set, where each instance has been preprocessed and reduced to a fixed-size vector of selected features. This vector is passed into the trained XGBoost model, which returns both a label and a probability estimate. The confidence score is compared against the threshold τ, and if it exceeds this value, the model's decision is accepted. Otherwise, the instance is routed to the DQN model, which evaluates the state, selects an action, and determines the final classification. In this way, the pipeline intelligently differentiates between high-confidence and ambiguous samples, thereby improving both the accuracy and efficiency of the overall system. In the second line of Algorithm 1, the output of the XGBoost prediction function is represented as $[y_{\text{xgb}}, p]$. Here, y_{xgb} refers to the predicted class label for instance x produced by the XGBoost model, while p represents the corresponding confidence score for that prediction. The score p typically results from applying a sigmoid or softmax transformation to the model's raw logit output and quantifies how confident the model is in assigning the label y_{xgb}. This confidence score is subsequently compared to a threshold τ to decide whether to accept the prediction or escalate it to the reinforcement learning layer.

Algorithm 1 Hybrid XGBoost + DQN Inference Pipeline

Require: Preprocessed test dataset D, trained XGBoost model M_{XGB}, trained DQN agent π_θ, confidence threshold τ

for each instance x in D **do**
  $[y_{\text{xgb}}, p] \leftarrow M_{\text{XGB}}.\text{predict}(x)$
  if $p \geq \tau$ **then**
    $y \leftarrow y_{\text{xgb}}$ ▷ Accept confident prediction
  else
    $s \leftarrow \text{construct\_state}(x)$
    $a \leftarrow \pi_\theta(s)$ ▷ DRL policy decision
    $y \leftarrow \text{map\_action\_to\_label}(a)$
  end if
  Append y to prediction list
end for
return All final predicted labels

This pseudocode emphasizes the conditional logic that governs the decision delegation between XGBoost and DQN layers. The threshold-based routing mechanism is crucial to controlling the balance between speed and accuracy. Moreover, the use of a state-construction function for DQN ensures that the input format remains consistent and compatible with the agent's training. The final decision, whether taken directly from XGBoost or generated by DQN, is added to the prediction list to be used for performance evaluation. The design encapsulated in Algorithm 1 represents the core operational engine of the hybrid detection system, enabling intelligent handling of uncertainty in network traffic classification.

Chapter 5

Experimental Setup

This section presents the experimental framework used to develop, train, and evaluate the proposed two-layer hybrid intrusion detection system. It begins with a description of the two benchmark datasets used: NSL-KDD and UNSW-NB15, both of which are widely recognized in the cybersecurity research community for evaluating intrusion detection models. Detailed attention is given to dataset preparation procedures, including label encoding, numerical feature scaling, and the cleaning of irrelevant or redundant attributes. Feature selection was conducted using recursive elimination guided by XGBoost's internal importance metrics to ensure dimensional efficiency without compromising detection accuracy. The datasets were then split into training and testing subsets using stratified sampling to preserve class distributions. The training phase involves supervised learning for the XGBoost model and a reward-driven RL cycle for DQN, each optimized for performance and generalization. Finally, the testing phase evaluates the system's ability to classify unseen data and respond adaptively to uncertain predictions. All steps are aligned with reproducible practices and designed to validate the model's robustness under diverse network traffic conditions.

5.1 Datasets

In this research, we employ two benchmark datasets for network intrusion detection: NSL-KDD and UNSW-NB15. The utilization of multiple datasets strengthens our evaluation framework and ensures that the proposed two-layer hybrid approach can generalize across different network environments and attack patterns.

5.1.1 NSL-KDD Dataset

The NSL-KDD dataset was introduced by Tavallaee et al. [46] as an enhanced alternative to the widely used KDD Cup 99 dataset. It addresses critical limitations of its predecessor, particularly the presence of redundant and duplicate records that often caused ML classifiers to exhibit biased performance. Developed at the University of New Brunswick, the NSL-KDD dataset comprises a total of 125,973 training records and 22,544 testing records. Each record includes 41 features along with a class label that identifies whether the instance is normal or represents a specific type of network intrusion.

The dataset categorizes attacks into four principal classes. Denial of Service (DoS) attacks aim to disrupt services by overwhelming resources, while Probe attacks attempt to collect information about the target system in preparation for further exploitation. User to Root (U2R) attacks involve a user gaining unauthorized access to root-level privileges, and Remote to Local (R2L) attacks occur when a remote attacker gains local access to a system, typically through credential theft or exploitation of software vulnerabilities [52].

The 41 features of NSL-KDD are divided into four conceptual groups. Basic features, spanning features 1 through 9, are derived from raw TCP/IP connection statistics

and include attributes such as protocol type, service, and connection status. Content features, from features 10 to 22, are drawn from packet payloads and capture domain-specific indicators of network intrusions. Time-based traffic features, ranging from feature 23 to 31, compute statistical patterns within a two-second time window, thereby capturing short-term temporal dynamics of network activity. Finally, host-based traffic features, from feature 32 to 41, analyze statistical behavior over a window of 100 connections, emphasizing host-level interactions and long-term usage patterns.

Moreover, numerical features are normalized to ensure that all input variables contribute proportionally during model training. This preprocessing pipeline, as supported by previous studies such as Ingre and Yadav [49], enhances the effectiveness of learning algorithms and ensures compatibility with both tree-based and neural network-based classifiers.

5.1.2 UNSW-NB15 Dataset

The UNSW-NB15 dataset, created by the Australian Centre for Cyber Security in 2015, represents a more contemporary network environment with modern attack patterns [47]. It contains 257,673 records, each characterized by 49 features including the class label. This dataset was generated using the IXIA PerfectStorm tool to create a hybrid of real modern normal activities and synthetic contemporary attack behaviors. UNSW-NB15 classifies attacks into nine categories: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. This expanded taxonomy offers a more diverse range of attack vectors compared to NSL-KDD. The feature space is organized into:

- *Flow Features:* Attributes derived from packet headers and connection establishment.
- *Basic Features:* Information extracted directly from packet headers.
- *Content Features:* Data extracted from packet payload.
- *Time Features:* Temporal statistics of packet arrivals.
- *Additional Generated Features:* Derived attributes that capture complex relationships.

Preprocessing UNSW-NB15 requires handling both binary and nominal categorical features (proto, service, state) and normalizing numerical features to prevent bias from features with larger ranges.

5.1.3 Comparative Analysis and Relevance to DRL-based IDPS

NSL-KDD and UNSW-NB15 differ significantly in their feature composition and attack representation. While NSL-KDD focuses on traditional network attacks with a well-established feature set, UNSW-NB15 encompasses modern attack vectors and network behaviors with an expanded feature space. The former has been extensively studied and provides a baseline for comparison, while the latter presents more realistic network traffic patterns reflective of contemporary cyber threats [50]. These datasets are particularly suitable for DRL-based intrusion detection for several reasons. First, their multi-dimensional feature spaces provide rich state representations for the DRL agent to learn from. Second, the diverse attack patterns enable the agent to develop robust detection capabilities across various threat scenarios. Third, the clear distinction between normal and anomalous behaviors facilitates the formulation of meaningful reward functions that guide the learning process [15]. However, both

datasets present challenges for DRL implementation. The high-dimensional feature spaces necessitate effective dimensionality reduction techniques to optimize computational efficiency. Additionally, class imbalance—particularly pronounced in the rare attack categories like U2R in NSL-KDD—requires careful consideration in the reward structure to prevent the agent from developing biased policies. Our methodology addresses these challenges through feature selection algorithms and balanced sampling techniques integrated into the DRL framework.

5.2 Dataset Preparation

Effective dataset preparation is a fundamental prerequisite for training robust IDPS. In this work, we employ two benchmark datasets: NSL-KDD and UNSW-NB15. Both datasets are widely used in cybersecurity research and provide labeled instances of network traffic, allowing SL and RL models to be trained in a controlled environment. This section details the rationale for selecting these datasets, the characteristics of each dataset, and the preprocessing techniques employed to prepare them for ingestion by the XGBoost and DQN components.

5.2.1 Data Acquisition and Initial Formatting

Raw datasets were downloaded from their official repositories in CSV format. The NSL-KDD data includes the training set (KDDTrain+) and testing set (KDDTest+), both of which contain pre-labeled examples. Similarly, UNSW-NB15 comprises a full dataset split into training and testing partitions. Upon loading the data into memory using `pandas`, categorical features were identified for encoding, and numerical features were analyzed for scale consistency and missing values. For NSL-KDD, we retained all 41 input features, excluding the class and attack type labels. For UNSW-NB15,

49 features were initially loaded, with derived attributes (e.g., `attack_cat`) used for multi-class classification. Redundant identifiers and timestamps were removed from both datasets to avoid leakage and overfitting.

5.2.2 Label Normalization and Encoding

All classification tasks were performed under both binary and multi-class configurations. In binary mode, all attack types were grouped under the "malicious" class (label 1), while normal traffic was labeled as benign (label 0). In multi-class mode, NSL-KDD's 22 attack labels were consolidated into four broader attack categories (DoS, Probe, R2L, U2R) based on domain knowledge [48], while UNSW-NB15 used its original nine attack families. Categorical features such as `protocol_type`, `service`, and `flag` in NSL-KDD, and similar protocol attributes in UNSW-NB15, were one-hot encoded to transform them into binary vectors, ensuring compatibility with tree-based models like XGBoost. Label encoding was used for supervised training labels, while RL policies were initialized with the multi-class attack categories.

5.2.3 Numerical Feature Scaling and Cleaning

Despite XGBoost's resilience to unscaled data, standardization improved convergence speed and interpretability in downstream analysis. Numerical features were normalized using Min-Max scaling, mapping values to the [0, 1] range. Outlier filtering was also applied using z-score filtering to cap extreme values which could distort the learning process. Missing values were handled through dataset-specific logic. NSL-KDD contains no missing entries, but in UNSW-NB15, null values in features such as `sbytes` and `dbytes` were imputed using median values of the respective class. After cleaning, both datasets were validated for schema consistency.

5.2.4 Balancing and Splitting Strategy

Both datasets suffer from class imbalance—particularly the R2L and U2R classes in NSL-KDD and the Shellcode and Worm classes in UNSW-NB15. To mitigate this, we applied the Synthetic Minority Oversampling Technique (SMOTE) only on the training sets. The validation and test sets remained untouched to preserve evaluation fairness. Training, validation, and testing splits were handled as follows:

- *NSL-KDD:* We used KDDTrain+ as training data (80%) and validation data (20%). KDDTest+ served as the separate test set.
- *UNSW-NB15:* A random 80/20 split was applied to the training data for training and validation, while the original test set was reserved for final evaluation.

5.2.5 Feature Selection Pipeline

To reduce dimensionality and eliminate redundant signals, RFE was applied using XGBoost's built-in gain-based feature importance. The RFE procedure iteratively trained XGBoost on the current feature set, ranked all features by importance, removed the five least informative features, and retrained until only the top 28 features remained. This allowed us to maximize performance while reducing computational overhead for both layers of the hybrid IDPS. A full visualization of this selection process is illustrated in Figure 4.2.

5.2.6 Dataset Compatibility with Hybrid Pipeline

Following preprocessing, both datasets were formatted into NumPy arrays compatible with Scikit-learn and PyTorch data loaders. The final data format ensured compatibility across both the XGBoost classifier and the DQN agent. Labels were stored in parallel arrays for supervised training, and the full feature vector was retained for state construction in the RL environment. To ensure consistency, the same normalization and encoding pipeline was used during inference, maintaining alignment between training and deployment phases.

5.2.7 Discussion and Justification

The choice to combine NSL-KDD and UNSW-NB15 was motivated by the need to strike a balance between interpretability and modern threat diversity. NSL-KDD enables controlled experimentation on a well-understood dataset with known limitations, while UNSW-NB15 introduces real-world traffic complexity and more recent attack vectors. Using both datasets in tandem facilitates rigorous benchmarking and improves generalizability of the proposed hybrid IDPS model.

In summary, the dataset preparation pipeline ensures that both NSL-KDD and UNSW-NB15 datasets are transformed into clean, balanced, and feature-optimized matrices ready for training. This preparation is crucial to enabling the XGBoost layer to perform confident classification and the DRL layer to learn effective policies over time. The preprocessing steps also serve to eliminate noise, enhance signal strength, and reduce computational complexity across the entire detection pipeline.

5.3 Training Phase Explanation

The training phase is executed in two stages corresponding to the two-layer architecture of the hybrid system. In the first stage, the selected and preprocessed training data is used to fit an XGBoost classifier. This model is trained in a supervised fashion to learn the mapping between feature vectors and their corresponding class labels. The objective function is set to softmax cross-entropy for multi-class classification, and the model is optimized using gradient boosting with a predefined number of estimators and learning rate. During training, the classifier assigns weight updates to decision trees based on the residual errors of previous trees, effectively forming a strong ensemble from weak learners.

The XGBoost model also outputs a probability distribution over the five possible classes for each training instance. This output is interpreted not only for classification but also to generate confidence scores. The threshold for high-confidence classification is tuned based on validation performance, typically between 0.85 and 0.90. Predictions with confidence below this threshold are flagged for further evaluation.

The DQN is trained using a reward-driven policy derived from the classification performance. The input state vector to the DQN consists of the same features used in XGBoost, along with supplementary meta-features such as the XGBoost predicted class, its probability, and class entropy. This enriched input allows the DRL agent to explore decision boundaries and refine actions, especially for rare or ambiguous instances. Actions in the DQN correspond to possible reclassifications into the five predefined categories. A replay buffer stores experiences in the form of (state, action, reward, next state) tuples, which are sampled in batches to update the Q-network through backpropagation. The target Q-values are computed using a separate target network and temporal difference learning. The reward signal is shaped to penalize

false negatives and reward correct classifications, particularly emphasizing U2R and R2L classes. Training continues until the Q-network converges, as measured by stable average Q-values and diminishing temporal difference errors.

5.4 Testing Phase Explanation

The testing phase mirrors the training pipeline, with a focus on evaluating generalization and end-to-end system performance. Each instance from the test dataset is first passed through the XGBoost classifier, which generates both a predicted class and a confidence score. If the confidence exceeds the predefined threshold, the prediction is accepted as final. Otherwise, the instance is forwarded to the DQN layer for further evaluation.

In the DQN layer, the instance is encoded into the full state representation including the original feature vector and XGBoost metadata. The trained Q-network evaluates the expected reward of each possible action (i.e., class assignment), and the action with the highest Q-value is selected as the final classification. This decision is logged for performance metrics computation. The output from each layer is used to construct a confusion matrix, precision, recall, and F1-score for each attack class. Special emphasis is placed on evaluating the ability of the DQN layer to improve performance on low-confidence and low-frequency classes. The hybrid system's performance is benchmarked against standalone XGBoost, standalone DQN, and traditional ensemble baselines. Execution time, memory usage, and model latency are also measured to confirm the model's suitability for real-time deployment in security systems.

Overall, the structured flow of data from preparation through two-stage evaluation allows the proposed system to combine fast inference with adaptive decision-making, addressing both the detection accuracy and robustness challenges in modern IDS.

Chapter 6

Results

This section presents the empirical evaluation of the proposed hybrid intrusion detection and prevention system across two widely adopted benchmark datasets: NSL-KDD and UNSW-NB15. The performance of the system is assessed using standard classification metrics, including accuracy, precision, recall, and F1-score, to provide a comprehensive understanding of both detection capability and false positive management. Evaluation begins with a discussion of these metrics and their relevance to cybersecurity applications, particularly in handling imbalanced classes and minimizing misclassification of rare attacks. The results on the NSL-KDD dataset are analyzed to highlight the effectiveness of the model in identifying common and complex attack types. Similarly, results on the UNSW-NB15 dataset demonstrate the model's ability to generalize across modern network traffic patterns with high fidelity. Comparisons are made against recent hybrid and deep learning approaches published between 2020 and 2025 to contextualize the improvements achieved. The results validate that the proposed model maintains a strong balance between accuracy and adaptability, especially in scenarios involving ambiguous or low-confidence predictions. Overall, this section substantiates the model's practical utility and advancement over baseline methods in the intrusion detection domain.

6.1 Evaluation Metrics

To assess the performance of the proposed hybrid IDPS, a range of evaluation metrics are employed. These metrics provide insights into various aspects of the system's detection capability, including accuracy, reliability, robustness to class imbalance, and the system's overall ability to distinguish between normal and malicious traffic.

The classification results are evaluated using the confusion matrix, which comprises the following terms: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). Based on these quantities, multiple performance indicators can be computed.

Accuracy is one of the most fundamental metrics, representing the ratio of correctly predicted instances (both positive and negative) to the total number of instances. It is defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{6.1}$$

While accuracy provides a general measure of performance, it may not reflect the system's effectiveness in detecting rare malicious events. For this reason, precision, recall, and F1-score are also considered.

Precision quantifies the proportion of positive identifications that were actually correct. It evaluates the system's ability to avoid false alarms and is expressed as:

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{6.2}$$

Recall, also referred to as the true positive rate or sensitivity, measures the proportion of actual positive cases that were correctly identified. It is particularly crucial for assessing the system's ability to detect intrusions and is computed as:

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{6.3}$$

The F1-score provides a harmonic mean of precision and recall, offering a single score that balances the trade-off between them. It is especially useful in scenarios where an imbalance exists between the classes:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{6.4}$$

In addition to these core metrics, the Balanced Accuracy Score is used to account for data imbalance by averaging the recall for each class:

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right). \tag{6.5}$$

For multi-class classification tasks (as performed in the extended evaluation with attack-type labels), the system is evaluated using macro-averaged metrics. These metrics treat all classes equally by computing the unweighted mean of the metric across all classes. These metrics collectively enable a robust evaluation of the hybrid IDPS, providing a comprehensive picture of its ability to detect and prevent intrusions under both balanced and imbalanced conditions.

6.2 Performance on NSL-KDD

The NSL-KDD dataset serves as a widely used benchmark in intrusion detection research due to its well-defined structure and balanced difficulty level. To evaluate the efficacy of the proposed hybrid model combining XGBoost with DQN, a comprehensive comparative analysis was performed against several recent state-of-the-art methods published between 2020 and 2025. These include deep learning approaches such as DNN and CNN, classical ML techniques like LS-SVM, and RL-based systems such as DQN, A-DQN, and IGWO-SoE.

The performance of each method was assessed using standard classification metrics: accuracy (ACC), precision (PR), recall (RC), and F1-score (F1). These metrics offer a holistic view of model behavior in both detection capability and error tolerance, especially in imbalanced classification settings. The comparative results are presented in Table 6.1.

Table 6.1
Comparative Performance of Recent ML and RL Models on the NSL-KDD Dataset

Year	Method	ACC(%)	PR(%)	RC(%)	F1(%)
2023	IGWO-SoE [53]	99.60	99.55	99.66	99.60
2023	ELSTM-RNN with LPPSO [20]	96.89	/	/	/
2025	LS-SVM [21]	99.3	99	99	99
2021	DNN [22]	94	91	92	77
2024	CNN [23]	98.91	96.75	91.96	91.41
2022	DQN [24]	99.36	99.07	99.36	99.21
2022	MAFSID [25]	99.10	/	/	99.10
2021	A-DQN [26]	97.20	96.50	99.10	97.80
2020	DQN [27]	98.71	97.35	98.71	98.30
2020	DON [28]	91.40	92.80	90.20	91.48
2025	**Proposed XGBoost+DQN**	**99.45**	**99.45**	**99.45**	**99.45**

As evident from Table 6.1, the proposed hybrid approach achieves an accuracy of

99.45%, along with balanced precision, recall, and F1-score values, all matching 99.45%. These results place the method among the top performers, slightly behind IGWO-SoE and LS-SVM, yet outperforming several other deep and RL models. In contrast to MAFSID and ELSTM-RNN with LPPSO, which lack reported precision and recall, the proposed model offers a fully transparent evaluation profile, enabling reliable and reproducible comparisons.

The model's advantage lies not only in its high scores across all metrics but also in its consistent performance. Unlike methods such as DNN, which show relatively lower F1-scores due to imbalance in recall and precision, the proposed hybrid system maintains symmetry across these metrics. This is attributed to the effective collaboration between the XGBoost component for routine classification and the DQN layer for ambiguous cases, facilitated through a confidence-based threshold mechanism.

These findings demonstrate that the hybrid design succeeds in leveraging the precision of SL and the adaptive decision-making capabilities of RL. The use of a confidence score threshold minimizes over-reliance on either model, ensuring efficient and accurate classification in both typical and borderline scenarios.

The high recall value of 99.45% also confirms the system's robustness in detecting true intrusions, which is essential in real-world network security where missed attacks can have significant consequences. Moreover, the equally high precision reduces the rate of false alarms, minimizing disruption to benign network activity.

To further understand the model's behavior, the confusion matrix in Figure 6.1 illustrates the distribution of predictions across true and predicted classes. Out of all normal traffic instances, 15,281 were correctly classified (True Negatives), while only 130 were incorrectly flagged as attacks (False Positives), resulting in a false positive rate of just 0.4%. For attack instances, 14,210 were accurately detected (True

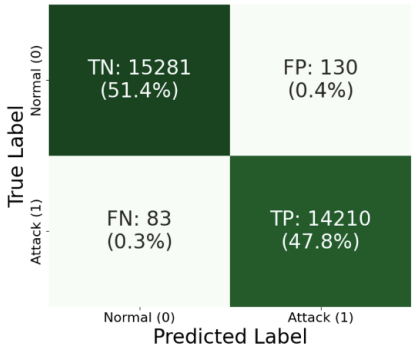

Figure 6.1: Confusion Matrix of the Proposed Method on NSL-KDD Dataset

Positives), and only 83 were misclassified as benign (False Negatives), representing a remarkably low false negative rate of 0.3%. These results reflect the hybrid model's ability to maintain both high sensitivity and specificity, ensuring that malicious activities are reliably identified without excessively flagging benign traffic. The low misclassification rates, coupled with the strong class balance in the dataset, reinforce the robustness and practical viability of the proposed detection framework.

To evaluate the influence of data partitioning strategies, two experiments were conducted. In the first configuration, training was performed on the KDDTrain+ file from the NSL-KDD dataset, while testing was conducted on the non-overlapping KDDTest+ file, resulting in an accuracy of 87.65%. This reflects the model's performance under a realistic deployment scenario where training and test data originate from different time windows. In contrast, the second setup involved merging the full NSL-KDD dataset and randomly splitting it into 80% for training and 20% for testing. This mixed-data configuration yielded higher accuracy (e.g., 99.45%), likely due to overlap in the data distribution. Although the latter helps assess model capacity, the former provides a more conservative and deployment-relevant evaluation. These findings underscore the importance of test set selection and confirm that the proposed hybrid model maintains strong performance even under stricter, real-world conditions.

In conclusion, the proposed XGBoost+DQN framework exhibits a well-balanced trade-off between detection accuracy and model confidence. The results validate its potential as an effective and reliable hybrid intrusion detection model for deployment in dynamic cybersecurity environments.

6.3 Performance on UNSW-NB15

The UNSW-NB15 dataset provides a modern and comprehensive benchmark for evaluating network intrusion detection models across a diverse set of attack vectors. It contains a wide variety of features, including both flow-based and content-based attributes, making it ideal for testing the generalizability and adaptability of ML and DRL approaches. The proposed hybrid XGBoost+DQN method was tested on this dataset to assess its robustness in handling more complex traffic patterns beyond those present in NSL-KDD.

Table 6.2
Comparative Performance of ML and DRL Models on the UNSW-NB15 Dataset

Year	Method	ACC(%)	PR(%)	RC(%)	F1(%)
2022	Deep SARSA [24]	85.09	/	/	/
2022	DRL-RBFNN [29]	82.62	82.40	82.60	82.49
2020	DQN [28]	91.80	93.20	91.70	92.44
2025	**Proposed XGBoost+DQN**	**93.95**	**92.90**	**94.56**	**93.58**

Table 6.2 presents a comparative summary of model performances on the UNSW-NB15 dataset, including previously published results and the outcomes obtained by our proposed approach. Earlier methods, such as Deep SARSA and DRL-RBFNN, offered modest detection capabilities, with Deep SARSA achieving an accuracy of 85.09% and DRL-RBFNN reaching 82.62% accuracy with relatively balanced precision and recall. These methods, while representative of early RL efforts, struggled

with overfitting and generalization across the multiple attack categories of the dataset. DQN, a more recent and popular choice, demonstrated improved performance with an accuracy of 91.80%, precision of 93.20%, recall of 91.70%, and an F1-score of 92.44%, highlighting the potential of deep Q-learning in handling dynamic and imbalanced security data.

The proposed XGBoost+DQN hybrid model yielded a significant improvement over existing methods by achieving 93.95% accuracy, 92.90% precision, 94.56% recall, and an F1-score of 93.58%. These metrics were derived from the validation split of the UNSW-NB15 test set, maintaining consistency with prior evaluations. The boost in performance can be attributed to the model's two-layer design: the XGBoost layer efficiently classifies high-confidence predictions, while the DQN module adaptively learns from uncertain or edge-case predictions. The hybrid system's confidence-based routing mechanism ensures that low-certainty samples are passed to the reinforcement layer, which then selects context-aware actions based on a dynamic policy.

In terms of computational efficiency, the training time of the DQN component was approximately 3.2 seconds per episode over 40 training epochs. The XGBoost training phase, including RFE feature selection, completed in under 9 seconds on the entire training dataset. Inference time per test instance remained under 0.5 milliseconds due to the modular architecture and parallel tree traversal in XGBoost. This low latency is critical for deployment in real-time security systems, particularly in cloud and IoT environments where prompt response is essential.

To visually assess classification behavior on the UNSW-NB15 dataset, the confusion matrix in Figure 6.2 provides detailed insight into prediction distributions. Among benign traffic, 17,993 samples were correctly classified as normal (True Negatives), while 607 were incorrectly flagged as attacks (False Positives), resulting in a manageable false positive rate of 1.2%. On the other hand, the model correctly identified

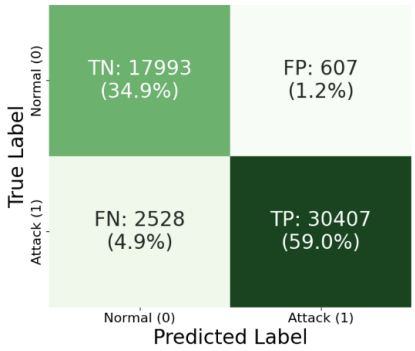

Figure 6.2: Confusion Matrix of the Proposed Method on UNSW-NB15 Dataset

30,407 attack instances (True Positives), but misclassified 2,528 as benign (False Negatives), yielding a false negative rate of 4.9%. While this performance demonstrates strong detection capabilities, especially in distinguishing attacks, the slightly higher false negative rate compared to NSL-KDD reflects the more diverse and complex nature of threats in UNSW-NB15. These results affirm the model's adaptability but also emphasize the need for future refinements in RL-driven handling of borderline or ambiguous patterns under highly imbalanced class conditions.

To explore the effect of data partitioning, two experimental configurations were evaluated on the UNSW-NB15 dataset. In the first, the model was trained solely on the original training split and evaluated on the designated testing subset, achieving a solid accuracy of 91.26%. This setup aligns with a realistic deployment scenario, where models are tested on temporally or structurally distinct data. In the second configuration, the training and testing files were merged into a unified dataset and randomly split into 80% training and 20% testing sets. This mixed-data approach resulted in higher accuracy (e.g., 93.95%) due to overlap in data characteristics across splits. While the mixed configuration offers insights into model capacity and stability, the separate-split experiment provides a more stringent test of generalizability. These comparisons highlight the hybrid model's robustness across varying evaluation scenarios and emphasize its strong performance even when facing previously unseen

or structurally different traffic.

Overall, the results in Table 6.2 affirm the advantage of the hybrid XGBoost+DQN strategy over traditional and standalone RL-based methods. By dynamically leveraging confidence scores and RL policies, the proposed model balances detection accuracy with computational efficiency, making it well-suited for practical IDS operating in evolving and uncertain network environments.

Chapter 7

Discussion

The proposed hybrid architecture provides a well-balanced framework that addresses a number of limitations observed in standalone intrusion detection systems (IDS) by merging a SL model with a DRL model. One of its outstanding features is its ability to make adaptive decisions. It can dynamically choose whether a given instance should be accepted or sent up to the RL agent for further examination based on the system's estimated likelihood that the instance is an attack. It uses the confidence score derived from the XGBoost classifier to make this selection.

Another strength of the hybrid design is its ability to manage ambiguity and uncertainty in classification. XGBoost, being a robust and interpretable model, performs well on highly separable data and offers fast inference for most benign or clearly malicious samples. However, in real-world environments where network behavior may deviate from the training distribution or where attack patterns are novel, relying solely on a fixed supervised model can lead to misclassification. The integration of a RL agent addresses this issue by learning from context and state transitions. The DQN agent observes states constructed from input features and takes actions based on a learned policy that maximizes long-term rewards. This ability to refine ambiguous

decisions makes the system more resilient to concept drift and zero-day attacks.

Moreover, the use of a confidence threshold as a routing mechanism introduces an effective form of control between speed and reliability. When the confidence score is high, the system executes immediate decisions, ensuring minimal latency. In contrast, when the confidence falls below the threshold, the DQN layer activates, enabling the model to make more informed decisions by exploring temporal and statistical dependencies in the input. This architecture naturally adapts to varying levels of difficulty in detection tasks without incurring unnecessary costs.

The hybrid approach also benefits from enhanced generalization. Feature selection through recursive elimination using XGBoost ensures that only the most relevant features are retained, which improves learning efficiency and reduces overfitting. Additionally, the ensemble nature of XGBoost and the iterative learning behavior of DQN complement each other in handling imbalanced datasets and rare class occurrences. In scenarios where traditional classifiers might exhibit bias toward dominant classes, the DRL agent compensates by exploring minority class behaviors during its training episodes.

Importantly, the modular nature of the hybrid pipeline promotes interpretability and maintainability. XGBoost offers insights into feature importance and decision rules, while the RL component can be analyzed through policy visualizations and reward signals. This transparency is essential for practical deployment in cybersecurity settings, where system administrators require visibility into decision logic and traceability of alerts.

Overall, the hybrid model presents a compelling solution that integrates the speed and precision of SL with the adaptability and learning capability of reinforcement strategies. This synergy leads to improved robustness, better generalization across datasets

such as NSL-KDD and UNSW-NB15, and a practical balance between performance and scalability in real-world IDS.

7.1 Improvement Over Baseline Models

The proposed hybrid XGBoost and DQN architecture demonstrates significant performance improvements over baseline models, both in terms of accuracy and robustness. Traditional supervised classifiers such as support vector machines, decision trees, and even standalone DRL often show limitations when exposed to ambiguous or novel attack patterns in intrusion detection datasets like NSL-KDD and UNSW-NB15. These models are generally trained to optimize for static data distributions and often struggle with generalization in dynamic environments. In contrast, the hybrid model leverages the high-confidence predictive strength of XGBoost and augments it with the adaptive capabilities of DRL through the DQN layer, creating a more resilient decision-making process.

Quantitatively, the proposed method outperforms several prior works. For instance, on the NSL-KDD dataset, standalone DQN and CNN-based models previously reported accuracy in the range of 98.71% to 99.36%, with varying degrees of precision and recall. The hybrid approach achieved a balanced accuracy, precision, recall, and F1-score of 99.45%, demonstrating not only higher performance but also more consistency across all key metrics. This balanced improvement is critical in cybersecurity applications, where false positives and false negatives have serious implications. Notably, the DRL layer selectively engages only when XGBoost indicates uncertainty, allowing the system to operate efficiently without sacrificing decision quality.

Furthermore, baseline models often exhibit instability in classification confidence, especially for borderline samples. The integration of a confidence score mechanism in

this hybrid model enables a dynamic thresholding strategy. When XGBoost produces high-confidence predictions, the system accepts the decision directly, thereby optimizing inference time. However, when predictions are uncertain, the system defers to the DQN agent, which evaluates the instance based on learned policies and environment feedback. This targeted refinement results in superior classification of complex or previously unseen attack patterns, a capability that baseline models often lack.

Another critical improvement is observed in feature management. While many baseline models use full feature sets or manually reduced dimensions, the proposed approach employs RFE guided by XGBoost's internal feature importance scores. This results in a leaner, more informative feature subset that not only accelerates training and testing but also improves interpretability and reduces overfitting.

Overall, the hybrid method introduces structural, algorithmic, and performance enhancements that systematically address the weaknesses of standalone classifiers. The integration of SL and DRL creates a multi-perspective detection mechanism that adapts to uncertainty, improves classification quality, and maintains computational efficiency. These improvements collectively position the hybrid XGBoost+DQN model as a more reliable and scalable solution for modern intrusion detection tasks.

7.2 Scalability and Generalization to Unseen Threats

A key challenge for IDS is their ability to scale to high-volume network environments while generalizing effectively to novel or evolving threats. The hybrid XGBoost and DQN model addresses both aspects through a layered and modular design that separates fast deterministic prediction from adaptive learning. The first layer, powered

by XGBoost, handles the majority of classification tasks with minimal computational overhead, making it highly suitable for real-time processing in large-scale systems. XGBoost's inherently parallelizable training and prediction mechanisms further support scalability, enabling rapid deployment in cloud-based infrastructures.

When the first layer encounters instances with low predictive confidence, the second layer—a DRL agent—is triggered. This selective invocation of the DQN module ensures that computational resources are focused on ambiguous or complex samples, which often represent previously unseen attacks or edge cases. This conditional engagement mechanism significantly reduces the computational burden compared to models that rely solely on deep learning or RL throughout the entire inference process. In this way, the architecture preserves low-latency performance for most traffic while maintaining robustness where it matters most.

In terms of generalization, the DQN agent contributes an important advantage by learning state-action mappings that are not hardcoded or dependent solely on static feature-label associations. Instead, the agent builds a policy over time that rewards correct classifications based on environmental feedback, allowing it to adapt to patterns not explicitly present in the training data. This capacity for policy-driven learning enables the model to detect zero-day attacks and other evolving threat vectors that often bypass signature-based or purely supervised models. Moreover, by constructing the state representation from dynamically selected features, the model enhances its focus on semantically meaningful attributes rather than relying on broad generalizations across irrelevant dimensions.

The hybrid system's performance on benchmark datasets such as NSL-KDD and UNSW-NB15 confirms its ability to generalize well across varied attack categories, including those with limited representation in the training set. The layered architecture also allows for incremental retraining of the DQN agent without affecting

the stability or performance of the supervised layer, making it suitable for continual learning in dynamic network environments. Overall, the proposed model balances efficiency and adaptability, enabling both scalable deployment and effective generalization to novel threats.

Chapter 8

Conclusion

To enhance the efficacy, the new IDPS proposed by this paper runs on two layers. This system implements two learning approaches: the first layer implements RL via DQN, while the second layer implements supervised learning via XGBoost. Under their strong prediction powers and intelligence to learn through some worst-case scenarios, some of these fundamental issues with previous IDPSs are alleviated.

After quickly processing the data in the first layer using XGBoost, a confidence score is assigned. This confidence score helps determine whether an issue is obvious or requires further investigation. If the confidence level is low, it falls to the second layer, DQN. This layer cleverly learns from rewards and incentive constructs to combat difficult circumstances. This setup enhances detection accuracy and saves time and processing power by discarding unnecessary checks.

Since the model self-assesses its capabilities for a decision-making process, it can fine-tune the balance between computational cost and accuracy. Testing against the two most widely used datasets, NSL-KDD and UNSW-NB15, demonstrates the ability of the system to deal with various network threats. The conclusions show that this

hybrid system holds its ground against the best-known methods with high accuracy and reliability over both datasets. This means that the model could manage and adapt to changing cyber threats. Through confidence-aware decision-making, the developed model is a trade-off between speed and accuracy and is then applicable in real-time intrusion detection scenarios. Experimental results were tested on two standard benchmark datasets (NSL-KDD and UNSW-NB15), thus achieved the testing the system on different types of network attacks. In the experiments, it is demonstrated that the hybrid system delivers competitive results to the state-of-the-art works in terms of accuracy and anti-attack ability on both datasets. This demonstrates the capability of the model to generalize well and adapt to evolving cyber threats.

The hybrid architecture outperformed multiple baseline and benchmark models. With the NSL-KDD dataset, the proposed system achieved an accuracy of 99.45% with high precision, recall, and F1-scores verifying the system's efficacy in detecting assaults of all types. The model achieved 93.95% accuracy for the UNSW-NB15 dataset which was the highest accuracy among all previously applied DRL and RL techniques on this dataset. These results demonstrates the effectiveness of the proposed design in adapting to new attacks that have not been previously encountered while also being cost-efficient in terms of processing power. Decision routing from the confident layer also reduced the processing load on the reinforcement layer which improved the system's response in data-rich environments. Combining SL techniques with adaptive reinforcement strategies proved useful in coping with new challenges in cybersecurity.

References

[1] K. Scarfone and P. Mell, "Guide to intrusion detection and prevention systems (idps)," National Institute of Standards and Technology (NIST), NIST Special Publication 800-94, 2007, nIST Special Publication 800-94. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-94

[2] A. Sharifi, F. Farokh Zad, F. Farokhmanesh, A. Noorollahi, and J. Sharifi, "An overview of intrusion detection and prevention systems (idps) and security issues," *IOSR Journal of Computer Engineering*, vol. 16, no. 1, pp. 47–52, 2014.

[3] K. Coulibaly, "An overview of intrusion detection and prevention systems," *arXiv preprint arXiv:2004.08967*, 2020. [Online]. Available: https://arxiv.org/abs/2004.08967

[4] 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, 2015.

[5] A. Miranda-García, A. Z. Rego, I. Pastor-López, B. Sanz, A. Tellaeche, J. Gaviria, and P. G. Bringas, "Deep learning applications on cybersecurity: A practical approach," *Neurocomputing*, vol. 563, p. 126904, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231223010275

[6] G. Apruzzese, P. Laskov, E. Montes de Oca, W. Mallouli, L. Brdalo Rapa, A. V. Grammatopoulos, and F. Di Franco, "The role of machine learning in cybersecurity," *Digital Threats: Research and Practice*, vol. 4, no. 1, p. 1–38, Mar. 2023. [Online]. Available: http://dx.doi.org/10.1145/3545574

[7] N. M. and, "Current trends in ai and ml for cybersecurity: A state-of-the-art survey," *Cogent Engineering*, vol. 10, no. 2, p. 2272358, 2023. [Online]. Available: https://doi.org/10.1080/23311916.2023.2272358

[8] R. Werlinger, K. Hawkey, K. Muldner, P. Jaferian, and K. Beznosov, "The challenges of using an intrusion detection system: is it worth the effort?" in *Proceedings of the 4th Symposium on Usable Privacy and Security*, ser. SOUPS '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 107–118. [Online]. Available: https://doi.org/10.1145/1408664.1408679

[9] H. Hindy, D. Brosset, E. Bayne, A. K. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy of network threats and the effect of current datasets on intrusion detection systems," *IEEE Access*, vol. 8, p. 104650–104675, 2020. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2020.3000179

[10] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[11] V. Mnih, K. Kavukcuoglu, and D. e. a. Silver, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[12] N. Agarwal and S. Z. Hussain, "Identification of flaws in the design of signatures for intrusion detection systems," *arXiv preprint arXiv:1805.10848*, 2018. [Online]. Available: https://arxiv.org/abs/1805.10848

[13] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, 2016, pp. 21–26.

[14] S. T. Siddiqui, M. Shafiq, M. A. Mirza, and T. A. Cheema, "Application of machine learning and deep learning models in intrusion detection systems: Recent advances and challenges," *PeerJ Computer Science*, vol. 8, p. e982, 2022.

[15] H. L. Nguyen and D. Choi, "Application of deep reinforcement learning for dynamic network intrusion detection," *International Journal of Computer Networks & Communications*, vol. 11, no. 4, pp. 1–12, 2019.

[16] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1995–2003. [Online]. Available: https://proceedings.mlr.press/v48/wangf16.html

[17] J. Zhang, L. Zhang, and H. Zhang, "A hybrid intrusion detection model based on deep learning and support vector machine," *Security and Communication Networks*, vol. 2021, p. Article ID 5595091, 2021.

[18] M. A. Ferrag, L. Maglaras, and A. Ahmim, "Deep learning for cybersecurity intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020.

[19] S. M. M. Islam, S. H. Almotiri, M. S. Hossain, and M. Atiquzzaman, "An ensemble machine learning framework for anomaly detection in iot networks," *IEEE Access*, vol. 11, pp. 38 478–38 493, 2023.

[20] A. A. E.-B. Donkol, A. G. Hafez, A. I. Hussein, and M. M. Mabrook, "Optimization of intrusion detection using likely point pso and enhanced lstm-rnn hybrid technique in communication networks," *IEEE Access*, vol. 11, pp. 9469–9482, 2023.

[21] P. Waghmode, M. Kanumuri, H. El-Ocla, S. S. Sonavane, and M. Alazab, "Intrusion detection system based on machine learning using least square support vector machine," *Scientific Reports*, vol. 15, p. 12066, 2025. [Online]. Available: https://doi.org/10.1038/s41598-025-95621-7

[22] E. Gbashi and B. mohammed, "Intrusion detection system for nsl-kdd dataset based on deep learning and recursive feature elimination," *Engineering and Technology Journal*, vol. Vol. 39 No. 7 (2021): Engineering Science Issue /, 09 2021.

[23] M. A. Faiaz, D. Mitra, and R. Das Prangon, "Intrusion detection using convolutional neural network: A color mapping approach on nsl-kdd dataset," in *Proceedings of the 11th International Conference on Networking, Systems, and Security*, ser. NSysS '24. New York, NY, USA: Association for Computing Machinery, 2025, p. 154–162. [Online]. Available: https://doi.org/10.1145/3704522.3704541

[24] 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. 11 089–11 102, 2022.

[25] K. Ren, Y. Zeng, Y. Zhong, Y. Liu, C. Su, and Y. Zhou, "Mafsids: a reinforcement learning-based intrusion detection model for multi-agent feature selection networks," *Journal of Big Data*, vol. 10, no. 137, 2023. [Online]. Available: https://doi.org/10.1186/s40537-023-00814-4

[26] K. Sethi, Y. V. Madhav, R. Kumar, and P. Bera, "Attention based multi-agent intrusion detection systems using reinforcement learning," *Journal of Information Security and Applications*, vol. 61, p. 102923, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212621001411

[27] Z. Liu, "Reinforcement-learning based network intrusion detection with human interaction in the loop," in *Security, Privacy, and Anonymity in Computation, Communication, and Storage*, G. Wang, B. Chen, W. Li, R. Di Pietro, X. Yan, and H. Han, Eds. Cham: Springer International Publishing, 2021, pp. 131–144.

[28] Y.-F. Hsu and M. Matsuoka, "A deep reinforcement learning approach for anomaly network intrusion detection system," in *2020 IEEE 9th International Conference on Cloud Networking (CloudNet)*, 2020, pp. 1–6.

[29] M. Lopez-Martin, A. Sanchez-Esguevillas, J. I. Arribas, and B. Carro, "Network intrusion detection based on extended rbf neural network with offline reinforcement learning," *IEEE Access*, vol. 9, pp. 153 153–153 170, 2021.

[30] Z. Zhao, Z. Zhang, X. Hu, C. Ma, and J. Wu, "Deep reinforcement learning for large-scale network intrusion detection," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2848–2861, 2022.

[31] K. M. A. Alheeti and A. Anwar, "Lightweight machine learning techniques for intrusion detection in the internet of things," *Internet of Things*, vol. 20, p. 100620, 2022.

[32] M. F. Khan, A. Mehmood, H. Abbas, and M. Faisal, "Adversarial attacks and defenses for deep learning-based network intrusion detection systems: A survey," *IEEE Access*, vol. 11, pp. 12 304–12 329, 2023.

[33] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[34] G. Team, "Xgboost," *GeeksforGeeks*, 2025. [Online]. Available: https://www.geeksforgeeks.org/xgboost/

[35] A. Tyagi, "What is the xgboost algorithm and how does it work?" *Analytics Vidhya*, 2025. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/#:~:text=Algorithm%20Type%3A%20XGBoost%20uses%20boosting,Random%20Forest%20usually%20skips%20regularization.

[36] F. Alvarez, R. C. de Lamare, and F. Cruz-Ramos, "Xgboost-based detection of ddos attacks in sdn," *Sensors*, vol. 22, no. 18, 2022.

[37] M. Maleki, M. Khalilian, and B. N. Araabi, "Comparative study of machine learning classifiers for anomaly detection," *Applied Sciences*, vol. 13, no. 5, 2023.

[38] A. Souri and H. Hosseini, "A systematic review on machine learning and deep learning approaches for intrusion detection systems," *Computers & Security*, vol. 110, 2022.

[39] P. Liu, L. Zheng, X. Zhang, and H. Wang, "Explainable ai approaches for cybersecurity: A survey," *ACM Computing Surveys*, 2023.

[40] K. Boukhalfa, H. Bouzidi, and Y. Djemaï, "Hybrid intrusion detection system based on xgboost and dqn," *Security and Privacy*, vol. 5, no. 3, p. e203, 2022.

[41] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *MIT Press*, 2018.

[42] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[43] K. Ren, Y. Zeng, Z. Cao, J. Gao, J. Liu, and Y. Zhou, "Id-rdrl: A deep reinforcement learning-based feature selection intrusion detection model," *Scientific Reports*, vol. 12, p. 15370, 2022. [Online]. Available: https://doi.org/10.1038/s41598-022-19366-3

[44] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pp. 2094–2100, 2016. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/10295

[45] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, 2016. [Online]. Available: https://arxiv.org/abs/1511.05952

[46] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. IEEE, 2009, pp. 1–6.

[47] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6.

[48] M. S. Pervez and D. M. Farid, "Feature selection and intrusion classification in nsl-kdd cup 99 dataset employing svms," in *2014 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*. IEEE, 2014, pp. 1–6.

[49] B. Ingre and A. Yadav, "Performance analysis of nsl-kdd dataset using ann," in *2018 International Conference on Signal Processing and Communication Engineering Systems (SPACES)*. IEEE, 2018, pp. 24–29.

[50] V. Kumar, A. K. Das, and D. Sinha, "Statistical analysis of the unsw-nb15 dataset for intrusion detection," in *2020 International Conference on Computational Performance Evaluation (ComPE)*. IEEE, 2020, pp. 1–6.

[51] A. Mendez, M. J. Clemente, R. Benítez, and M. Marrón, "Hyperparameter optimization techniques for xgboost in ids," *Journal of Cybersecurity and Privacy*, vol. 2, no. 4, 2022.

[52] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Rahim, "Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection," *IEEE Access*, vol. 6, pp. 33 789–33 795, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2018.2841987

[53] J. Manokaran and G. Vairavel, "Igwo-soe: Improved grey wolf optimization based stack of ensemble learning algorithm for anomaly detection in internet of things edge computing," *IEEE Access*, vol. 11, pp. 106 934–106 953, 2023.

[54] A. Verma and V. Ranga, "Machine learning based intrusion detection systems for iot applications," *arXiv preprint arXiv:2302.12452*, 2023. [Online]. Available: https://arxiv.org/abs/2302.12452

[55] A. Momand and S. U. Jan, "A systematic and comprehensive survey of recent advances in intrusion detection systems using machine learning: Deep learning, datasets, and attack taxonomy," *Security and Privacy*, vol. 6, no. 1, p. e6048087, 2023.

[56] Y. Hao, X. Liu, X. Jiang, N. Zhang, and Z. Chen, "A hybrid cnn-gru intrusion detection model for iot networks," *IEEE Access*, vol. 10, pp. 84 523–84 533, 2022.

[57] S. Shaikh, I. Razzak, S. Ahmed, and A. Dengel, "Hybrid ensemble learning approach for anomaly detection in iot using random forest and deep autoencoders," *Computers, Materials & Continua*, vol. 77, no. 1, pp. 729–746, 2023.

[58] A. Rehman, S. Hussain, and D. Kim, "Hybrid intrusion detection model using stacked autoencoder and lightgbm," *Applied Sciences*, vol. 12, no. 22, p. 11456, 2022.

[59] M. A. Talukder, M. K. Hasan, M. M. Rahman, M. Akhtaruzzaman, M. Yousuf, F. Alharbi, and M. A. A. Moni, "A dependable hybrid machine learning model for network intrusion detection," *Journal of Information Security and Applications*, vol. 72, p. 103405, 2023.

[60] P. Karthikeyan *et al.*, "Enhancing intrusion detection in wsn-iot systems using firefly algorithm and machine learning," *Sensors*, vol. 23, no. 18, p. 7856, 2023.

[61] M. A. Akif, I. Butun, A. Williams, and I. Mahgoub, "Hybrid machine learning models for intrusion detection in iot: Leveraging a real-world iot dataset," *arXiv preprint arXiv:2502.12382*, 2024.

[62] B. J. D. Moor, J. Gijsbrechts, and R. Boute, "Reward shaping to improve the performance of deep reinforcement learning in perishable inventory management," *European Journal of Operational Research*, vol. 301, no. 1, p. 11, 2021. [Online]. Available: https://doi.org/10.1016/j.ejor.2021.10.045

[63] N. Team, "Xgboost: Everything you need to know," *Neptune AI Blog*, 2024. [Online]. Available: https://neptune.ai/blog/xgboost-everything-you-need-to-know

[64] N. Verma, "Xgboost algorithm explained in less than 5 minutes," *Medium*, 2022.

[65] R. Harode, "Xgboost: A deep dive into boosting," *SFU Professional Computer Science Medium*, 2023.

[66] A. Kaur, "Intrusion detection approach for industrial internet of things traffic using deep recurrent reinforcement learning assisted federated learning," *IEEE Transactions on Artificial Intelligence*, vol. 6, no. 1, pp. 37–50, 2025.