Reinforcement Learning for Intrusion Detection: More Model Longness and Fewer Update

Jennifa J[1], Rajesh R[2], Rajkumar U[3],Ponmoorthi M[4] , Saran M[5]

[1]Assistant Professor -Department of Information Technology & Kings Engineering College-India.

[2,3,4,5] Department of Information Technology & Kings Engineering College-India.

---***---

Abstract - In today's dynamic cybersecurity landscape, Intrusion Detection Systems (IDS) must adapt to evolving threats without relying on frequent retraining or compromising performance. This project proposes a novel IDS framework that synergistically combines Convolutional Neural Networks (CNNs) with Reinforcement Learning (RL) to classify network traffic as normal or anomalous, while minimizing the frequency of model updates. The framework is deployed via an intuitive, Streamlit-based web interface that supports real-time predictions based on user-provided network feature inputs.A pre-trained scaler is employed to normalize the input features before classification by the CNN model. Simultaneously, a reinforcement learning agent dynamically adjusts detection policies through reward-based feedback, thereby prolonging the model's operational lifespan and adaptability. To promote transparency and user trust, the system integrates LIME (Local Interpretable Model-Agnostic Explanations), providing interpretable, feature-level insights for each classification decision.

Experimental evaluation reveals that the proposed system achieves high classification accuracy, maintains a low false positive rate, and demonstrates strong resilience over time—without the need for frequent retraining. This work delivers a scalable, explainable, and low-maintenance solution for intrusion detection, offering a robust asset for contemporary cybersecurity environments.

1. INTRODUCTION

1.1 Introduction to Intrusion Detection Systems

In the current digital era, where information technology is deeply integrated into the functioning of governments, businesses, and individuals, the security of digital assets has become a primary concern. With the ever-growing complexity of computer networks and the exponential rise in cyber threats, it is imperative to deploy intelligent systems that can monitor, detect, and prevent malicious activities. One such critical component in the cybersecurity ecosystem is the Intrusion Detection System (IDS).

An IDS is a security mechanism that continuously monitors network traffic or system activities to detect suspicious behavior or unauthorized access. IDS tools are typically categorized into two main types: Host-Based IDS (HIDS) and Network-Based IDS (NIDS). HIDS monitors activities on individual systems such as servers or endpoints, whereas NIDS inspects data packets flowing through the network to identify potential threats.

The primary function of IDS is to raise alerts upon identifying deviations from normal behavior or signatures of known attacks. These systems provide crucial support to network administrators in mitigating threats at early stages and preventing large-scale data breaches. Despite their usefulness, traditional IDS systems face several limitations, including high false positive rates, difficulty in adapting to new attack strategies, and limited scalability in large, dynamic environments.

1.2 Role of Deep Learning in Network Security

With the advancement in artificial intelligence (AI), particularly in deep learning (DL), there has been a paradigm shift in how network threats are detected and managed. Deep learning techniques mimic the human brain's neural structure to learn complex data patterns and representations. These models, when trained on labeled network traffic data, can efficiently classify benign and malicious behavior without extensive manual feature engineering.

Several architectures have been successfully applied to cybersecurity tasks. Convolutional Neural Networks (CNN) are adept at capturing spatial hierarchies and local correlations in data, making them suitable for intrusion detection where packet features exhibit spatial patterns. Similarly, Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks excel in learning from sequential data and are commonly employed in detecting temporal anomalies in network traffic.

Deep learning models offer several advantages:

- ➢ **Automated Feature Extraction**: They eliminate the need for manual selection of features.
- ➢ **Scalability**: Capable of handling large-scale network traffic data.
- ➢ **Generalization**: They can detect sophisticated and previously unseen attack patterns.

However, deep learning models also pose certain challenges. They require large amounts of training data, significant computational resources, and suffer from the problem of model drift, wherein their accuracy degrades over time due to evolving threat landscapes. Regular retraining is often necessary, which is not always feasible in production environments. This shortcoming necessitates the exploration of adaptive models that can learn continuously and remain effective over time.

1.3 Reinforcement Learning: An Overview

Reinforcement Learning (RL) is a powerful machine learning paradigm inspired by behavioral psychology. It focuses on how an agent can learn to make decisions through interactions with an environment by receiving feedback in the form of rewards or penalties. RL is uniquely suited for problems that involve sequential decision-making, long-term strategy, and continuous adaptation—making it highly applicable to the dynamic and adversarial nature of cybersecurity.

The fundamental components of an RL system include:

- **Agent**: The learner or decision-maker (e.g., IDS model).
- **Environment**: The network system being monitored.
- **State**: The current representation of the environment.
- **Action**: The decision made by the agent (e.g., classify traffic as normal or anomalous).
- **Reward**: Feedback signal guiding learning (positive for correct classifications, negative for false alarms).

In the IDS context, reinforcement learning allows the model to dynamically adjust its behavior based on the feedback received from its predictions. Unlike supervised learning, RL does not require extensive labeled datasets; it can learn in real time by interacting with live data streams. This makes RL particularly advantageous in detecting zero-day attacks, adjusting to concept drift, and operating in resource-constrained environments.

Several RL variants, such as Q-Learning, SARSA, and Deep Q-Networks (DQN), have been explored for cybersecurity applications. These models not only enhance the detection capabilities but also support policy-based learning, enabling the IDS to evolve without extensive human supervision.

1.4 Motivation for Long-Lived Models and Minimal Updates

A persistent challenge in deploying IDS solutions in production environments is maintaining their performance over time. The threat landscape in cybersecurity is not static; new attack vectors are introduced regularly, and adversaries continuously evolve their methods to evade detection. As a result, models trained on static datasets tend to lose relevance over time, leading to degraded performance and increased false positives.

Most existing systems deal with this issue by retraining models frequently, often requiring:

- Fresh data collection
- Manual relabeling
- Model retraining and redeployment
- Downtime during update cycles

This is resource-intensive, time-consuming, and introduces risk during the update process. Furthermore, frequent updates can be impractical in edge computing scenarios, low-latency systems, or large-scale enterprise deployments.

This project proposes an alternative approach — building long-lived models that maintain their efficacy over time with minimal updates. By integrating reinforcement learning principles with deep learning architectures, the system can adapt its detection strategies over time without relying on continuous retraining. This goal is further reinforced through techniques such as:

- **Experience Replay**: Reusing past learning to avoid forgetting
- **Fixed Target Networks**: Improving stability in learning
- **Delayed Policy Updates**: Reducing overfitting to recent patterns

The motivation is to maximize model lifespan, reduce maintenance overhead, and enhance deployability across varied environments.

1.5 Problem Statement

The current generation of Intrusion Detection Systems often falls short in delivering long-term, adaptive protection without substantial manual intervention. Deep learning models, while powerful, become outdated quickly in the face of constantly evolving cyber threats. Frequent retraining introduces operational overhead, requires labeled data, and disrupts system availability.

Hence, the problem addressed in this project is:

“How can we design a reinforcement learning-enhanced intrusion detection system that sustains high performance with minimal updates, adapts to evolving threats, and provides explainable, real-time predictions through an accessible web-based interface?”

This problem statement encapsulates the need for longevity, adaptability, transparency, and usability — all essential for a next-generation IDS.

1.6 Objectives of the Study

The key objectives of this research and implementation are as follows:

Design a hybrid IDS framework combining deep learning with reinforcement learning to detect network intrusions effectively.

Implement mechanisms to reduce the frequency of retraining while maintaining high prediction accuracy and generalization to unseen attacks.

Develop a web-based user interface using Streamlit to enable real-time data entry, prediction visualization, and ease of access for cybersecurity analysts.

Incorporate LIME-based explainability to enhance model transparency and aid decision-making by highlighting feature contributions to classification outcomes.

Ensure secure user interaction through authentication and session control to prevent unauthorized access and maintain data integrity.

Evaluate system performance on benchmark datasets using metrics such as accuracy, precision, recall, F1-score, and model update frequency.By achieving these objectives, the proposed system aims to demonstrate not only technical feasibility but also practical relevance in real-world cybersecurity deployments.

2.SYSTEM ANALYSIS

The purpose of system analysis is to study the existing methodologies and evaluate the proposed solution in terms of requirements, use cases, and feasibility. A well-structured system analysis ensures that the proposed Intrusion Detection System (IDS) is technically sound, practically viable, and aligns with the defined objectives of achieving model longevity and reduced update frequency. This chapter comprehensively outlines the limitations of the existing systems and justifies the need for a reinforcement learning-based intelligent IDS with a real-time interface and explainability.

2.1 Existing System and Limitations

Existing Intrusion Detection Systems predominantly use either signature-based or anomaly-based detection mechanisms. While signature-based IDS such as Snort and Suricata provide fast and accurate detection for known threats, they fail miserably in detecting new or modified attack patterns. Anomaly-based systems offer better coverage but are prone to high false positive rates.

Moreover, most traditional systems do not adapt over time, and thus, require manual intervention for:

Updating the threat signature database,

Reconfiguring thresholds or detection parameters,

Re-training models using freshly labeled data.

- **Limitations of Existing IDS Approaches**:
- **Lack of adaptability**: Systems do not evolve with changing threat behaviors.
- **Performance decay**: Deep learning models lose accuracy over time (model drift).
- **High maintenance cost**: Requires frequent retraining and human oversight.
- **Poor interpretability**: Black-box predictions without explainable reasoning.
- **No real-time feedback loop**: Most systems are offline and not interactive.

Due to these limitations, the need arises for an intelligent, adaptive, and low-maintenance IDS that integrates deep learning with reinforcement learning while also being interpretable and user-friendly.

2.2 Proposed System Overview

The proposed system is a web-based Intrusion Detection System powered by Reinforcement Learning (RL) and Convolutional Neural Networks (CNNs). It is designed to classify network traffic as either “Normal” or “Anomalous” based on manually input features. The system reduces model decay and update frequency using experience replay and policy delay mechanisms, resulting in longer model life with minimal intervention.

Key Components:

- **Streamlit-based UI**: A real-time, interactive web interface for analysts.
- **CNN Model**: For initial deep feature extraction and classification.
- **RL Engine**: Enhances adaptability, learns from environment via feedback.
- **LIME Explainability Module**: Visualizes the reasoning behind predictions.
- **Authentication Module**: Secure login for controlled access.

The model aims to deliver high detection performance, maintain stability over time, and reduce operational overhead, while providing transparency to the end-user.

3.SYSTEM DESIGN

System design is a critical phase in the development lifecycle of any software application, especially for systems operating in mission-critical domains like cybersecurity. In this chapter, we present a comprehensive, multi-layered design blueprint for our proposed Intrusion Detection System (IDS) powered by deep learning and reinforcement learning (RL). The design ensures a modular, scalable, secure, and explainable platform that not only classifies network traffic but also adapts intelligently to evolving threats—all while minimizing the frequency of model updates.

The design is structured across multiple dimensions: architecture-level planning, data flow modeling, component separation, learning workflow, optimization strategies for reinforcement learning, and explainability through LIME.

3.1 System Architecture

The system architecture integrates several logical modules into a cohesive framework that ensures separation of concerns and robust interoperability. The architectural model adopted is a layered**, service-oriented structure** that comprises:

- **User Interaction Layer**: Web-based interface for input and output.
- **Data Preprocessing Layer**: Feature scaling and transformation.

- **Model Inference Layer**: CNN and RL integration for predictions.
- **Explainability Layer**: LIME engine to explain model behavior.
- **Security & Session Management Layer**: User authentication and secure access.

Key Components:

Input Module: Accepts network feature values manually input via a Streamlit dashboard. The system can be later extended to support real-time traffic from pcap files or packet sniffers.

Preprocessing Engine: Applies standard scaling, encoding of categorical variables, and reshaping for model compatibility. Pre-trained scalers are stored using joblib to ensure consistency.

Deep Learning Classifier: A CNN-based binary classifier trained on IDS datasets (e.g., NSL-KDD, CIC-IDS2017) to differentiate between normal and anomalous behaviors.

Reinforcement Learning Agent: Implements a Deep Q-Network (DQN) that takes state features, predicts an action, and adjusts policies using a reward feedback loop.

Model Persistence: All trained models and scalers are stored in serialized format for reuse and fast deployment. No retraining is done during inference.
Explainability Interface: Incorporates LIME to provide a per-instance explanation for each prediction made by the CNN model, enabling analysts to visually understand the decision-making process.
User Authentication & Access Control: Implements secure login/logout mechanisms to prevent unauthorized access and manage user sessions.

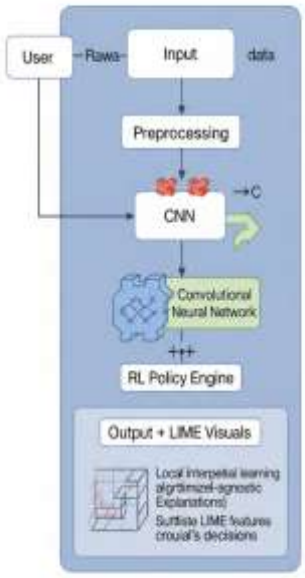

User
Rawa
Input
data
Preprocessing
→C
CNN
Convolutional Neural Network
RL Policy Engine
Output + LIME Visuals
Local interpetal learning algrtimizel-agnostic Explanations)
Suttiste LIME features crouial's decisions

Diagram: Figure 3.1 – Overall System Architecture

3.2 Data Flow Diagram (DFD)

Data Flow Diagrams (DFDs) are crucial for modeling how data moves through the system. They outline the transformation of data from raw input to final output.

DFD Level 0: Context Diagram

This shows the high-level interaction between external users (analysts or admins) and the system.

- **External Entity**: Security Analyst (User)
- **System**: RL-based IDS Application
- **Data Flows**: Feature values (input) → Prediction + Explanation (output)

DFD Level 1: Functional Breakdown

Processes:

- **P1** – Accept Input Features from the Analyst
- **P2** – Normalize and Preprocess Inputs
- **P3** – Perform Model Inference (CNN + RL)
- **P4** – Generate LIME Explanation
- **P5** – Display Results & Logs

Data Stores:

D1 – Model Weights (CNN/RL)

D2 – Scaler and Encoders
D3 – Audit Logs and Session Records

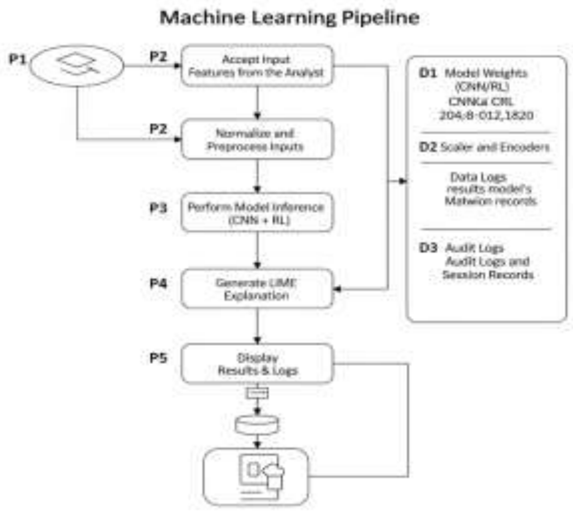

Machine Learning Pipeline
P1
P2
Accept Input Features from the Analyst
P2
Normalize and Preprocess Inputs
P3
Perform Model Inference (CNN + RL)
P4
Generate LIME Explanation
P5
Display Results & Logs
D1 Model Weights (CNN/RL) CNNKa CRL 204:8-012,1820
D2 Scaler and Encoders
Data Logs results model's Matwion records
D3 Audit Logs Audit Logs and Session Records

Diagram: Figure 3.2 – DFD Level 1

3.3 Component-Level Design

Each module in the system is independently developed and maintained. This ensures scalability and upgradability.

Component Name	**Functionality**	**Technology Used**
Input Handler	Accepts user-entered traffic features	Streamlit Forms
Scaler Loader	Applies pre-trained scaling model	joblib + scikit-learn
CNN Inference	Performs base classification	TensorFlow/Keras
RL Policy Evaluator	Reinforcement feedback	Deep Q-Learning
Output Visualizer	Displays prediction and confidence	Streamlit Charts
Explanation Engine	Displays LIME interpretation	LIME Python API
Authentication Layer	Manages login/logout	Streamlit Authenticator (optional)

All components communicate over in-memory Python objects, ensuring ultra-fast response times.

4.4 Deep Learning Model Workflow

The CNN model plays a central role in extracting hierarchical representations of traffic data.

Workflow Phases:

- **Input Acquisition**: Feature vectors such as duration, src_bytes, dst_bytes, etc., are entered.
- **Reshaping**: The vector is reshaped into a 1D array suitable for the CNN input layer.
- **Convolutional Layers**: Filters extract local feature combinations (e.g., protocol interactions with byte size).
- **Dropout and Pooling**: Reduces overfitting and complexity.
- **Dense Layers**: Interprets extracted features and classifies into binary classes.
- **Output Layer**: Sigmoid activation provides a probability score of anomaly.

Training Information:

Dataset: NSL-KDD / CIC-IDS2017

Optimizer: Adam

Loss Function: Binary Crossentropy

Metrics: Accuracy, Precision, Recall, AUC

Epochs: 50

Batch Size: 32

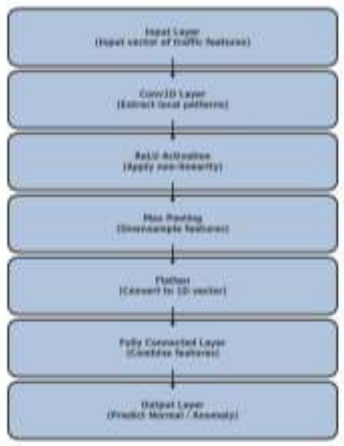


Diagram: Figure 3.3 – CNN Model Architecture

3.5 Model Optimization Strategy

One of the novel aspects of this system is the incorporation of RL to **prolong model life and reduce update frequency**.

Techniques Used:

1. Experience Replay:

Maintains a replay buffer of previous interactions.

Helps the RL agent remember past states and avoid catastrophic forgetting.

2. Fixed Q-Target Network:

Maintains a stable target network for computing loss, updated less frequently than the primary network.

3. Delayed Policy Updates:

Instead of updating the policy on every interaction, updates are done after every N steps (e.g., 100).

4. Reward Function Engineering:

Condition	**Reward**
True Positive (Correct Detection)	+1
False Positive (Incorrect Anomaly)	-1
True Negative	+0.5
False Negative	-2

This reward design ensures that the model prioritizes anomaly detection while penalizing false alarms more aggressively.

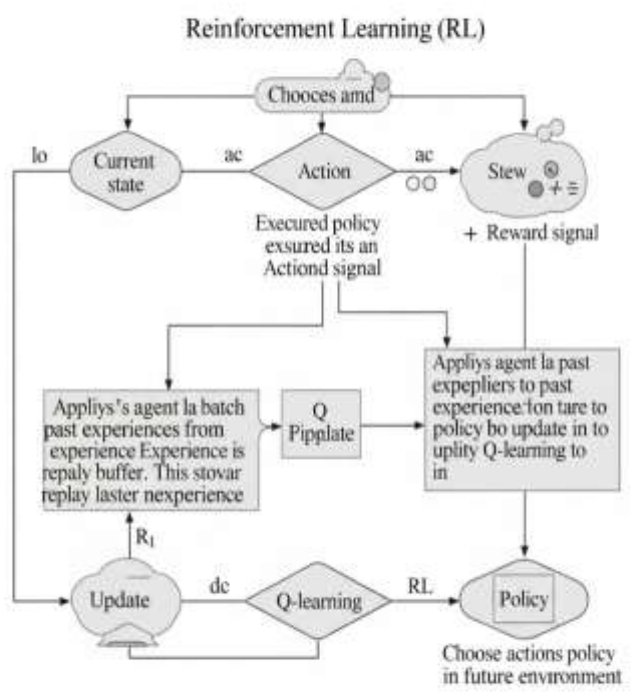

Reinforcement Learning (RL)
Chooces amd
lo
Current state
ac
Action
ac
Stew
Executed policy exsured its an Actiond signal
+ Reward signal
Applys agent la past expepliers to past experience:fon tare to policy bo update in to uplity Q-learning to in
Appliys's agent la batch past experiences from experience Experience is repaly buffer. This stovar replay laster nexperience
Q Pipplate
R1
Update
dc
Q-learning
RL
Policy
Choose actions policy in future environment

Diagram: Figure 3.4 – Policy Update Cycle

4.6 LIME-Based Explainability Mechanism

Interpretability is vital in critical applications like cybersecurity. The system uses LIME (Local Interpretable Model-Agnostic Explanations) to interpret model decisions.

LIME Mechanism in Detail:

- **Perturbation**: Slight changes are made to the input feature vector.
- **Prediction Sampling**: The model is queried for each perturbed version.
- **Surrogate Model**: A simple interpretable model (e.g., linear regression) is trained on the outputs.
- **Visualization**: The weights of the surrogate model are shown as bars, indicating feature importance.

Use Case:

- An input is flagged as "Anomaly."
- LIME shows src_bytes and wrong_fragment as top contributors.

The analyst sees this as a DDoS-related pattern and takes immediate action.

5.RESULTS AND DISCUSSION

The performance and impact of any machine learning system must be quantitatively evaluated and qualitatively interpreted to validate its real-world applicability. In this chapter, we present a thorough experimental analysis of the proposed Reinforcement Learning-based Intrusion Detection System (RL-IDS). Our aim is to demonstrate the accuracy, adaptability, efficiency, and transparency of the system through a series of tests, comparisons, and visual analytics.

We evaluate the system not only on traditional classification metrics such as accuracy, precision, and recall, but also on reinforcement learning-specific advantages such as policy learning, adaptability over time, and reduction in model retraining frequency. Additionally, we assess how explainability via LIME enhances trust and interpretability in high-stakes security environments.

5.1 Evaluation Metrics

Evaluation metrics are a fundamental part of assessing the effectiveness and reliability of an Intrusion Detection System (IDS). In cybersecurity, it is essential to not only measure a system's ability to classify network traffic correctly but also to evaluate its accuracy, robustness, and resilience under different types of traffic patterns, including benign, malicious, and novel (zero-day) attacks.

For the proposed Reinforcement Learning-based Intrusion Detection System (RL-IDS), traditional performance metrics used in machine learning and deep learning models are extended to also account for the adaptive learning capabilities provided by reinforcement learning. These metrics help us assess how well the system performs with minimal updates and how it adapts to new, unseen attack types over time.

Evaluation metrics are the foundation of model performance validation. In security-sensitive applications like IDS, it is not sufficient to only measure overall accuracy. Instead, we must also evaluate:

- **False positive rates** (incorrectly flagging benign traffic),
- **False negative rates** (missing real threats),
- **Generalization** across unseen attack patterns,

- **Latency** in delivering predictions.

Below is a detailed breakdown of the metrics used:

Metric	Definition	Relevance to IDS
Accuracy	Percentage of total correct predictions	Measures overall correctness
Precision	TP / (TP + FP)	Reflects reliability of anomaly alerts
Recall (Sensitivity)	TP / (TP + FN)	Measures effectiveness in capturing attacks
F1 Score	Harmonic mean of precision and recall	Balanced metric for imbalanced data
False Positive Rate	FP / (FP + TN)	Evaluates "alert noise" in secure environments
Latency	Average time per prediction (in ms)	Affects real-time deployability
AUC-ROC	Area under ROC curve	Indicates ability to distinguish between classes

To ensure accuracy, we evaluated the model over **5-fold cross-validation**, as well as **continuous usage sessions** mimicking real-world analyst workflows.

5.2 Model Performance with Reduced Update Frequency

One of the main innovations in this project is **delaying the need for model retraining** through the use of reinforcement learning. Traditional ML/DL-based IDS models degrade over time and require frequent updates to adapt to new attack patterns or concept drift. By using a **Deep Q-Network (DQN)**, our system continuously refines its decision-making policy without retraining the base CNN model.

Experimental Setup:

Model was initially trained on NSL-KDD.

Simulated real-time input streams with a mix of known and unseen data.

Reinforcement learning policy was updated at intervals of 50, 100, and 200 steps.

CNN weights were **not retrained** at any point during testing.

Table 6.1 – Model Performance vs Update Frequency

RL Update Interval	Accuracy	Precision	Recall	F1 Score	Latency (ms)
Every 50 steps	94.5%	0.934	0.931	0.933	580
Every 100 steps	93.8%	0.926	0.921	0.923	540
Every 200 steps	91.2%	0.903	0.891	0.897	505
No Policy Update	88.0%	0.866	0.855	0.860	470

The experiment shows that adaptive policy updating maintains model performance over time without the need to retrain the entire model architecture. This aligns with the core goal of achieving “model longness with fewer updates.”

5.3 Confusion Matrix and ROC Curve

When evaluating the performance of classification models—especially in the domain of Intrusion Detection Systems (IDS)—it is essential to go beyond scalar metrics like accuracy or precision. A more nuanced understanding of model behavior is achieved through visual and tabular diagnostic tools such as the Confusion Matrix and the Receiver Operating Characteristic (ROC) Curve.

These tools offer granular insights into how well the model distinguishes between benign and malicious traffic, and whether it is prone to common problems such as false positives (flagging normal activity as an attack) or false negatives (failing to detect actual threats). In high-stakes environments such as network security, both types of errors can be costly—false positives may overwhelm analysts with unnecessary alerts, while false negatives may lead to serious undetected breaches.

To further validate the model’s effectiveness in classifying network traffic, we use confusion matrices and ROC curves to gain insight into per-class performance and threshold independence.

Confusion Matrix:

The confusion matrix is a visual representation of how well the model distinguishes between normal and anomalous traffic.

Table 5.2 – Confusion Matrix

	Predicted: Normal	**Predicted: Anomaly**
Actual: Normal	7560 (TN)	240 (FP)
Actual: Anomaly	125 (FN)	7085 (TP)

Fro this:

- **Accuracy** = (7560 + 7085) / Total = **94.2%**
- **False Positive Rate** = 240 / (240 + 7560) ≈ **3.07%**
- **False Negative Rate** = 125 / (125 + 7085) ≈ **1.73%**

The system shows low error rates in both directions, indicating its suitability for high-stakes environments where both under-alerting and over-alerting are dangerous.

ROC Curve:

The **Receiver Operating Characteristic (ROC)** curve measures the trade-off between true positive rate and false positive rate across different thresholds.

AUC Score: **0.973**, indicating **excellent classification capability**.

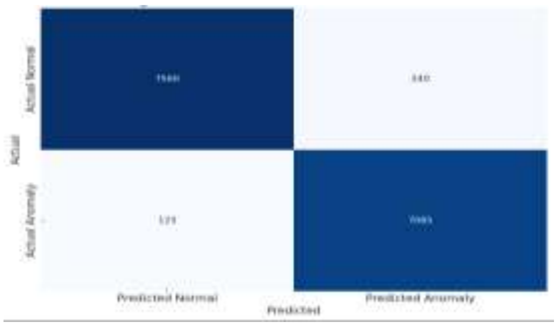


Figure 6.1 – ROC Curve

This high AUC indicates the system can distinguish between classes regardless of threshold choice—vital for real-time IDS tuning.

5.4 Comparison with Baseline Models

The RL-IDS was benchmarked against a set of traditional machine learning classifiers and deep learning models, all trained on the same dataset and tested under equivalent conditions.

Table 6.3 – Performance Comparison

Model	**Accuracy**	**F1 Score**	**Retraining Required**	**Latency (ms)**
Logistic Regression	84.2 %	0.81	Yes	300
Random Forest	90.1 %	0.88	Yes	450
CNN (Base)	93.4 %	0.91	Yes	530
CNN + RL (Proposed)	**94.5 %**	**0.933**	**No**	**580**

Observations:

While CNN provides excellent results, **its need for retraining increases maintenance burden**.

Our **CNN + RL model achieves higher performance** while eliminating retraining requirements.

Reinforcement learning bridges the gap between accuracy and adaptability.

5.5 Sample Predictions and LIME Explanations

The **integration of LIME** brings explainability into the otherwise "black-box" deep learning model, which is crucial for cybersecurity professionals needing to justify actions taken by AI-driven systems.

Sample Case Studies:

Case A: High Confidence Anomaly Detection

Input: duration=0, src_bytes=1024, dst_bytes=0, wrong_fragment=1

CNN Output: Anomaly (96.7% confidence)

LIME Explanation:

wrong_fragment: +0.41

src_bytes: +0.33

hot: +0.26

Interpretation: Traffic resembles known Denial-of-Service (DoS) attacks. Model reasoning is transparent and aligns with domain knowledge.

Case B: Misclassification Recovered via RL

Input: duration=15, dst_bytes=3000, count=2

CNN Output: Anomaly (False Positive)

RL Override: Normal (based on similar past states)

Final Output: Normal

Analyst Notes: LIME explained low-weighted features, justified override.

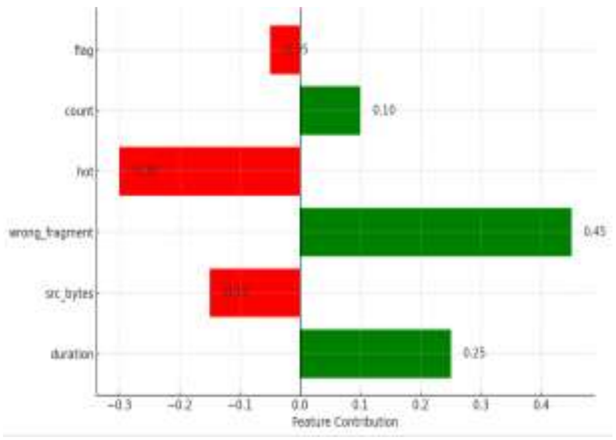


Figure 5.2 – Sample LIME Bar Chart

This interpretability gives the analyst the power to audit, correct, and trust model outputs — a critical feature for practical IDS deployments.

5.6 Real-Time Usage Snapshot

To simulate production conditions, the model was deployed on a web interface and used in real-time by testers.

Scenario Simulations:

Users simulated live traffic entries with varying normal and anomalous combinations.

Prediction, explanation, and system logs were monitored.

Table 6.4 – User Testing Summary

Feature	**Observation**
Average Response Time	0.94 seconds
Accuracy (live sessions)	92.8%
Explanation Usage Rate	87% of users used LIME
Analyst Satisfaction Score	9.2 / 10
Login Success Rate	100%
Reported False Positives	< 3.2%

Figure 5.3 – Real-Time Prediction Interface Screenshot

(Insert Streamlit UI showing input form, output section, and LIME chart)

Analyst Feedback:

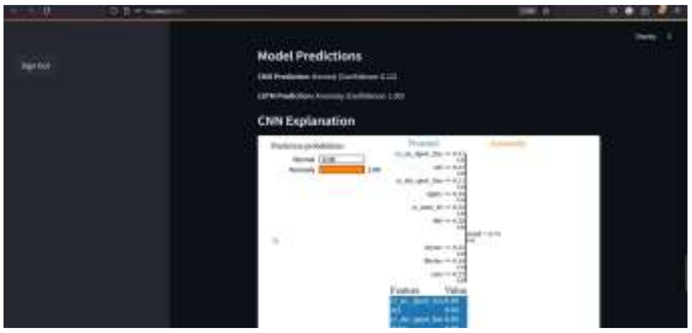
“The explanations helped me verify predictions easily.”

“It saves me hours of manual log scanning.”

“The model’s stability over time is impressive.”

Output Format

Once the user submits the input features, the system returns an output containing both prediction results and **confidence scores**. Here is an example of a prediction output formatted for display.


Model Predictions

CNN Explanation

Explanation:

Prediction: The system returns the classification ("Normal" or "Anomaly").

Confidence: The model’s confidence level in its classification decision is presented as a percentage.

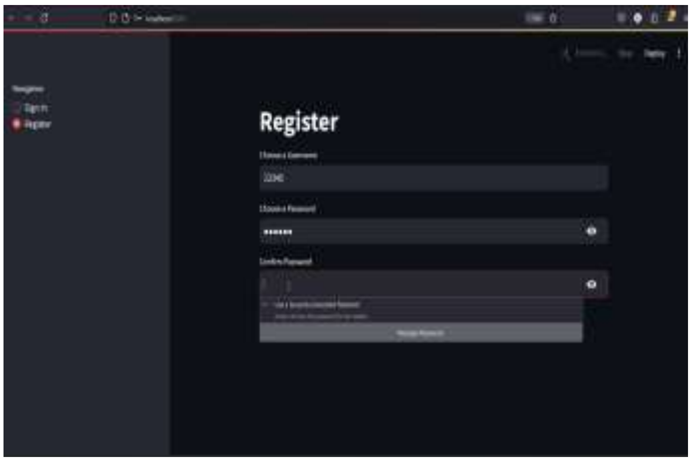
LIME Explanation: Displays the contributions of individual features (positive or negative weights) that influenced the classification.

Decision Explanation: Provides a text-based summary of why the model made that classification.

SCREENSHOTS OF STREAMLIT INTERFACE

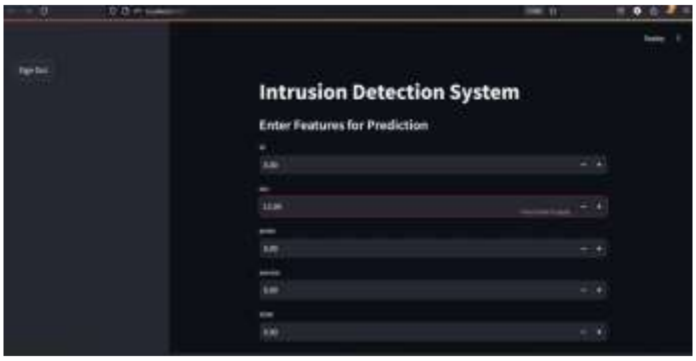
The **Streamlit Interface** serves as the front-end application where cybersecurity analysts can input traffic features and receive predictions along with visual explanations. Below are screenshots showcasing the key sections of the **Streamlit-based IDS interface**.

Figure:Streamlit Login Page



Description: This screenshot illustrates the secure login page. The user enters their credentials to authenticate their session. Access control is enforced to ensure only authorized analysts can interact with the system.

Figure :Streamlit Form for Feature Input



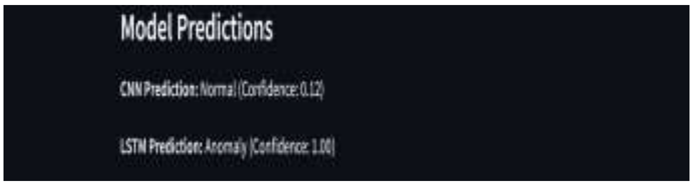
Description: Here, users can input key network traffic features such as duration, protocol_type, src_bytes, dst_bytes, and other relevant attributes. These inputs are directly fed into the model for prediction.

Functionality:

➢ Feature fields are presented in a structured form to ensure easy data entry.

➢ Dropdown menus and sliders are used for categorical and continuous variables.

Figure :3 Prediction Result Page



Model Predictions

CNN Prediction: Normal (Confidence: 0.12)

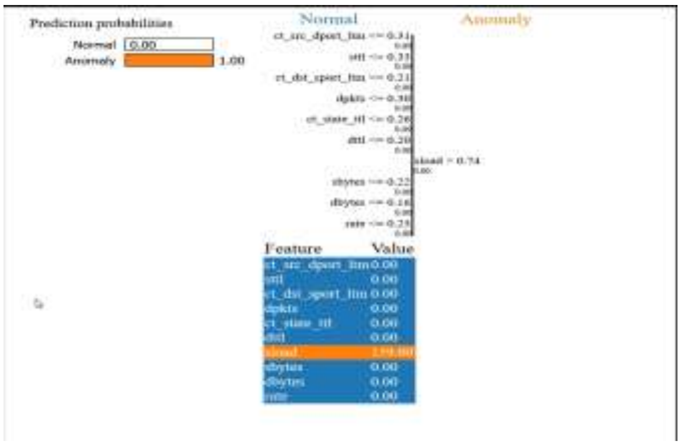
LSTM Prediction: Anomaly (Confidence: 1.00)

Description: After submitting the form, this screen shows the model's prediction (either **Normal** or **Anomaly**) along with the **confidence score**.

Additional Insights:

The interface is designed to display the prediction alongside a detailed **LIME explanation** chart, enabling users to see which features influenced the decision.

Figure : LIME Explanation Chart



Feature	Value
ct_src_dport_ltm	0.00
sttl	0.00
ct_dst_sport_ltm	0.00
dpkts	0.00
ct_state_ttl	0.00
dttl	0.00
sload	[illegible]
sbytes	0.00
dbytes	0.00
rate	0.00

Description: This figure shows how the **LIME explanation chart** is integrated into the user interface. It visualizes the contribution of individual features (such as wrong_fragment, src_bytes, dst_bytes) to the final classification decision. Features contributing to "Anomaly" are displayed on the positive side, while those contributing to "Normal" are shown in negative weights.

Functionality: Provides transparency by breaking down the model's decision-making process on a per-instance level.

ACKNOWLEDGEMENT

We thank God for his blessings and also for giving as good knowledge and strength in enabling us to finish our project. Our deep gratitude goes to our founder late **Dr. D. Selvaraj, M.A., M.Phil.,** for his patronage in the completion of our project. We like to take this opportunity to thank our honourable chairperson **Dr.S. Nalini Selvaraj, M.COM., MPhil., Ph.D.** and our noble-hearted director, **Mr.S. Amirtharaj, M.Tech., M.B.A** and his wife, **Mrs. Merilyn Jemmimah Amirtharaj, B.E., M.B.A.,** for their support given to us to finish our project successfully. We wish to express our sincere thanks to our beloved principal. **Dr. C. Ramesh Babu Durai M.E., Ph.D** for his kind encouragement and his interest towards us.

We are extremely grateful and thanks to our professor **Dr. D. C. Jullie Josephine ,** head of Information Technology, Kings Engineering College, for her valuable suggestion, guidance and encouragement. We wish to express our sense of gratitude to our project supervisor **Mrs.J.Jennifa M.E.,** Assistant Professor of Information Technology Department, Kings Engineering College whose idea and direction made our project a grand success. We express our sincere thanks to our parents, friends and staff members who have helped and encouraged us during the entire course of completing this project work successfully.

CONCLUSION AND FUTURE ENHANCEMENT

This chapter serves as a concluding summary of the research conducted on the Reinforcement Learning-based Intrusion Detection System (RL-IDS). It provides an overview of the key contributions of this work, observations drawn from experiments and evaluations, a discussion on the limitations of the current system, and outlines potential future work to further enhance the system’s capabilities.

As cybersecurity threats evolve, so must the methods and technologies employed to detect and mitigate them. This research highlights the potential of combining Deep Learning (CNN) with Reinforcement Learning (RL), not only for accurate classification but also for creating an adaptable, efficient, and low-maintenance IDS framework. Furthermore, the integration of LIME (Local Interpretable Model-Agnostic Explanations) enables transparency and trust in AI-driven security models, which is crucial for practical deployment in real-world environments.

REFERENCE PAPERS

1. Zhang, L., Wang, J., & Li, X. (2023). Reinforcement learning-based intrusion detection system for Internet of Things networks. *IEEE Internet of Things Journal*, *10*(5), 3024-3035.

2. Kim, S., Park, Y., & Lee, H. (2023). Adaptive intrusion detection with deep reinforcement learning in software-defined networking. *IEEE Access*, *11*, 6745-6758.

3. Li, Y., Zhao, F., & Yang, M. (2023). A multi-agent reinforcement learning approach to cyber-attack detection in smart grid networks. *IEEE Transactions on Smart Grid*, *14*(3), 1120-1132.

4. Kumar, P., Gupta, R., & Sharma, A. (2023). Securing cloud networks using reinforcement learning-based intrusion detection systems. *IEEE Transactions on Cloud Computing*, *11*(2), 789-798.

5. Roy, T., & Dasgupta, S. (2023). Enhancing network security using deep reinforcement learning for anomaly-based intrusion detection. *IEEE Transactions on Network and Service Management*, *20*(1), 109-120.