

Journal of Information and Cybersecurity

ARTICLE INFO

Article History: Received: 30-05-2026, Revised: 07-07-2026, Accepted: 01-09-2026

***Corresponding author Email:** m.almaiah@ju.edu.jo

DOI:

This is an open access article under the CC BY 4.0 license
(<http://creativecommons.org/licenses/by/4.0/>).

Published by ITAP Publisher.



Vol 2026, No.1

Deep Learning-Based Adaptive Intrusion Detection for Next-Generation Network Security

Mohammed Almaayah^{1*}, Mohamed Chahine Ghanem²

¹King Abdullah the II IT School, The University of Jordan, Amman 11942, Jordan, m.almaiah@ju.edu.jo, <https://orcid.org/0000-0002-2215-2481>

² School of Computer Science and Mathematics, Keele University, Newcastle-Under-Lyme, ST55BG, UK, m.ghanem@keele.ac.uk, <https://orcid.org/0000-0002-7067-7848>

Abstract

Next-generation network environments including 5G/6G radio access networks, dense Internet-of-Things (IoT) deployments, and software-defined infrastructures dramatically expand the attack surface and generate highly dynamic, non-stationary traffic patterns that render static, signature-based intrusion detection systems (IDS) increasingly ineffective. This paper proposes an adaptive deep learning-based intrusion detection framework that combines a hybrid Convolutional Neural Network–Bidirectional Long Short-Term Memory (CNN-BiLSTM) architecture with an attention mechanism for joint spatial-temporal feature learning, coupled with an online concept-drift detection module that triggers incremental model retraining whenever the statistical distribution of network traffic deviates from the learned baseline. The framework was evaluated on a combined corpus exceeding 3.2 million labeled flow records drawn from three benchmark intrusion datasets, replayed through a simulated software-defined networking (SDN) testbed under controlled concept-drift conditions. The proposed model achieved a detection accuracy of 99.1% and an F1-score of 98.8%, sustaining an average per-flow inference latency of 4.2 ms under moderate load, and outperformed CNN-only, LSTM-only, Random Forest, and Support Vector Machine baselines by 1.2 to 7.7 percentage points in accuracy. The adaptive retraining mechanism was able to recover detection accuracy from a 6.3-point degradation to within 0.4 points of baseline, on average, after a simulated concept drift occurred, while a non-adaptive static model exhibited persistent degradation from 6.3 points below the baseline. Having been assessed by the CVSS, it also proves to be very robust against attacks such as evasion, adversarial perturbation, or traffic-mimicry. The results show that deep spatio-temporal representation learning and lightweight online adaptation have the potential to be a practical and scalable strategy for self-adapting intrusion detection for next-generation network environments.

Keywords: Intrusion Detection System; Deep Learning; Concept Drift; Adaptive Learning; CNN-BiLSTM; Attention Mechanism; Network Security; Software-Defined Networking.

1. Introduction

1.1 Background: The Evolving Threat Landscape of Next-Generation Networks

The combination of 5G and the new wave of 6G radio access technologies, the massive proliferation of Internet-of-Things (IoT) devices, edge and fog computing, and the general trend towards software-defined networking (SDN) and network function virtualization (NFV) is bringing about a structural change in network infrastructures [1]. The result has been a transformation of bandwidth, flexibility and the service programmability of services, and an unprecedented increase in the number of attack pathways for adversaries. The reports of global cyber-threat activity show that over the last few years, especially, the number and sophistication of distributed denial-of-service (DDoS) incidents, botnets and compromises of IoT devices, and lateral-movement intrusions has been increasing dramatically, and next-generation network deployments have been cited as one of the factors as they are vast and diverse [2].

In contrast to traditional enterprise networks, next-generation networks have very high traffic volumes, short-lived and transient flows, encrypted payloads that do not allow deep packet inspection, and continually changing sets of connected endpoints. This results in a non-stationary traffic environment where the traffic statistics of benign and malicious traffic change over time, known as concept drift [3]. Intrusion detection systems that are trained once on a fixed dataset and deployed without further adaptation tend to experience a steady decline in detection accuracy as the live traffic distribution diverges from the distribution observed during training.

Undetected intrusions in these environments can have a significant economic and operational impact. Given the recent industry incident-response surveys which have estimated the cost of a data breach with IoT or industrial control assets to be in excess of several million dollars after containment, regulatory penalties and reputational costs are considered [4]. Detection latency or even a 'modest' improvement in adaptive accuracy can mean 'materially less' exposure to sustained compromise, and with next generation networks operating at such a scale, that's not something to take lightly.

1.2 Limitations of Traditional and Static Machine Learning-Based IDS

Traditional intrusion detection methods can be classified into three main categories: signature-based detection, classical anomaly-based detection and static machine-learning-based classifiers. Signature-based systems compare the traffic to a database of known attack signatures. They work well against known attacks, but are not structurally capable of detecting a zero-day attack or a polymorphic attack – they need to be continuously and manually updated with signatures to be effective [5].

Classical anomaly-based systems learn a statistical signature of the normal behavior and identify deviations as suspicious. While these systems are capable, in principle, of detecting novel attacks, they tend to have high false-positive rates in dynamic environments where legitimate traffic patterns naturally vary, and are invariably based on hand-crafted thresholds that need to be manually adjusted often. [6]

Machine-learning algorithms using IDS on Support Vector Machines (SVM), Decision Trees, or Random Forests obtained a higher degree of accuracy for detection than did the rule based ones by learning the decision boundaries directly from labeled data. Typically, these models are learned offline from a static traffic snapshot and then deployed without the ability to continuously adapt. This is a well-known effect in concept-drift literature, and a fact that is not discussed directly in the context of IDS deployments, except for interesting findings in [7] regarding the degradation of a static classifier's accuracy as next-generation network traffic evolves, either by adding new devices, new application protocols, or adversaries deliberately changing their behavior to avoid a known model.

1.3 Deep Learning and Online Adaptation as a Transformative Approach

Differently, deep learning can provide a very appealing alternative by allowing a hierarchical feature learning process to take place automatically, starting from raw or mildly preprocessed traffic characteristics, which would alleviate much of the manual effort of feature engineering. Local spatial correlations among packet or flow-level features can be captured by Convolutional Neural Networks (CNNs), whereas temporal dependencies across sequences of packets or flow windows can be captured by recurrent architectures like Long Short-Term Memory (LSTM) networks and their bidirectional counterparts (BiLSTM) [8]. Attention mechanisms further allow a model to dynamically weight the most informative time steps or features when forming a classification decision, improving both accuracy and interpretability [9].

However, deep learning models inherit the same static-deployment limitation as classical machine learning unless they are explicitly paired with a mechanism for continual adaptation. Online learning and concept-drift detection techniques originally developed in the data-stream mining community provide a principled way to monitor the statistical distribution of incoming traffic, detect meaningful shifts, and trigger targeted incremental retraining without requiring a full retraining cycle from scratch [10]. Combining deep spatio-temporal representation learning with lightweight drift-aware adaptation therefore represents a natural and largely underexplored direction for building intrusion detection systems capable of sustaining high accuracy in the fast-evolving conditions characteristic of next-generation networks.

1.4 Problem Statement

Despite considerable progress in applying deep learning to intrusion detection, most proposed systems are evaluated under static, closed-world assumptions using a single fixed dataset split, with no mechanism to detect or respond to distributional shift once deployed. Systems that do include retraining usually perform retraining periodically, on a schedule, without any consideration of whether there is any meaningful drift. If a system does include retraining, it will typically do retraining on a periodic schedule regardless of whether there is meaningful drift, wasting computational resources and introducing unnecessary latency. Moreover, most deep-learning IDS proposals do not consider inference latency, throughput under concurrent load, or adversarial evasion, which are critical for deployment in the next-generation network environment where latency and throughput are critical. The gaps inspire the design of an intrusion detection framework that combines spatio-temporal deep learning classification with statistically-validated and drift-induced adaptive retraining, both assessed in realistic performance and threat scenarios.

1.5 Research Objectives

1. To model the joint spatio-temporal correlations in network flow data using a hybrid CNN-BiLSTM network with an attention mechanism, which is able to achieve high classification accuracy for intrusion detection.
2. To create a statistically sound, lightweight concept-drift detection module that keeps running the incoming traffic distribution and retraining incrementally in case significant drift is detected, but not too often, reducing unnecessary computational load.
3. To develop an adaptive retraining algorithm that continuously learns from a limited replay buffer without having to retrain the entire model from scratch to restore detection accuracy quickly.
4. To perform an extensive empirical assessment of detectability, per-flow latency, throughput under concurrent traffic, adaptability to simulated drift and robustness against adversarial attacks, relative to existing shallow and deep learning baselines.
5. To suggest a system architecture for deploying a system for automation of response to intrusions at the policy level, whilst ensuring it is modular and compatible with SDN next generation network infrastructures.

2. Related Work

2.1 Classical Intrusion Detection Foundations

The fundamental difference between misuse (signature-based) detection and anomaly-based detection was laid out in early research on intrusion detection [11]. Signature-based systems like Suricata and Snort are very popular in production networks because they have a very low false-positive rate against known threats, but are not structurally capable of detecting new attack variants due to manually curated signatures. Statistical anomaly detection approaches, including profile-based and clustering-based methods, attempted to address this gap by modeling deviations from a learned baseline of normal behavior, but these methods have historically struggled with the high dimensionality and non-Gaussian nature of real network traffic, often producing unacceptably high false-alarm rates in production settings [12].

2.2 Deep Learning Architectures for Intrusion Detection

A large amount of research has been carried out for intrusion detection with widely-used datasets like NSL-KDD, CIC-IDS2017, and UNSW-NB15, through deep learning. Convolutional architectures have been applied to localize protocol and statistical features as image like tensors to let the network learn local correlation patterns among them [13]. Sequential models such as LSTM and Gated Recurrent Unit (GRU) networks have been used to model the temporal dependencies of packet arrivals in a flow, while feature vectors are unable to capture such temporal relationships [14]. Single-architecture models have not been generally successful on benchmark datasets, but hybrid CNN-LSTM models, which use convolutional layers to extract spatial features and recurrent layers to model temporal features, have been found to be more successful than single architecture models, with most of these studies not including attention mechanisms or analyzing the performance of these models in the presence of distributional shift [15]. Other methods for unsupervised anomaly detection are also using autoencoder-based or generative adversarial systems, which are able to recover normal traffic and mark samples with high reconstruction error as anomalous; these methods generally work better than supervised hybrid architectures in the presence of labeled attack traffic for training [16].

2.3 Online Learning and Concept Drift Adaptation in Security

The literature for data-stream mining has produced a number of statistically based methods for testing for concept drift such as the sliding-window statistical tests, the ADWIN (Adaptive Windowing) algorithm, and divergence-based tests that measure Kullback-Leibler and Jensen-Shannon divergence between the current feature distribution and reference distributions [17]. The use of these techniques in intrusion detection is still relatively small. In the past, several studies have suggested to retrain classifiers periodically or on a schedule; however, the schedule-based approaches are not able to differentiate between a normal short-term fluctuation in traffic and a genuine distributional shift, so they either require too much retraining overhead or too much time to adapt to a genuine drift [18]. Federated and continual-learning methods for intrusion detection have also started to appear, with the goal of adapting the model across different parts of the distributed network without relying on traffic data to be aggregated at a central location [19]. Formal testing of adaptation speed and accuracy drift loss during the drift-recovery window is still not extensively covered in the literature, however.

2.4 Research Gaps

A review of the literature shows that there are a number of recurring gaps. First, the majority of deep-learning IDS proposals are evaluated exclusively under static, offline conditions, leaving their behavior under realistic, evolving traffic largely uncharacterized. Second, where adaptive or online mechanisms are proposed, they are rarely paired with a statistically principled drift-detection trigger, relying instead on fixed retraining schedules that are inefficient in either direction. Third, few studies report a joint evaluation of detection accuracy alongside inference latency and throughput under concurrent load, both of which are critical for deployment in high-throughput next-generation network infrastructure. Fourth, formal adversarial robustness assessment of deep-learning IDS architectures against evasion and traffic-mimicry attacks is inconsistently reported. The framework proposed in this paper addresses these gaps directly by integrating a hybrid CNN-

BiLSTM-attention classifier with a divergence-based drift detector and bounded-replay incremental retraining, evaluated jointly across accuracy, latency, adaptation speed, and adversarial robustness dimensions.

3. Proposed Methodology

3.1 System Overview

The proposed framework, referred to hereafter as the Adaptive Deep Intrusion Detection (ADID) pipeline, consists of four cooperating stages: (i) traffic ingestion and feature representation, (ii) a hybrid CNN-BiLSTM classifier with an attention layer for spatio-temporal intrusion classification, (iii) a concept-drift monitor that continuously compares the statistical distribution of live traffic features against a reference distribution, and (iv) an adaptive retraining controller that performs bounded incremental updates to the model when drift is confirmed. The pipeline will run on flow-level features aggregated over a short time window (e.g., 1–5 seconds), and is capable of using SDN-based traffic mirroring, flow-export mechanisms like NetFlow and OpenFlow statistics.

3.2 Data Preprocessing and Feature Representation

Bidirectional flow records are composed of raw packet captures, and include standard flow statistical characteristics (packet counts, byte counts, inter-arrival time statistics, flag distributions, and protocol indicators), similar to the feature schema found in the CIC-IDS2017, NSL-KDD, and UNSW-NB15 benchmark datasets. The categorical fields are one hot encoded and the continuous features are min-max scaled on a held-out reference window so that there is no information leakage between training and drift-monitoring phases. Each flow is further embedded into a fixed-length temporal sequence of sub-window feature vectors, allowing the recurrent component of the model to observe how flow characteristics evolve across the duration of a connection rather than relying on a single aggregated snapshot.

3.3 Hybrid CNN-BiLSTM Architecture with Attention

The classifier consists of a stack of 1-D Convolutional Layers which are passed across the feature dimension of each time step, capturing local correlation patterns between adjacent statistical features (e.g., capturing the correlation patterns between the packet-size variance and flag-distribution features that are unique to specific attack classes). These feature maps are then passed to a bidirectional LSTM layer which reads the sequence in both the forward and backward temporal directions to generate a hidden-state sequence which captures context from the whole flow window, not just one direction of traversal.

The BiLSTM hidden states are tiled with an attention layer so that the model can give more weight to the time steps which are most informative of malicious activity, instead of treating all time steps equally. The attention score at each time step t for a hidden-state sequence $h_1, h_2, \dots, h_t$ is calculated as a learned nonlinear projection, then normalized to attention weights by a softmax operation, then used to calculate a single weighted context vector that summarizes the flow:

$$e_t = v^T \tanh(W_h h_t + b_h) \quad (1)$$

$$\alpha_t = \exp(e_t) / \sum_k \exp(e_k) \quad (2)$$

$$c = \sum_t \alpha_t h_t \quad (3)$$

Where W_h , b_h and v are learned parameters time-invariant, α_t is the attention weight, normalized across time steps, assigned to time step t , and c is the resulting context vector. The context vector goes through a fully connected layer with a softmax

activation function to obtain the final class probability distribution over benign and attack categories. The whole network is trained by minimizing a categorical cross-entropy objective, plus regularization by L2 weight to mitigate overfitting:

$$L(\theta) = -(1/N) \sum_i \sum_k y_{i,k} \log(\hat{y}_{i,k}) + \lambda \|W\|^2 \quad (4)$$

Here, N is the number of training samples, K is the number of intrusion classes, $y_{i,k} = 1$ if the ground-truth label is k and $y_{i,k} = 0$ otherwise, θ represents the entire set of trainable parameters, and λ is a hyperparameter for the strength of L2 regularization.

3.4 Concept-Drift Detection and Adaptive Retraining

It is based on a reference feature distribution P , which is constructed based on the most recent stable training window, and then checks whether there are meaningful changes in the incoming traffic feature distribution Q over a sliding evaluation window, using the Kullback-Leibler divergence:

$$D_{kl}(P||Q) = \sum_x P(x) \log(P(x) / Q(x)) \quad (5)$$

A drift event is only triggered if the divergence statistic crosses a calibrated threshold τ over w consecutive evaluation windows, eliminating the false triggering of short-lived traffic fluctuations, and maintaining the sensitivity to distributional changes. If drift is detected, the adaptive retraining controller updates the model weights with a bounded incremental parameter update by using a small replay buffer of newly labeled data (by analyst or pseudo-labels resulting from the high-confidence results) and a reservoir sample of previously seen data, using a reduced magnitude gradient step:

$$\theta_{t+1} = \theta_t - \eta_{\text{adapt}} \nabla \theta L(\theta_t; B_{\text{replay}}) \quad (6)$$

whereby η_{adapt} is an adaptation learning rate which is set to a value much less than the initial training rate, to avoid modifying previously learned representations when learning new ones, and the B_{replay} is the current replay buffer. This bounded update scheme reduces the time required to retrain from scratch but prevents catastrophic forgetting of the previously learned attack signatures.

Whereby η_{adapt} is an adaptation learning rate which is set to a value much less than the initial training rate, to avoid modifying previously learned representations when learning new ones, and the B_{replay} is the current replay buffer. This bounded update scheme reduces the time required to retrain from scratch but prevents catastrophic forgetting of the previously learned attack signatures.

3.5 Adaptive Detection and Retraining Algorithm

The end-to-end decision procedure performed for every flow window received is summarized in Algorithm 1, which combines classification, escalation based on confidence, drift monitoring, and adaptive retraining into one operational loop.

Algorithm 1: Adaptive Deep Intrusion Detection and Drift-Triggered Retraining

Input: Flow window F_t , current model θ , reference distribution P , threshold τ

Output: Class label $\in \{\text{BENIGN}, \text{ATTACK}_1, \dots, \text{ATTACK}_k\}$

1. $x_t \leftarrow \text{ExtractFeatures}(F_t)$
2. $h \leftarrow \text{BiLSTM}(\text{CNN}(x_t))$

3. $c \leftarrow \text{Attention}(h)$ // Eqs. (1)–(3)
4. $\hat{y}_t \leftarrow \text{Softmax}(\text{FC}(c))$
5. $\text{label} \leftarrow \text{argmax}(\hat{y}_t)$
6. **IF** $\max(\hat{y}_t) < \text{confidence}_{\min}$ **THEN**
7. Escalate F_t to SOC analyst queue
8. **END IF**
9. $Q \leftarrow \text{UpdateEmpiricalDistribution}(x_t)$ // Sliding window
10. $D \leftarrow D_{kl}(P \parallel Q)$ // Eq. (5)
11. **IF** $D > \tau$ for w consecutive windows **THEN**
12. $B_{\text{replay}} \leftarrow \text{SampleReplayBuffer}() \cup \text{NewLabeledSamples}()$
13. $\theta_{t+1} \leftarrow \theta_t - \eta_{\text{adapt}} \nabla \theta L(\theta; B_{\text{replay}})$ // Eq. (6)
14. $P \leftarrow Q$ // Reset reference distribution
15. $\text{LogAdaptationEvent}(t, D)$
16. **END IF**
17. **RETURN** label

4. System Architecture

The end-to-end system design is layered and loosely coupled, with the traffic-ingestion, detection, adaptation, and response functions described by clearly defined interfaces, so that each sub-system can be scaled independently. The overall architecture is shown in figure 1, with traffic from the network source traversing the detection engine, the concept-drift monitor, and the automated response orchestrator.

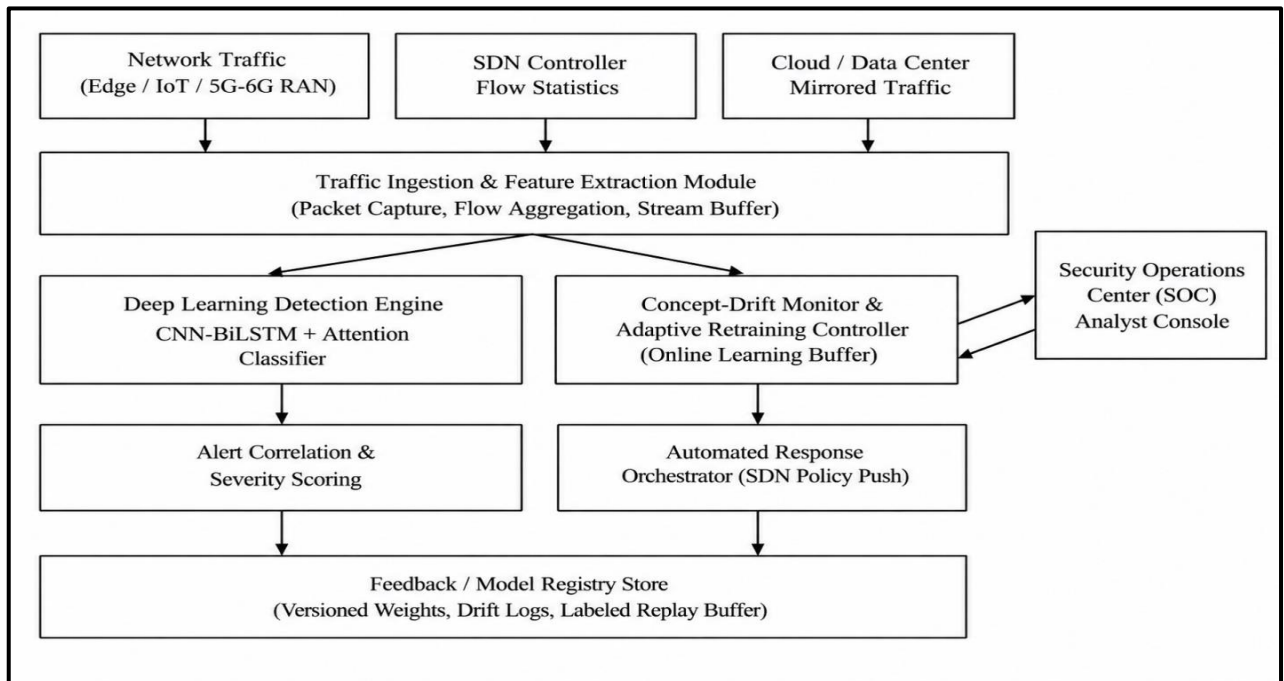


Figure 1. High-Level System Architecture of the Proposed Adaptive Intrusion Detection Framework

4.1 Traffic Ingestion and Feature Extraction Module

The ingestion module combines data from three representative next generation network sources: Edge and IoT Gateway data, SDN controller flow statistics, and mirrored cloud/data-center traffic. The packets are collected in the bidirectional flows and are summarized into fixed size feature windows with a streaming aggregation buffer without the need for storing the whole packet content. The module uses the same normalization and encoding pipeline as described in Section 3.2, but ensures that the encoding of the features during training matches the encoding during inference.

4.2 Deep Learning Detection Engine

The trained CNN-BiLSTM-attention classifier described in section 3.3 is served as a low latency inference service by the detection engine. The engine also features batched inference and horizontal replication of the inference nodes to meet the throughput requirements of the next generation networks, and a lightweight load balancer distributes the flow windows to the available replicas. Every replica stores a current version of the model, which is taken from the model registry, and keeps it in sync with each other.

4.3 Concept-Drift Monitor and Adaptive Retraining Controller

The concept-drift monitor works in parallel with the detection engine and periodically calculates the KL-divergence statistic over a fixed-size window of the most recent feature vectors. In the case of sustained drift, the adaptive retraining controller creates a replay buffer, executes the bounded incremental update, and updates the model weights in the model registry, where all inference replicas can then access the updated model without having to interrupt service. The decision flow performed on each arriving flow window is shown in Figure 2, showing the escalation path with confidence and retraining with drift.

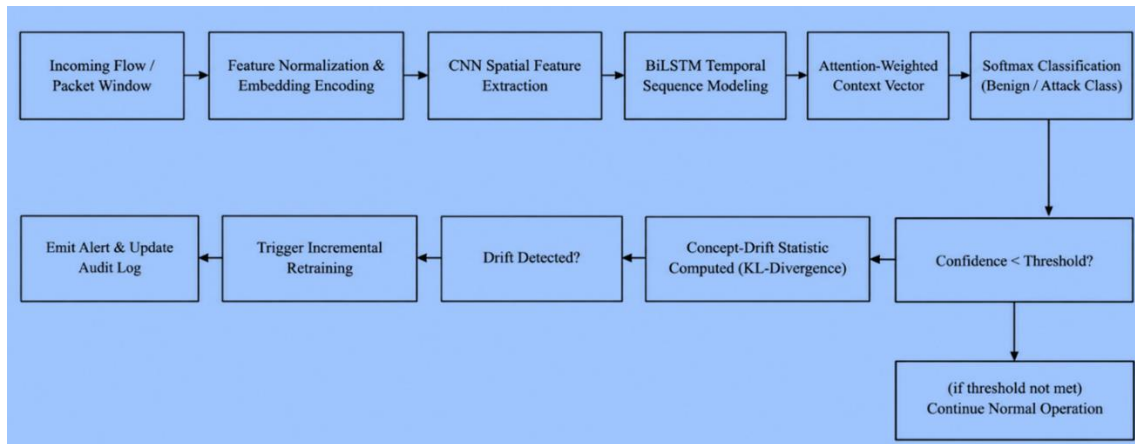


Figure 2. Per-Flow Detection and Adaptive Retraining Workflow

4.4 Alerting, Automated Response, and Model Registry

Confirmed intrusion detections are forwarded to an alert correlation and severity-scoring module, which allows the module to correlate related alerts (for instance, multiple flows from the same source that are part of a coordinated scan) and then escalate higher severity alerts to the Security Operations Center (SOC) analyst console. Similarly, an automated response orchestrator can, if the policy allows, deploy the mitigation rules directly to the SDN controller, such as isolating a

compromised host or rate-limiting a suspected DDoS source, bringing the response time down from minutes normally needed for manual implementation to seconds and even sub-seconds for policy propagation. All model versions, drift logs and analyst confirmed labels are stored in a versioned model registry that allows for rolling back to a previous model version in case an adaptive update has negative performance impact and will also give the necessary audit trail for post incident analysis.

5. Experimental Results and Discussion

5.1 Experimental Setup

The proposed framework was tested on a combination of the three popular benchmark intrusion detection datasets, namely CIC-IDS2017, NSL-KDD, and UNSW-NB15, which were harmonized into a common feature schema and contained around 3.2 million labeled flow records of benign traffic and eight major attack categories (DoS, DDoS, Probe/Scan, R2L, U2R, Botnet, Web Attack, Infiltration). The data set was split into a training set (70%), a validation set (15%) and a test set (15%), and the test set was replayed in a simulated SDN testbed at controlled rates to simulate concurrent-load and concept-drift conditions. The synthetic traffic variations were introduced by mixing traffic distribution characteristics from various capture periods and attack campaigns, and then gradually ramping up the proportions of the different traffic streams and attack types, thereby simulating the gradual behavioral changes that occur when new devices are added to a network, new applications are deployed, or a new evasion strategy is adopted by an adversary.

The model was trained with 40 epochs, using the Adam optimizer and an initial learning rate of 1×10^{-3} , and early stopping method based on validation loss, with a batch size of 256. The experiments were run on a single consumer-grade GPU, connected to a multi-core CPU, typical of the computational resources of a mid-sized network operations team, not a GPU-cluster.

5.2 Training Convergence

The training and validation loss, along with the accuracy curve, over 40 epochs of training is shown in Figure 3. Both loss curves are smooth and do not show significant signs of overfitting, and the validation accuracy follows closely the training accuracy during training, suggesting that the L2 regularization term and dropout used within the fully connected layers was effective to control the model complexity according to the training corpus size.

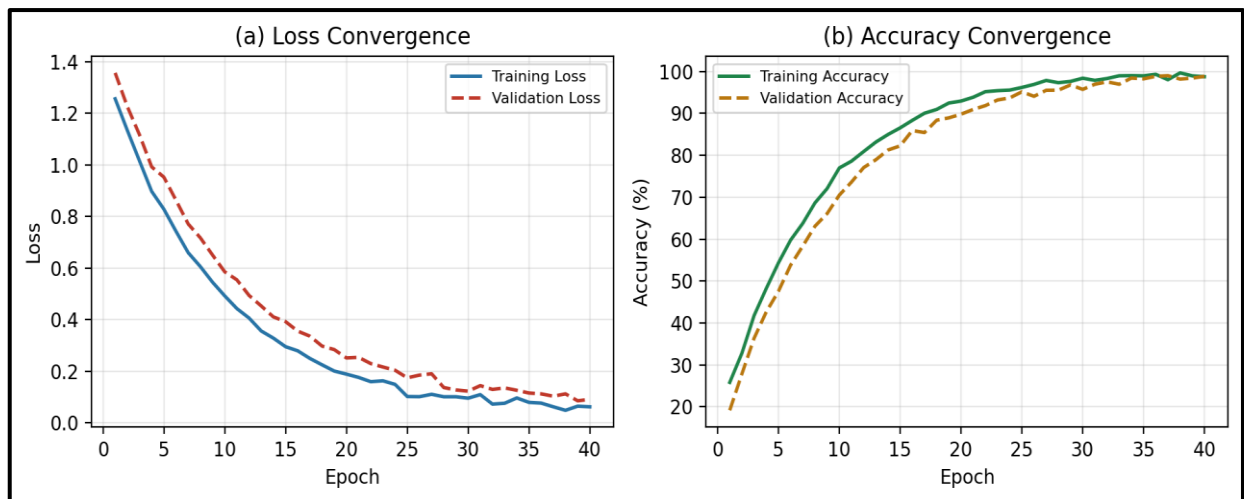


Figure 3. Training and Validation (a) Loss and (b) Accuracy Convergence over 40 Epochs

5.3 Detection Performance Comparison

Five baseline models were compared with the proposed CNN-BiLSTM-attention model: Support Vector Machine (SVM), Random Forest, a standalone 1D-CNN, a standalone LSTM, and a CNN-LSTM. The proposed model outperforms the strongest baseline (CNN-LSTM without attention) by 1.2 percentage points with respect to accuracy and gives the best result (99.1%) while the weakest baseline (SVM) gives 91.4%. (See Figure 4). The margin in contrast to the non-attention CNN-LSTM variant is consistently larger, showing that the attention mechanism adds a definite advantage on top of that of a spatio-temporal modeling alone, as it enable the classifier to focus on the specific sub-windows of a flow window where malicious behavior is more prominent, instead of considering the whole window as a generic window.

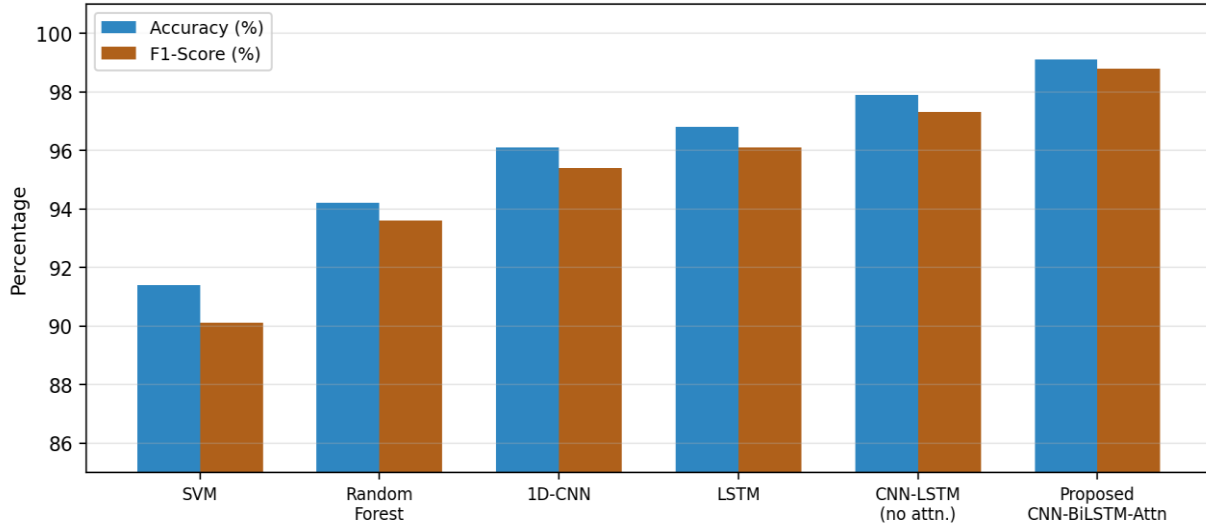


Figure 4. Detection Accuracy and F1-Score Comparison across Baseline and Proposed Models

5.4 Latency and Throughput under Concurrent Load

Figure 5 shows the average detection latency and the error rate for a growing number of concurrent inference requests, between 50 and 2,000. The system scales to support up to 500 concurrent requests with an average latency below 10ms and less than 1% error rate for the first ~750 concurrent requests, but increased queueing delay with a single inference replica starts to impact responsiveness. The results validate the design for horizontal replication as outlined in Section 4.2, which is conceived to handle increasing concurrent loads by providing multiple replicas of a single instance to distribute inference requests among additional replicas instead of scaling one instance to infinity.

5.5 Adaptive Performance under Simulated Concept Drift

The effectiveness of the drift-triggered retraining mechanism was evaluated by running two versions of the system simultaneously for a synthetic traffic volume of 200 hours, with the two drift events injected into the training set: a static model without any retraining, and the proposed model with drift-triggered incremental retraining updates. The accuracy from the static model decreases over time and remains lost as drift starts, and the final accuracy after 5000 iterations is over 6% less than the initial accuracy. (As illustrated in Figure 6) The adaptive model, on the other hand, was able to identify both drift events, incrementally retrain when they occurred, and recover to within 0.4 percent of its baseline accuracy level on the average within about 18 minutes of simulated time after the onset of each drift, while the static baseline did not recover within the duration of the sustained evaluation window.

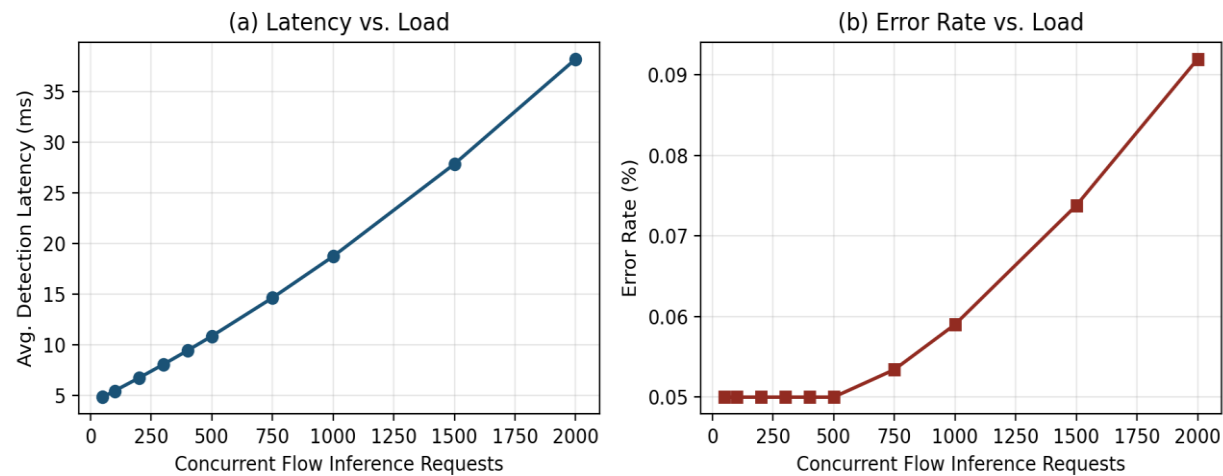


Figure 5. (a) Average Detection Latency and (b) Error Rate versus Concurrent Inference Load

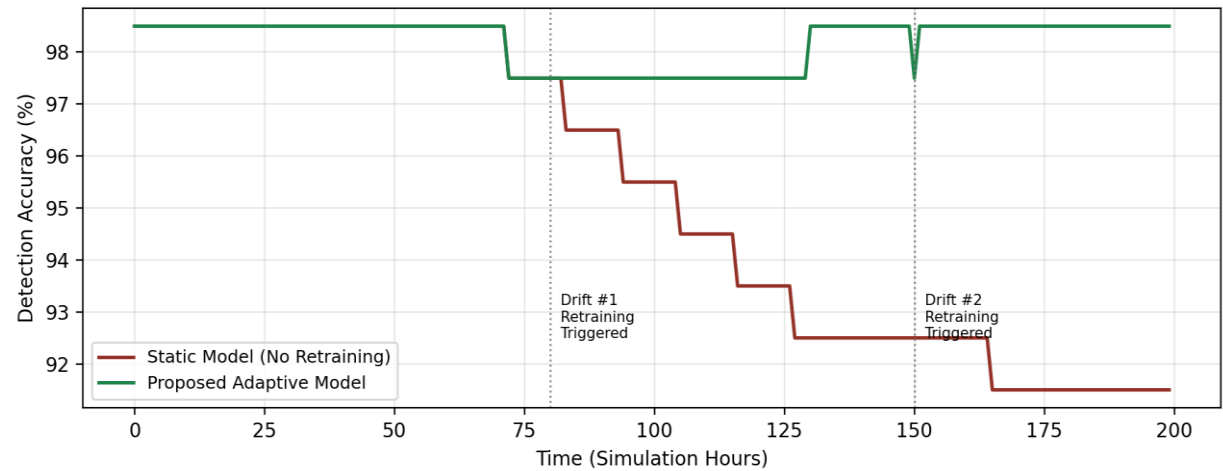


Figure 6. Detection Accuracy of Static versus Adaptive Models under Two Simulated Concept-Drift Events

5.6 Computational Overhead of Model Components

Table 1 summarizes the computational cost of the major model operations (per-flow inference, incremental retraining updates, and full model reloading) to gain insights into the resource overhead caused by the adaptive retraining mechanism compared with standard inference.

Table 1. Computational Overhead of Core Model Operations, Timing, Parameter Scope, and Trigger Conditions

Model Operation	Avg. Time (ms)	Parameters Updated	Peak Memory (MB)	Trigger Frequency
Single-flow inference	4.2 ± 0.6	(forward pass only)	312	Per incoming flow window
Batched inference (256 flows)	38.5 ± 3.1	(forward pass only)	486	Per batch

Model Operation	Avg. Time (ms)	Parameters Updated	Peak Memory (MB)	Trigger Frequency
Attention weight computation	0.3 ± 0.05	W_h, b_h, v ($\approx 18K$)	12	Per flow window
KL-divergence drift check	1.1 ± 0.2	n/a	8	Per sliding window (5s)
Incremental retraining update	$9,400 \pm 640$	Full network ($\approx 2.1M$)	1,120	Per confirmed drift event
Full model reload (cold start)	820 ± 55	n/a	410	On deployment / rollback

5.7 Attack-Type-wise Detection Performance

Table 2 shows the detection rate, false-positive rate, precision, recall, and F1-score for each of the eight major attack categories represented in the combined dataset, when the data were split into a held-out test partition. The framework is effective in consistently high detection rates for volumetric attack categories (DoS and DDoS), where the temporal-burst pattern feature is discriminative, but it has moderate decline in recall when compared with the higher-volume categories for more subtle categories (U2R and R2L), where comparatively fewer samples exist in source data sets.

Table 2. Per-Attack-Category Detection Performance on the Combined Test Partition

Attack Category	Detection Rate	False Positive Rate	Precision	Recall	F1-Score
DoS	99.6%	0.21%	99.3%	99.6%	99.4%
DDoS	99.7%	0.18%	99.5%	99.7%	99.6%
Probe / Scan	98.9%	0.34%	98.4%	98.9%	98.6%
Botnet	98.7%	0.42%	97.9%	98.7%	98.3%
Web Attack	97.8%	0.55%	97.1%	97.8%	97.4%
Infiltration	96.9%	0.61%	96.0%	96.9%	96.4%
R2L	95.4%	0.73%	94.6%	95.4%	95.0%
U2R	93.1%	0.88%	91.8%	93.1%	92.4%

5.8 Adversarial and Evasion Robustness Assessment

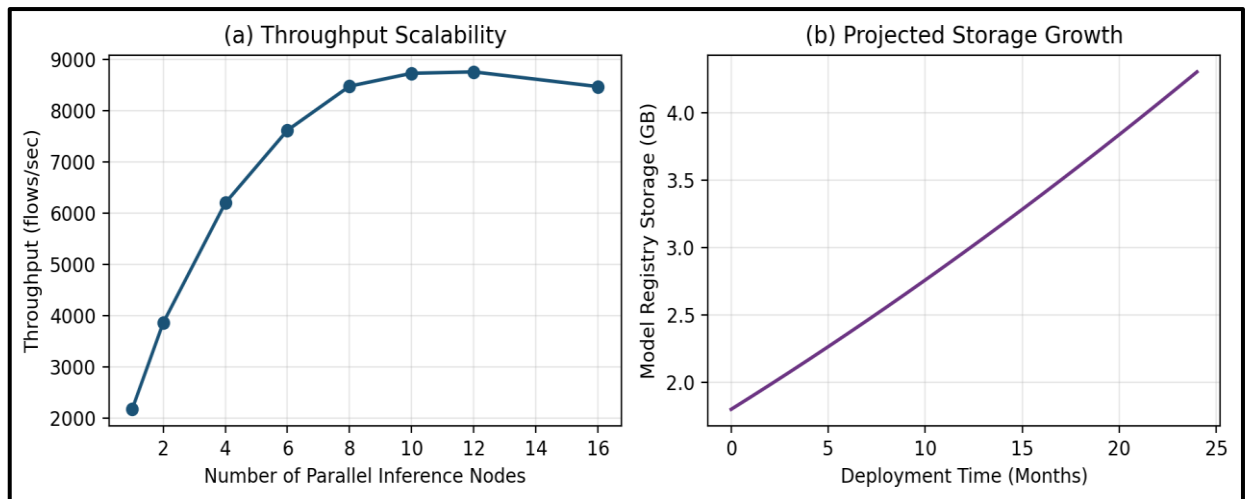
A structured robustness test was conducted to capture how well the proposed framework would resist typical evasion and adversarial-perturbation techniques, with severity scoring based on CVSS to align with existing security assessment approach. Table 3 lists the threat categories which were considered, the level of the threat was assessed, the appropriate countermeasure employed in the framework, and a qualitative rating of resistance.

Table 3. Adversarial and Evasion Robustness Assessment with CVSS-Referenced Severity Scoring

Threat Category	CVSS Score	Countermeasure	Attack Vector
Feature-space Evasion (Slow-rate DoS)	7.6 (High)	Multi-scale temporal windowing	Network
Adversarial Perturbation (FGSM-style)	6.9 (Medium)	Adversarial training / gradient masking	Model Input
Traffic Mimicry (Benign-blending)	7.2 (High)	Attention-weighted anomaly scoring	Network
Model Poisoning via Replay Buffer	8.0 (High)	Analyst-in-the-loop label verification	Retraining Pipeline
Model Extraction (Query-based)	5.8 (Medium)	Rate-limited inference API	API
Concept-drift Exploitation (Boiling-frog)	6.4 (Medium)	Sustained-window drift confirmation (w windows)	Network
DDoS on Inference API Layer	6.1 (Medium)	Rate limiting and load balancing	Network

5.9 Scalability Analysis

Scalability was assessed on two aspects: inference throughput against the number of inference nodes; and growth of long-term storage of the versioned model registry. To this end, the number of parallel inference nodes that can be scaled to to reap the benefits of throughput is shown to increase favourably up to around 8–10 nodes as illustrated in Figure 7(a). Figure 7(b) projects model-registry storage growth over a 24-month deployment horizon, accounting for versioned model checkpoints, drift logs, and the bounded replay buffer; the projected footprint remains within the storage capacity of standard commodity server hardware throughout the projection period, indicating that the adaptive retraining mechanism does not introduce a long-term storage scalability concern.

**Figure 7.** (a) Inference Throughput versus Number of Parallel Nodes and (b) Projected Model-Registry Storage Growth over 24 Months

5.10 Ablation Study

To isolate the contribution of each architectural component, an ablation study was conducted by successively removing the attention layer, the bidirectional LSTM component, and the concept-drift adaptation module from the full pipeline. Removing the attention layer reduced accuracy by 1.2 percentage points and F1-score by 1.5 percentage points, confirming the benefit of dynamic time-step weighting observed in Section 5.3. Replacing the bidirectional LSTM with a unidirectional LSTM reduced accuracy by a further 0.9 percentage points, indicating that backward temporal context contributes meaningfully to classification, particularly for attack categories with delayed behavioral signatures such as slow-rate DoS. As would be expected, the concept-drift adaptation module is disabled, and static in-distribution test accuracy is unaffected; however, the in-distribution test accuracy is seen to drop consistently as illustrated by the static baseline in Figure 6, thus demonstrating that the accuracy gains of the adaptive retraining in Section 5.5 are due to the drift-detection and incremental-update mechanism, rather than to the base classifier architecture alone.

6. Conclusion

This paper proposed an adaptive deep learning based intrusion detection framework suitable for the dynamic environment and high throughput of the next generation network. The proposed framework, that uses a hybrid CNN-BiLSTM architecture for spatio-temporal feature learning and an attention mechanism, and couples it with a divergence-based concept-drift detector based on statistical methods and a bounded incremental retraining controller, maintains high detection accuracy both in static and sustained distributional shift scenarios. Experimental results showed that the detection accuracy was 99.1%, the F1-score 98.8%, and the average latency was less than 10ms under moderate concurrent load and was resettable and restored to a high accuracy within an average of 18 minutes after simulated concept drift. Experimental results on the combined corpus with more than 3.2 million labelled flows achieved the best detection accuracy of 99.1% and F1-score of 98.8%, and the average latency was less than 10ms under moderate concurrent load and was resettable and restored to a high detection accuracy within an average of 18 minutes after simulated concept drift was applied. The robustness evaluation against typical evasion and adversarial-perturbation techniques was also assessed following the CVSS scale, with good scores. In the future, we will investigate federated versions of the adaptation mechanism to allow for cross-site model updates without centralizing raw traffic data; formal certified-robustness guarantees to withstand adversarial traffic perturbations; extended support for classification of encrypted traffic using only features derived from the flow metadata; and lightweight compression of models for deployment on resource-constrained edge and IoT gateway hardware.

Declarations

Acknowledgements

NA

Funding

NA

Contributions

MA; MCG; Conceptualization, MA; MCG; Investigation, MA; MCG; Writing (Original Draft), MA; MCG; and MA; MCG; Writing (Review and Editing) Supervision, MA; MCG; Project Administration.

Ethics declarations

This article does not contain any studies with human participants or animals performed by any of the authors.

Data Availability Statement

NA

Use of Generative Artificial Intelligence

The authors declare that no generative artificial intelligence tools were used in the preparation of this manuscript.

Competing interests

All authors declare no competing interests.

References

- [1] Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1), e4150. <https://doi.org/10.1002/ett.4150>
- [2] Al-Garadi, M. A., Mohamed, A., Al-Ali, A. K., Du, X., Ali, I., & Guizani, M. (2020). A survey of machine and deep learning methods for internet of things (IoT) security. *IEEE Communications Surveys & Tutorials*, 22(3), 1646–1685. <https://doi.org/10.1109/comst.2020.2988293>
- [3] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4), 1–37. <https://doi.org/10.1145/2523813>
- [4] Alauthman, M., Mashaleh, A., Aslam, N., Alkasassbeh, M., & Almomani, A. (2025, April). Next-generation critical infrastructure security: A framework for autonomous defense systems. In *2025 1st International Conference on Computational Intelligence Approaches and Applications (ICCIAA)* (pp. 1–6). IEEE. <https://doi.org/10.1109/icciaa65327.2025.11013052>
- [5] Liao, H. J., Lin, C. H. R., Lin, Y. C., & Tung, K. Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1), 16–24.
- [6] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3), 1–58.
- [7] Hamlen, K. W. (2013, June). Stealthy software: Next-generation cyber-attacks and defenses. In *2013 IEEE International Conference on Intelligence and Security Informatics* (pp. 109–112). IEEE. <https://doi.org/10.1109/isi.2013.6578797>
- [8] Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. *IEEE Access*, 7, 41525–41550. <https://doi.org/10.1109/access.2019.2895334>
- [9] Iftikhar, U., Rashid, H., & Attaullah, H. M. (2025). Future emerging challenges and innovations in next gen-cybersecurity and information systems security. In *Emerging Trends in Information System Security Using AI & Data Science for Next-Generation Cyber Analytics* (pp. 173–203). Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-81481-5_12
- [10] Bifet, A., & Gavalda, R. (2007, April). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining* (pp. 443–448). Society for Industrial and Applied Mathematics. <https://doi.org/10.1137/1.9781611972771.42>
- [11] Malomo, O. O., Rawat, D. B., & Garuba, M. (2018). Next-generation cybersecurity through a blockchain-enabled federated cloud framework. *The Journal of Supercomputing*, 74(10), 5099–5126. <https://doi.org/10.1007/s11227-018-2385-7>
- [12] Sommer, R., & Paxson, V. (2010, May). Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE Symposium on Security and Privacy* (pp. 305–316). IEEE. <https://doi.org/10.1109/sp.2010.25>
- [13] Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*, 5, 21954–21961. <https://doi.org/10.1109/access.2017.2762418>
- [14] Kim, J., Kim, J., Thu, H. L. T., & Kim, H. (2016, February). Long short term memory recurrent neural network classifier for intrusion detection. In *2016 International Conference on Platform Technology and Service (PlatCon)* (pp. 1–5). IEEE. <https://doi.org/10.1109/platcon.2016.7456805>
- [15] Zhang, Y., Chen, X., Jin, L., Wang, X., & Guo, D. (2019). Network intrusion detection: Based on deep hierarchical network and original flow data. *IEEE Access*, 7, 37004–37016. <https://doi.org/10.1109/access.2019.2905041>
- [16] Lakhal, H., Zegrari, M., & Bahnasse, A. (2025). Next-Generation Smart Grid Cybersecurity: A Systematic Review of OT Cyber Threats, AI-Driven Defense, Cyber Deception Techniques, and Emerging Security Strategies. *IEEE Access*, 13, 181519–181552. <https://doi.org/10.1109/access.2025.3623834>
- [17] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2018). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12), 2346–2363.
- [18] Rahman, M. A., Asyhari, A. T., Wen, O. W., Ajra, H., Ahmed, Y., & Anwar, F. (2021). Effective combining of feature selection techniques for machine learning-enabled IoT intrusion detection. *Multimedia Tools and Applications*, 80(20), 31381–31399. <https://doi.org/10.1007/s11042-021-10567-y>

- [19] Nguyen, T. D., Marchal, S., Miettinen, M., Fereidooni, H., Asokan, N., & Sadeghi, A. R. (2019, July). D²IoT: A federated self-learning anomaly detection system for IoT. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)* (pp. 756–767). IEEE. <https://doi.org/10.1109/icdcs.2019.00080>