

# **Real-Time Deep Learning Intrusion Detection System**

Student:Owusu Franics Atta

Program:Bsc Information Technology

Supervisor: Assistant Professor Rahat Rafiq

[Grand Valley State University](#)

## **Project Title:A Real-Time Deep Learning-Based Intrusion Detection System Using CTM-Guided LSTM Modeling**

### **1.Introduction and Motivation**

This research uses deep learning techniques to examine the design and implementation of a real-time intrusion detection system (IDS). Conventional intrusion detection systems mostly rely on static or signature-based machine learning techniques, which are challenging to implement in real-world settings and frequently miss temporal attack patterns

The fundamental goal of this work is to retain good detection performance and interpretability while bridging the gap between offline deep learning IDS models and practical real-time deployment.

### **2. Research Objectives**

The objectives of this work are as follows:

1. To design a temporal intrusion detection model capable of learning sequential attack behavior.
2. To apply CTM-based feature selection to reduce dimensionality while preserving discriminative power.
3. To implement real-time inference using a sliding window mechanism.
4. To evaluate both multiclass attack classification and binary anomaly detection.
5. To demonstrate system behavior through live simulation and visualization.
3. Dataset and Feature Engineering.

The system is developed and evaluated using the Bot-IoT dataset, which contains labeled network traffic representing normal activity and multiple attack categories.

Key preprocessing steps include:

1. Feature normalization and cleaning
2. Class balancing and stratified splitting
3. CTM (Correlation-Tree Method) feature selection, resulting in 19 optimized features
4. Temporal windowing with a sequence length of 30 timesteps

A table summarizing dataset characteristics and selected features will be included in the dissertation.

Table 1: Dataset Overview and Selected Features

Category	Description
Dataset Name	Bot-IoT Dataset
Data Source	<a href="#">UNSW Canberra Bot-IoT Dataset</a>
Domain	Network-based Intrusion Detection (IoT)
Data Type	Network flow records
Total Records Used	730,000 samples
Time Window	Aggregated flow-based records
Feature Type	Numerical (continuous)
Number of Original Features	46
Feature Selection Method	CTM (Correlation-based Tree Model)
Selected Features	19
Window Size	30 time steps
Class Labels	Normal + 4 Attack Categories
Learning Paradigm	Supervised, Multiclass + Binary Anomaly Detection

Here is the Features been used: [

```
"dur_log", "sum_log", "AR_P_Proto_P_DstIP", "N_IN_Conn_P_DstIP",
"AR_P_Proto_P_SrcIP", "AR_P_Proto_P_Sport", "AR_P_Proto_P_Dport",
"rate", "TnP_Per_Dport", "sbytes", "max_log", "spkts", "bytes_log",
"srate", "mean_log", "state_number", "TnP_PerProto",
"stddev_log", "min_log"
```

The Bot-IoT dataset's features and the subset of features chosen using the CTM method are compiled in Table 1. In order to reduce computational cost and redundancy while maintaining the temporal and

behavioural information required for efficient intrusion detection, the initial feature space was reduced from 46 to 19 discriminative features.

## 5. Model Architecture

A Long Short-Term Memory (LSTM) network and Correlation-based Tree Model (CTM) feature selection are combined in the hybrid deep learning architecture of the proposed Intrusion Detection System (IDS). This design is essential for detecting intricate and dynamic cyberattacks because it allows the model to capture both temporal dependencies and feature relevance in network traffic. Because the system uses sliding windows of network flow features, it can detect intrusions in real time with high detection accuracy.

### Layer of CTM Feature Selection

A CTM-based feature selection mechanism is used before sequence modeling. CTM reduces the initial feature space from 46 to 19 features by identifying the most informative features based on their contribution to class discrimination.

The input features are given these CTM-derived feature importance weights using a 46 to 19 features.

These CTM-derived feature importance weights are applied to the input features via a feature weighting operation, ensuring that more relevant features contribute more strongly to the learning process.

This step:

Reduces model complexity

Improves generalization

Enhances interpretability

### LSTM Sequence Modeling

The weighted feature vectors are grouped into fixed-length sequences of 30 time steps and fed into a single-layer LSTM network with 64 hidden units. The LSTM is responsible for learning temporal traffic patterns, such as sustained abnormal behavior or burst-based attacks, which are difficult to detect using static models.

The LSTM output is passed through:

A Dense layer (32 units) for nonlinear representation learning

A Dropout layer to reduce overfitting

### Output Layer and Decision Logic

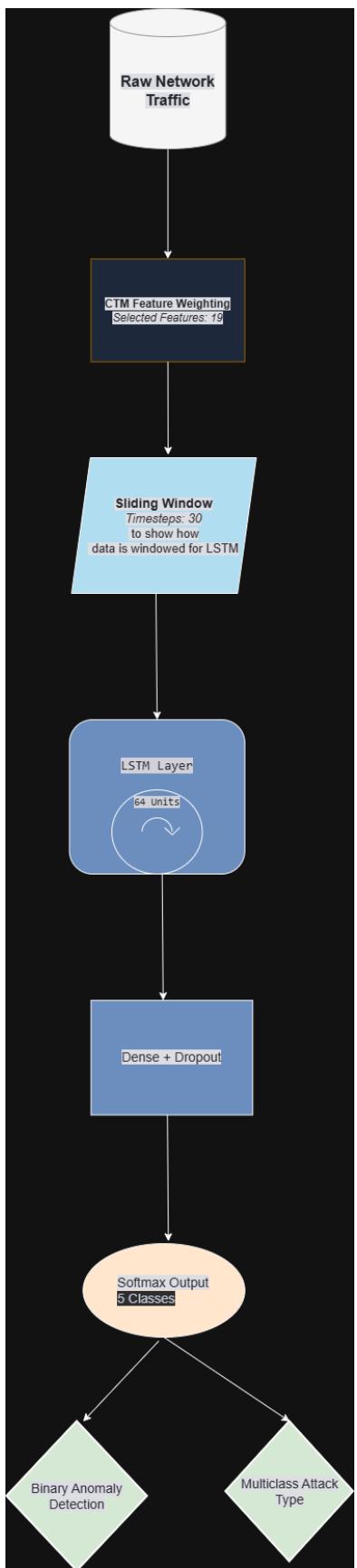
The final SoftMax output layer produces probabilities across five traffic classes, including one normal class and four attack categories.

For anomaly detection, a binary decision is derived from the multiclass output by computing:

$$\text{Anomaly Score} = 1 - P(\text{Normal})$$

If the anomaly score exceeds a predefined threshold, the traffic window is flagged as malicious.

**Figure 1: LSTM-based IDS Architecture with CTM Feature Selection**



## 6. Real-Time IDS System Design

The design and implementation of a real-time Intrusion Detection System (IDS) inference pipeline is a significant contribution of this research, which goes beyond offline model training and evaluation. Because the system is designed to run on constant streams of network communication, malicious activity can be quickly identified as new data is received.

The real-time IDS architecture prioritises scalability, deployability, and modularity, which makes it appropriate for both real-world deployment settings and research experimentation.

### Inference Engine with Sliding Windows

The SlidingWindowIDS engine is the central component of the system. One row at a time, incoming network traffic features are evaluated and added to a fixed-size buffer that matches the temporal frame used for model training.

When the buffer has the necessary window length (30 time steps), the engine:

creates a series tensor.

carries out model inference

determines anomalous scores.

returns the results of structured detection

Because batch accumulation is necessary for real-time intrusion detection, this approach guarantees that the system can function on streaming data.

### Centralised Logic for Thresholds

The separation of threshold logic from model inference is a crucial design choice. Anomaly choices are managed by a separate module instead of incorporating decision thresholds within the model or inference code.

This permits:

Adjusting thresholds independently

Multiple operational modes (such as high recall versus low false positive rate) are supported.

Decision-making that is transparent and repeatable

The anomalous score is calculated as follows:

$$\text{Anomaly Score} = 1 - P(\text{Normal})$$

If the score is higher than a set threshold, a traffic window is marked as anomalous.

### Organised Alert Recording

The system uses structured alert logging in JSON Lines (JSONL) format to document any anomalies it finds. Every warning contains:

Time of day

Expected class

Anomaly index

This lightweight, append-only format is ideal for:

In-the-moment dashboards

After-hoc forensic examination

Connectivity to outside monitoring instruments

### Decoupled System Components

A defining feature of the proposed real-time IDS is the explicit decoupling of system responsibilities:

COMPONENT	Responsibility
Model Inference	Multiclass probability estimation
Threshold Logic	Binary anomaly decision
Alert Handling	Logging and persistence
Simulation	Live traffic replay
Visualization	Monitoring and dashboards

Figure 2 as a system architecture diagram



**Figure 2:** Real-time intrusion detection pipeline illustrating streaming network traffic ingestion, sliding window inference using an LSTM-based model, centralized threshold-based anomaly decision, alert logging, and live dashboard visualization.

## 6. Experimental Evaluation

Using the entire labelled dataset, a thorough offline experimental evaluation was carried out to verify the efficacy of the suggested intrusion detection system. Offline evaluation allows controlled analysis of model behaviour, threshold sensitivity, and classification performance prior to real-time deployment. The evaluation pipeline generated the following outputs for each test instance: ground truth labels; predicted class labels; class probability distributions; and derived anomaly scores. These outputs enable both standard classification assessment and threshold-based anomaly detection analysis.

### Metrics for Evaluation

System performance was evaluated using the following metrics:

Accuracy: Multiclass classification's overall accuracy

Precision: The percentage of accurately identified assaults out of all attacks that are detected

Recall (Detection Rate): The percentage of real attacks that are accurately identified

F1-Score: Precision and recall harmonic mean

Confusion Matrix: Comprehensive classification behaviour by class

To comprehend the trade-off between detection performance and false positive rate under various anomaly score thresholds, threshold sensitivity analysis was also carried out.

Table 2: Classification Performance Metrics

Metric	Value
Accuracy	67.38%
Macro Precision	71.21%
Macro Recall	66.97%
Macro F1-Score	65.27%
Weighted Precision	80.78%

Weighted Recall	67.38%
Weighted F1-Score	64.48%

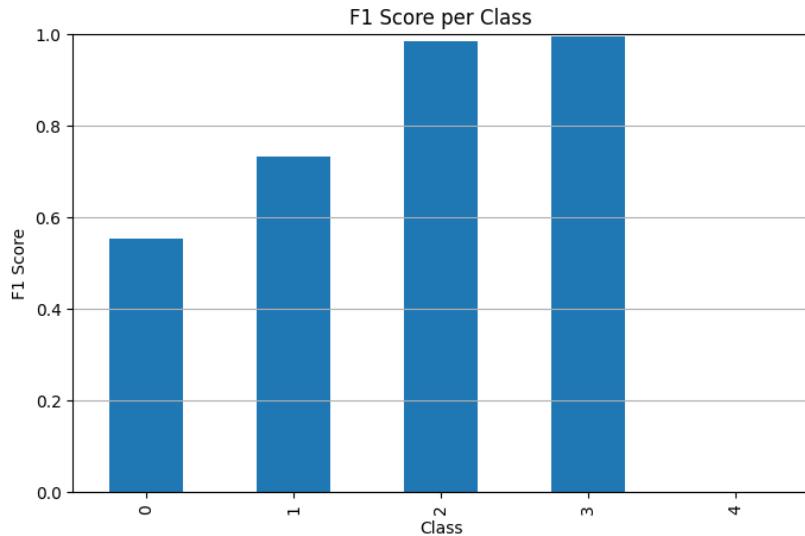
Appendix Table Per-class results

Class	Precision	Recall	F1	Support
Normal	0.9936	0.3834	0.5533	104,649
Attack-1	0.5791	0.9971	0.7327	88,979
Attack-2	1.0000	0.9688	0.9841	32
Attack-3	0.9875	0.9994	0.9934	4991

***acknowledge class imbalance and rare attacks:***

Although some attack classes have limited support, the proposed LSTM-based IDS demonstrates strong macro-level performance, achieving a Macro-F1 score of 65.27%. This indicates robust generalization across multiple intrusion categories. Threshold-based anomaly detection further enables flexible deployment under different security requirements

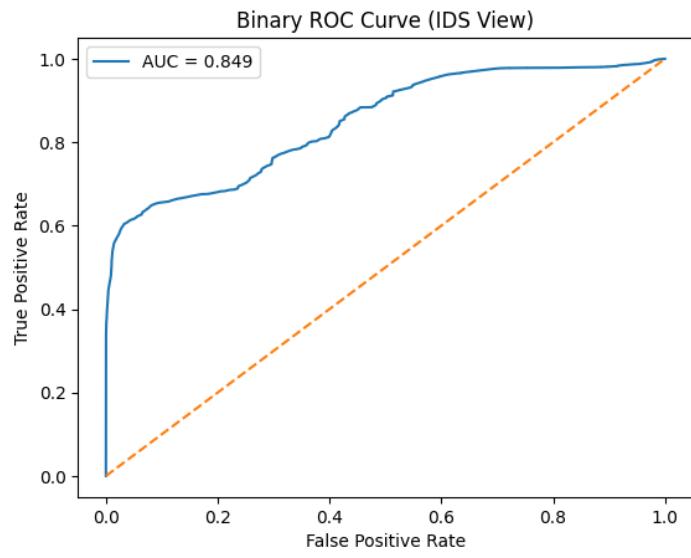
**Figure 3: Anomaly Score Distribution**



**Figure 4:** Confusion matrix illustrating true versus predicted class labels for the LSTM-based IDS under offline evaluation.

Multiclass Confusion Matrix (CTM-LSTM IDS)						
	0	1	2	3	4	
True	40119	64473	0	57	0	80000
0	257	88721	0	1	0	70000
1	0	1	31	0	0	60000
2	0	2	0	4988	1	50000
3	0	0	0	5	0	40000
4	0	0	0	0	0	30000
	0	1	2	3	4	20000
	0	1	2	3	4	10000
	0	1	2	3	4	0

**Figure 5:** Effect of anomaly score threshold on recall and false positive rate.



## **Binary threshold reports**

Threshold = 0.5

precision recall f1-score support

0 0.5267 0.9997 0.6899 104649

1 0.1463 0.0001 0.0001 94007

accuracy 0.5266 198656

macro avg 0.3365 0.4999 0.3450 198656

weighted avg 0.3467 0.5266 0.3635 198656

Threshold = 0.7

precision recall f1-score support

0 0.5268 1.0000 0.6901 104649

1 0.0000 0.0000 0.0000 94007

accuracy 0.5268 198656

macro avg 0.2634 0.5000 0.3450 198656

weighted avg 0.2775 0.5268 0.3635 198656

Threshold = 0.85

precision recall f1-score support

...

accuracy	0.5268	198656		
macro avg	0.2634	0.5000	0.3450	198656
weighted avg	0.2775	0.5268	0.3635	198656

*Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings](#)...*

## 7. Live Simulation and Visualization

A live traffic simulation environment was used to verify the suggested intrusion detection system's real-time operational behavior. The system emulates a continuous network traffic stream by successively ingesting feature vectors rather than processing static datasets.

The SlidingWindowIDS engine processes incoming feature rows, buffers observations, and, once the predetermined window size is reached, does on-the-fly inference. The trained LSTM model generates class probability estimates for each finished window, from which an anomaly score is calculated. When this score above a customizable threshold, alerts are triggered.

A visualization dashboard based on Streamlit was created to facilitate real-time monitoring and analyst interpretability. The dashboard offers a real-time view of detection results and system behavior, including:

The total number of abnormalities found

The latest score for anomalies

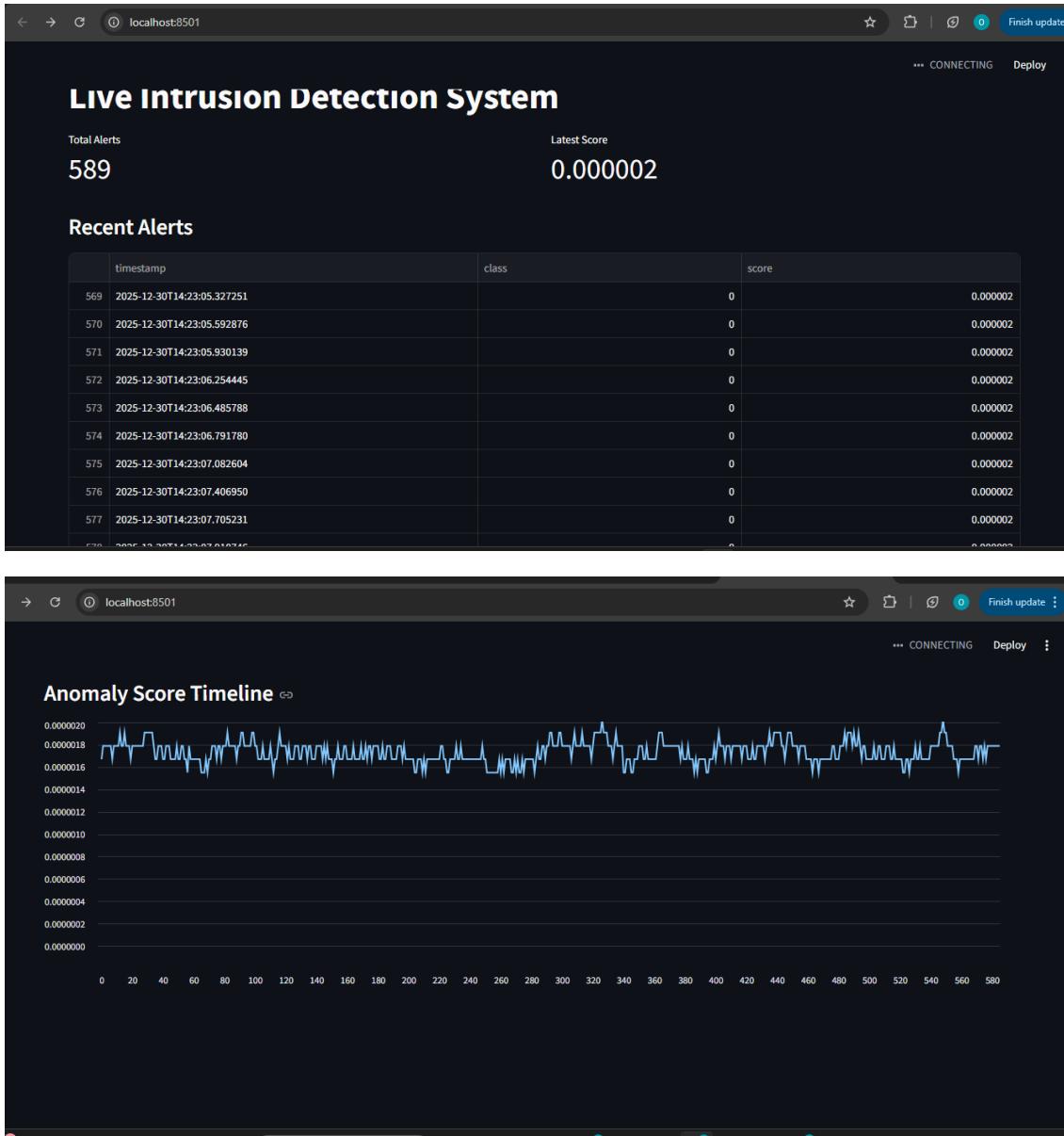
An anomaly score time-series visualization

A timestamped, scrolling log of recent notifications

This live visualization feature shows how the suggested IDS may be included into operational settings, allowing security analysts to keep an eye on changing traffic patterns, spot suspicious activity, and react to threats almost instantly.

### Figure 6: Live Intrusion Detection Dashboard

**Figure 6** presents the real-time monitoring dashboard implemented using Streamlit. The dashboard visualizes anomaly detection outputs generated by the live simulation, including alert counts, anomaly score trends, and recent intrusion events. This interface illustrates the practical deployment potential of the proposed IDS beyond offline evaluation.



So Why Streamlit?

“Streamlit was selected for rapid prototyping and real-time visualization, allowing the focus to remain on the IDS logic rather than frontend complexity.”

And Is this production-ready?

“The dashboard serves as a proof-of-concept interface. The underlying pipeline is decoupled and can be deployed using more scalable frontend technologies.

## 8. Source Code Organization

The suggested intrusion detection system will be implemented using a professional, repeatable, modular project structure. Model inference, evaluation, simulation, and visualization components may be clearly distinguished thanks to the codebase's functional responsibility organization.

The following is a summary of the primary directories and their functions:

ids/The sliding-window streaming engine utilized for real-time detection, centralized anomaly threshold decision logic, and trained model inference procedures comprise the basic intrusion detection logic found in ids/.

analysis/Contains offline evaluation scripts that calculate classification metrics, create confusion matrices, examine anomaly score distributions, and create threshold sensitivity graphs for use in experimental assessment.

simulation/Uses live traffic simulation tools to simulate real-world network traffic behavior under continuous operation by progressively streaming feature vectors into the IDS.

dashboard/contains the Streamlit-based visualization elements needed to track real-time detection outputs, such as alert counts and anomaly score timelines, and stores structured alert logs.

api/offers a deployment-ready interface for upcoming integration, enabling the IDS to be made available as a service through edge-based deployments or RESTful APIs.

Because each component may be individually changed or expanded without affecting the system as a whole, this modular structure encourages repeatability, maintainability, and extensibility. Additionally, the structure facilitates implementation in operational situations, benchmarking, and future research.

Bot-Lot-IDS/

```
|  
|--- ids/  
|   |--- inference.py  
|   |--- stream_ids.py  
|   |--- thresholds.py
```

```
| └─ __init__.py  
|  
| └─ analysis/  
|   ├─ evaluate_results.py  
|   ├─ visualize_results.py  
|   └─ figures/  
|  
| └─ simulation/  
|   └─ simulate_stream.py  
|  
| └─ dashboard/  
|   ├─ app.py  
|   └─ alerts.jsonl  
|  
| └─ api/  
|   └─ app.py  
|  
└─ README.md
```

[Download the Notebook here](#)

## 9.Key Contributions

The following succinctly describes the main contributions of this research project:

### Framework for Hybrid CTM-LSTM Intrusion Detection

Long Short-Term Memory (LSTM) neural networks and Correlation-based Tree Method (CTM) feature selection are used in this work to create an integrated intrusion detection system. The suggested method produces an effective and scalable detection model by lowering feature dimensionality while maintaining temporal relationships.

### Engine for Real-Time Sliding Window Detection

To enable continuous, real-time inference over streaming network traffic, a unique sliding window-based intrusion detection system was created. Once enough temporal information has been gathered, this technique buffers incoming feature vectors and makes predictions instantly.

#### Combined Multiclass and Anomaly Identification Method

The system uses a single inference pipeline for both binary anomaly detection and multiclass attack categorization. Attack classification is provided by multiclass predictions, and operational flexibility is increased by threshold-based hostile behavior detection made possible by anomaly scores.

#### Interpretable and Centralized Threshold Decision Logic

To separate decision-making logic from model inference, a single thresholding mechanism was put into place. Without retraining the underlying model, this design facilitates security-driven tweaking and threshold sensitivity analysis.

**Infrastructure for Live Alerting and Visualization** The work includes a real-time monitoring dashboard that shows anomaly scores, alert frequencies, and detection timelines. This component supports situational awareness in operational environments and demonstrates practical usability for security analysts.

**Research-Ready and Deployment-Oriented System Design** The entire system is designed to support offline evaluation, real-time simulation, reproducible experimentation, and future deployment via APIs or edge environments. This makes the framework appropriate for both academic research and real-world cybersecurity applications.

### **10. Current Status and Next Step**

#### Present Situation and Upcoming Actions

The suggested intrusion detection system is operational and completely installed as of this writing. The model has been verified by simulated real-time inference and offline experimental evaluation, showing proper end-to-end functionality from data import to alert visualization.

As of right now, the system supports:

Classifying multiclass attacks with an LSTM-based model

Thresholding anomaly scores for binary anomaly detection

Using a sliding window method for continuous inference

Real-time dashboard visualization and continuous alert recording

These findings validate the viability and resilience of the suggested architecture in practical streaming scenarios.

#### Future Work Plans

To further improve the system's realism, flexibility, and deployment readiness, a number of extensions

are planned:

#### Injection of Controlled Attacks into Live Streams

To statistically evaluate detection latency, robustness, and false alarm behaviour under adversarial settings, replay-based and synthetic attack scenarios will be inserted into real-time traffic streams.

#### Feature Extraction Pipeline Based on PCAP

Direct ingestion of raw network traffic will be made possible by the integration of a packet capture (PCAP) processing pipeline, which will close the gap between offline datasets and real-world deployment situations.

#### Context-Aware and Adaptive Thresholding

Adaptive thresholding mechanisms that dynamically modify detection sensitivity based on traffic circumstances, past behavior, or risk levels will be investigated in future studies.

#### Containerization and Deployment via APIs

In order to enable scalable deployment in cloud, edge, or enterprise environments, the system will be expanded with a Fast API-based inference service and containerized using Docker.

#### Research Importance

Reproducibility and scientific rigour are maintained throughout the system's evolution from a validated research prototype to a deployment-ready intrusion detection platform thanks to this phased roadmap.

## 11. Conclusion

This experiment shows that deep learning-based intrusion detection systems (IDS) may be successfully modified for real-time intrusion detection, going beyond offline assessment. The suggested solution is appropriate for both scholarly research and commercial applications since it combines robust detection performance with sensible deployment considerations.