

A Lightweight Hybrid Convolutional Neural Network–Long Short-Term Memory (CNN–LSTM) and AdaBoost Framework for Edge-Based Internet of Things (IoT) Intrusion Detection

Dev H. Prajapati ¹, Vishvendu Bhatt ²

1. Asha M. Tarsadia Institute of Computer Science and Technology (AMTICS), Uka Tarsadia University, Surat, IND

2. Information Science and Technology, Asha M. Tarsadia Institute of Computer Science and Technology (AMTICS), Uka Tarsadia University, Surat, IND

Received: April 30, 2026 | Review began: May 31, 2026 | Review ended: July 18, 2026 | Published: July 18, 2026

© Copyright 2026

This is an open access article distributed under the terms of the Creative Commons Attribution License CC-BY 4.0., which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Abstract

As Internet of Things (IoT) infrastructure scales, the demand for reliable intrusion detection systems increases. However, modern hybrid deep learning architectures are often too computationally heavy for resource-constrained edge gateways. This study introduces a TinyML-optimized 1D Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) and AdaBoost framework to address these hardware limitations. Using a merged UNSW-NB15 and RT-IoT2022 dataset stratified into a 70/30 normal-to-attack split, the methodology applies Correlation-Based Elimination and Mutual Information to select the top 30 network features (K-30). A custom Focal Loss function handles class imbalance natively without requiring synthetic data overhead. The core CNN-LSTM extracts deep spatiotemporal features, which are then passed to an AdaBoost ensemble. Sigmoid calibration is applied to restrict the False Positive Rate, and the entire pipeline is compressed via TensorFlow Lite dynamic range quantization. Deployed inside a simulated Docker-based edge gateway processing three continuous CCTV video streams, the quantized model occupied just 24 KB of memory. It consumed 23.8% of the host CPU and maintained an inference latency of ~60.16 ms under load. While yielding a raw accuracy of 94.87%, the mathematically calibrated threshold secured a False Positive Rate of 1.94%. These results demonstrate that structural compression and strict feature reduction provide viable, real-time edge security without overloading host hardware.

Categories: IoT Security and Privacy, Deep Learning, Edge Computing

Keywords: intrusion detection, iot security, cnn-lstm model, model compression, feature selection

Introduction

The ongoing integration of the Internet of Things (IoT) has fundamentally changed how devices connect across homes, industrial facilities, and healthcare networks [1-4]. While this connectivity enables unprecedented automation, it simultaneously expands the attack surface, making these infrastructures prime targets for network intrusions [1,3,5]. Early software-centric prediction studies, such as those analyzing bug severity and text-based faults, documented the growing complexity of these threats and underscored the need for adaptable detection systems [6,7]. Consequently, the security field has shifted heavily toward deep learning and hybrid models to identify subtle behavioral patterns that traditional signature-based firewalls routinely miss [8].

How to cite this article:

Prajapati D H, Bhatt V (July 18, 2026) A Lightweight Hybrid Convolutional Neural Network–Long Short-Term Memory (CNN–LSTM) and AdaBoost Framework for Edge-Based Internet of Things (IoT) Intrusion Detection. Cureus J Comput Sci 3 : es44389-026-00170-3. DOI https://doi.org/10.7759/s44389-026-00170-3

Securing IoT environments presents a unique engineering challenge because the devices themselves operate with minimal memory, limited processing power, and diverse communication protocols [1,3,9]. These physical constraints render older approaches, such as Deep Packet Inspection, largely obsolete, especially since modern malicious traffic is typically encrypted or obfuscated [5,8]. Recent literature demonstrates that anomaly-based detection driven by machine learning offers a stronger defense, as it trains systems to recognize normal network behavior rather than relying on known attack signatures [1,10]. This methodology has proven effective across adjacent fields, including industrial fault diagnosis, reinforcing the need for intelligent IDS solutions that do not overwhelm host hardware [9,11].

To process high-volume, encrypted traffic, researchers have found success using hybrid deep learning architectures - specifically combinations of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) networks [7,12,13]. In these setups, CNN layers capture localized spatial patterns within packet flows [7,13], while LSTM components track long-term dependencies to reveal how attack sequences unfold over time [1,13,14]. As these networks grow in complexity, recent studies have begun exploring model compression techniques, such as pruning and quantization, to make them viable for different IoT deployment scenarios [5,15-17]. Collectively, these findings suggest that an optimized CNN-LSTM pipeline is the most practical approach to securing network perimeters with high accuracy and low latency [11,18].

Although recent hybrid frameworks - most notably the baseline CNN-LSTM-AdaBoost architecture proposed by Eguavoen et al. [4] - achieve high detection performance, they are optimized exclusively for high-compute environments. This study introduces a fundamental departure by prioritizing resource-constrained edge deployability. The distinct novelty of this framework lies in the synergetic integration of (1) non-linear feature-fusion mapping through a tightly bounded single-step sequence LSTM branch, and (2) post-training structural integer quantization via TensorFlow Lite. Unlike prior studies that treat compression as an afterthought, this methodology designs the network layers explicitly to maintain a sub-25 KB structural footprint, enabling robust spatiotemporal intrusion detection directly on native edge microcontrollers.

Materials And Methods

Related work

The evolution of IoT network defense has largely been shaped by the widespread adoption of end-to-end encryption. Because inspecting payload contents is no longer an option, current research heavily emphasizes deep learning architectures that can evaluate network metadata to identify malicious activity [1,6].

The Shift to Hybrid Deep Learning

Early machine learning applications in IoT security showed promise in detecting basic anomalies but struggled to follow multi-stage, sequential attack vectors [10]. To address this limitation, developers integrated Recurrent Neural Networks (RNNs) and LSTM layers to evaluate the temporal flow of network traffic [5,8]. The standard approach today merges these recurrent layers with CNNs. In these hybrid models, the CNN handles the extraction of localized spatial features from data packets, while the LSTM tracks the sequence [7,13]. Researchers have deployed this specific CNN-LSTM structure across multiple disciplines, ranging from encrypted traffic classification to hardware fault diagnosis [12,14], validating its strong pattern-recognition capabilities.

The Computational Bottleneck on Edge Devices

Despite their high detection rates, modern hybrid architectures require significant computing power. Standard IoT edge gateways operate under strict memory, CPU, and thermal limitations [3,9]. Running unoptimized deep neural networks on these microcontrollers often leads to system lag or complete hardware failure [11]. Recognizing this, academic focus has shifted toward physical model compression. Studies on structured pruning and weight quantization demonstrate that the memory footprint of temporal networks can be reduced without compromising predictive ability [2,15,16].

How to cite this article:

Prajapati D H, Bhatt V (July 18, 2026) A Lightweight Hybrid Convolutional Neural Network–Long Short-Term Memory (CNN–LSTM) and AdaBoost Framework for Edge-Based Internet of Things (IoT) Intrusion Detection. Cureus J Comput Sci 3 : es44389-026-00170-3. DOI <https://doi.org/10.7759/s44389-026-00170-3>

Furthermore, researchers note that passing raw, unfiltered datasets into these networks creates unnecessary processing loads, making feature selection methods such as PCA or Recursive Feature Elimination a baseline requirement for edge-level deployment [17].

Ensemble Methods and the Hardware Gap

To improve classification confidence against zero-day threats, recent architectures often pass deep neural representations into secondary ensemble classifiers. A highly relevant baseline by Eguavoen et al. [4] utilized a two-stage pipeline in which features extracted by a CNN-LSTM were classified by an AdaBoost ensemble [4]. Tested against the UNSW-NB15 and RT-IoT2022 datasets, this design achieved 99.70% accuracy [4]. Similar configurations incorporating attention mechanisms have also recorded near-perfect metrics in controlled laboratory tests [19-21].

The persistent engineering gap across these advanced implementations is that they prioritize simulated accuracy over physical deployability [22,23]. While architectures such as the CNN-LSTM and AdaBoost pipeline perform exceptionally well on data-center GPUs, they lack the data truncation and structural quantization necessary to function on a standard 5-volt IoT router managing live traffic [24,25]. The methodology presented in this paper directly addresses this issue by optimizing and quantizing the baseline architecture into a lightweight framework capable of surviving edge deployment.

To clearly delineate the structural and environmental differences between existing state-of-the-art frameworks and our proposed method, a comprehensive comparative matrix is provided in Table 1. This table contrasts the proposed network against recent baseline models, outlining the intentional design shift from heavy, data-center architectures toward a physically compressed footprint engineered for edge gateways.

Study & Reference	Core Architecture	Datasets	Model Complexity	Reported Accuracy	Primary Technical Limitation
Standard Attention Models [19-21]	CNN-LSTM + Attention	Various IoT Data	High	99%	High computational latency; lacks structural quantization
Eguavoen et al. [4]	CNN-LSTM + AdaBoost	UNSW-NB15, RT-IoT2022	Massive	99.70%	Prioritizes raw accuracy metrics over physical hardware constraints; no quantization applied
Proposed Framework	Quantized 1D-CNN-LSTM + AdaBoost	UNSW-NB15, RT-IoT2022	Extremely Low (24 KB footprint)	94.87%	Minor reduction in accuracy as an intentional trade-off for edge-native deployment

TABLE 1: Comparative Analysis of Hybrid Intrusion Detection Architectures

CNN, Convolutional Neural Network; LSTM, Long Short-Term Memory; IDS, Intrusion Detection System; KB, Kilobytes

Method

Constructing a reliable anomaly detection system requires training data that mirrors the chaotic nature of an active network. Older security models frequently utilized isolated traffic logs that failed to represent modern, multi-stage intrusions [1,3]. Recent analyses indicate that training models on heavily imbalanced data encourages networks to simply predict the majority class, inflating accuracy scores while ruining real-world reliability [6,19]. To avoid this, the proposed pipeline uses a merged dataset combining UNSW-NB15 (for complex attack vectors) and RT-IoT2022 (for device-level protocols), expanding upon the data foundation seen in recent ensemble studies [4].

The combined records were deliberately stratified into a strict 70/30 ratio of normal to malicious traffic. This proportion forces the algorithm to recognize benign background operations without developing an aggressive bias toward predicting attacks. The 80/20 training and testing partitioning of this dataset is detailed in Table 2.

Traffic Class	Training Set (80%)	Testing Set (20%)	Total Instances	Split Ratio
Normal (0)	68,380	17,095	85,475	70.00%
Attack (1)	29,306	7,327	36,633	30.00%
Total	97,686	24,422	122,108	100.00%

TABLE 2: Dataset Partitioning (80% Training/20% Testing Split)

This table presents the final distribution of network traffic records after merging the UNSW-NB15 and RT-IoT2022 datasets. The data was deliberately stratified to achieve a 80/20 ratio of training and testing split, ensuring the model trained on a realistic representation of network environments.

Data Preprocessing

Before structural feature selection could occur, the raw network telemetry required normalization. Network traffic contains categorical variables (e.g., protocol types, connection states) alongside continuous numerical arrays. The pipeline applied One-Hot Encoding (pd.get_dummies) to translate categorical string data into machine-readable binary columns. Subsequently, continuous numerical inputs underwent standard Z-score normalization (StandardScaler). This critical step ensures that high-magnitude variables, such as total flow byte counts, do not mathematically overshadow smaller, highly sensitive temporal features like inter-packet arrival times during gradient descent.

Feature Selection and Optimization

Feeding raw, 100+ dimensional packet headers into deep neural networks demands continuous memory allocation, which easily triggers thermal throttling on edge devices [8,9]. Consequently, deterministic data optimization must occur before the neural network processes any tensors [17]. This implementation utilizes a strict two-step feature reduction pipeline:

- Correlation-Based Elimination: Initially, Correlation-Based Elimination was applied to the normalized matrix to eradicate multicollinearity. A Pearson correlation matrix was calculated across the entire training set. Any feature exhibiting a correlation coefficient above the strict 0.90 threshold with another variable was automatically dropped, substantially reducing the overall tensor size and preventing redundant data processing.
- Mutual Information Selection: The remaining uncorrelated inputs underwent Mutual Information classification to measure their non-linear dependency with the target threat labels.

$$I(X; Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right) \quad (1)$$

where $I(X; Y)$ represents the mutual information between feature X and target label Y , $p(x, y)$ is their joint probability mass function, and $p(x)$ and $p(y)$ are their respective marginal probabilities.

The system mathematically ranked these inputs and rigidly isolated only the top 30 highest-scoring features. Unlike baseline models utilizing smaller feature arrays [4], expanding the subset to K-30 provides the sequence model with critical spatiotemporal context to detect sophisticated zero-day threats, as detailed in Table 3.

How to cite this article:

Prajapati D H, Bhatt V (July 18, 2026) A Lightweight Hybrid Convolutional Neural Network–Long Short-Term Memory (CNN–LSTM) and AdaBoost Framework for Edge-Based Internet of Things (IoT) Intrusion Detection. Cureus J Comput Sci 3 : es44389-026-00170-3. DOI <https://doi.org/10.7759/s44389-026-00170-3>

Feature Rank	Feature Name	Description/Network Metric
1	sbytes	Total bytes transmitted from source to destination
2	smean	Mean packet size transmitted by the source
3	sttl	Source to destination Time to Live (TTL) value
4	rate	Overall packet transmission rate
5	sinpkt	Source inter-packet arrival time (ms)
6	sload	Source network load (bits per second)
7	ct_srv_dst	Connections containing the same service and destination address
8	spkts	Total packet count from source to destination
9	ct_dst_ltm	Connections of the same destination address in 100 flows
10	ct_srv_src	Connections containing the same service and source address
11	ct_src_ltm	Connections of the same source address in 100 flows
12	ct_dst_src_ltm	Connections of the same source and destination address
13	ct_dst_sport_ltm	Connections of the same destination address and source port
14	dttl	Destination to source Time to Live (TTL) value
15	ct_src_dport_ltm	Connections of the same source address and destination port
16	dmean	Mean packet size transmitted by the destination
17	fwd_header_size_min	Minimum header size in forward packets

How to cite this article:

Prajapati D H, Bhatt V (July 18, 2026) A Lightweight Hybrid Convolutional Neural Network–Long Short-Term Memory (CNN–LSTM) and AdaBoost Framework for Edge-Based Internet of Things (IoT) Intrusion Detection. Cureus J Comput Sci 3 : es44389-026-00170-3. DOI <https://doi.org/10.7759/s44389-026-00170-3>

18	id.orig_p	Originator (source) port number
19	fwd_subflow_pkts	Packet count within forward subflows
20	id.resp_p	Responder (destination) port number
21	fwd_header_size_tot	Total header size in forward packets
22	dinpkt	Destination inter-packet arrival time (ms)
23	dpkts	Total packet count from destination to source
24	flow_pkts_payload.avg	Average payload size across flow packets
25	fwd_pkts_payload.max	Maximum payload size within forward packets
26	fwd_subflow_bytes	Byte count within forward subflows
27	flow_pkts_payload.tot	Total payload size for the entire flow
28	dload	Destination network load (bits per second)
29	down_up_ratio	Ratio of download (reverse) to upload (forward) traffic
30	fwd_init_window_size	Initial TCP window size for forward traffic

TABLE 3: Top 30 Network Features Selected via Mutual Information

This table lists the 30 optimal network features (K-30) retained from the merged UNSW-NB15 and RT-IoT2022 dataset. Feature selection was conducted using a Mutual Information (MI) scoring algorithm to identify the attributes most critical for accurate attack classification. Isolating these specific features reduces the dimensionality of the data, minimizing the computational overhead required for the TinyML-optimized edge deployment.

Sequence Reshaping

Standard machine learning classifiers process 2D matrices, but temporal neural networks require sequential 3D tensors. The final K-30 feature array was structurally reshaped from (Samples, Features) into a 3D time-series format of (Samples, 1, 30). This conversion explicitly frames packet telemetry as a sequence, enabling recurrent layers to process the data over discrete time steps, as detailed in Figure 1.

How to cite this article:

Prajapati D H, Bhatt V (July 18, 2026) A Lightweight Hybrid Convolutional Neural Network–Long Short-Term Memory (CNN–LSTM) and AdaBoost Framework for Edge-Based Internet of Things (IoT) Intrusion Detection. Cureus J Comput Sci 3 : es44389-026-00170-3. DOI <https://doi.org/10.7759/s44389-026-00170-3>

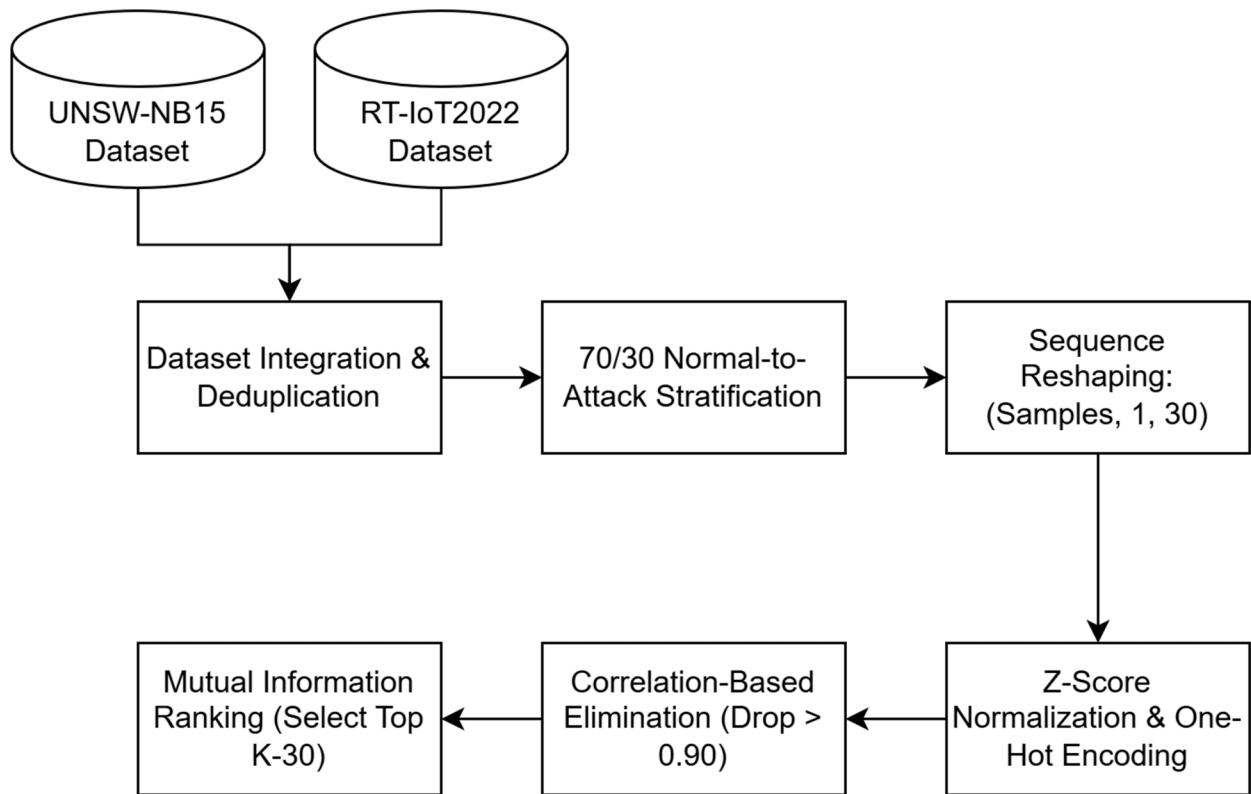


FIGURE 1: Data Pipeline and Reshaping Flowchart

This flowchart illustrates the systematic data preparation pipeline for the hybrid IDS model. The process begins with the integration of the UNSW-NB15 and RT-IoT2022 datasets, followed by deduplication and a 70/30 stratification to maintain class balance. Data is then reshaped for the 1D-CNN input layer and subjected to Z-score normalization and one-hot encoding. The final stage involves a two-step feature reduction process: first removing highly correlated features (threshold > 0.90) and then selecting the top 30 attributes via Mutual Information ranking to ensure computational efficiency on edge hardware.

Hybrid CNN-LSTM architecture

Core Structure

The core structure is a hybrid 1D-CNN-LSTM model configured strictly as a deep spatiotemporal feature extractor rather than a final classifier.

Focal loss integration: Handling the 70/30 dataset split poses a challenge because standard cross-entropy loss algorithms inherently favor the majority class [10,20]. Rather than relying on computationally heavy synthetic data generation like SMOTE, this architecture utilizes a mathematical Focal Loss function to balance the gradient updates during training natively.

$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t) \quad (2)$$

where $FL(p_t)$ represents the Focal Loss, p_t is the model's estimated probability for the true class, α_t is the weighting factor used to handle class imbalance, and γ is the focusing parameter that smoothly down-weights easily classified normal traffic. In this architecture, γ is set to 2.0 and α is set to 0.25. By down-weighting the loss associated with easily classified normal traffic, the network is forced to focus its trainable parameters on harder-to-detect attack signatures.

CNN Branch

How to cite this article:

Prajapati D H, Bhatt V (July 18, 2026) A Lightweight Hybrid Convolutional Neural Network–Long Short-Term Memory (CNN–LSTM) and AdaBoost Framework for Edge-Based Internet of Things (IoT) Intrusion Detection. Cureus J Comput Sci 3 : es44389-026-00170-3. DOI <https://doi.org/10.7759/s44389-026-00170-3>

The reshaped (1, 30) tensor is first passed into a 1-Dimensional Convolutional layer (Conv1D). Configured with 16 filters, a kernel size of 1, and He-Normal weight initialization, this branch maps the localized spatial patterns within the packet flow. A subsequent MaxPooling1D layer reduces the spatial dimensions, outputting a flattened, highly dense spatial feature map.

LSTM Branch

Rather than relying on large sequence windows to track long-term historical network dependencies, the LSTM branch is deliberately bounded to a sequence length of one. This TinyML structural choice prevents memory buffer overflow on constrained edge hardware while still leveraging the layer's internal gating mechanism to dynamically regulate the extracted spatiotemporal features.

In parallel to the CNN branch, the raw sequence is passed directly into an LSTM layer configured with 16 internal units. This layer retains historical packet information through its internal gating mechanisms, tracking the long-term sequential evolution of the network flow [13,14], as detailed in Table 4.

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 1, 30)	0
conv1d (Conv1D)	(None, 1, 16)	496
max_pooling1d (MaxPooling1D)	(None, 1, 16)	0
flatten (Flatten)	(None, 16)	0
lstm (LSTM)	(None, 16)	3,008
concatenate (Concatenate)	(None, 32)	0
dense (Dense)	(None, 32)	1,056
deep_features (Dense)	(None, 16)	528
output (Dense)	(None, 1)	17
Total params:	5,105	
Trainable params:	5,105	

TABLE 4: Functional Parameters Output of the Hybrid CNN-LSTM Model

CNN, Convolutional Neural Network; LSTM, Long Short-Term Memory

Feature Extraction Output Layer

The spatial outputs from the CNN branch and the temporal outputs from the LSTM branch are concatenated into a unified tensor. This tensor passes through an L2-regularized dense layer (32 neurons) before terminating at the final feature extraction layer (16 neurons, ReLU activation). This specific deep_features layer outputs the highly refined

multidimensional array that serves as the final representation of the network traffic, as detailed in Figure 2.

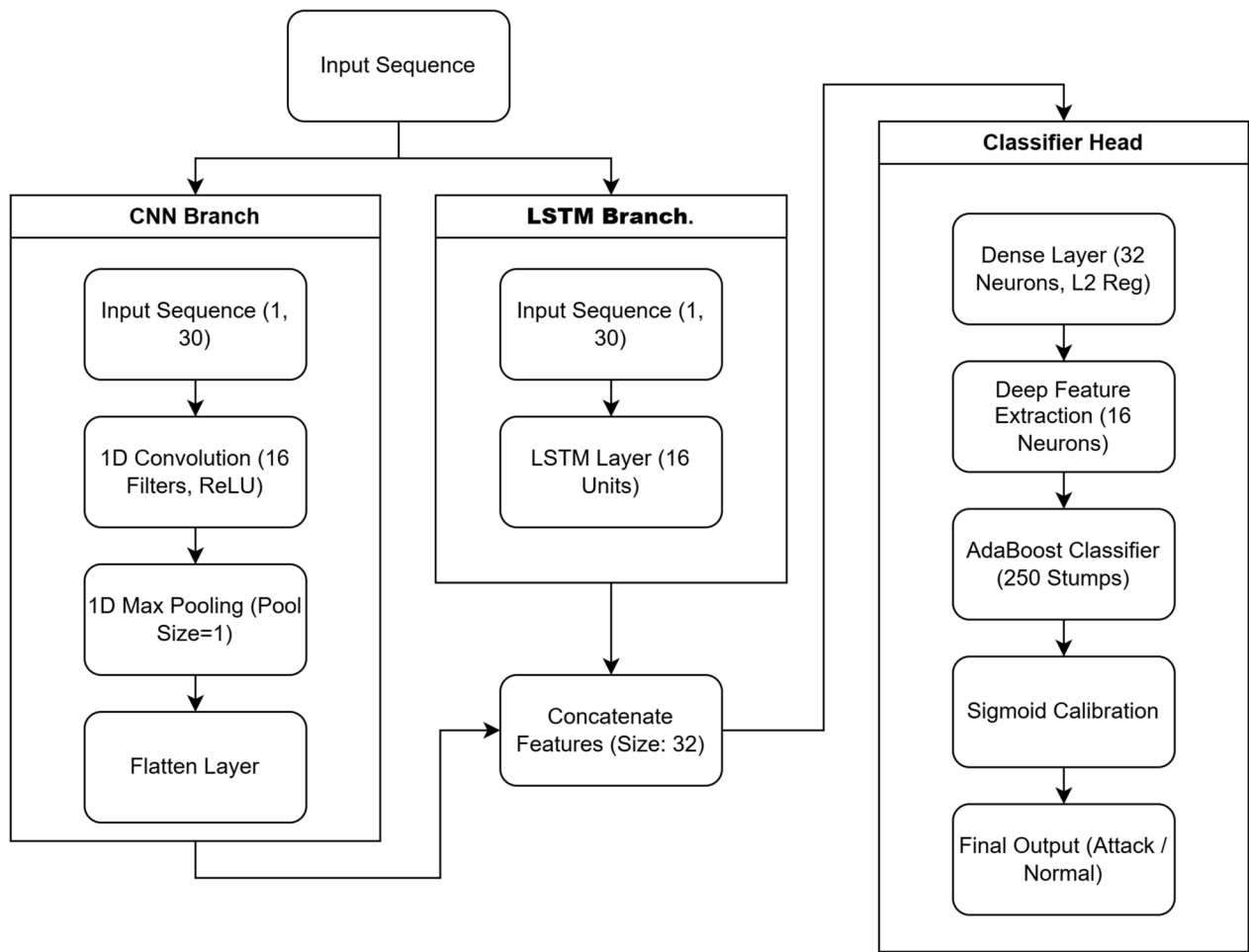


FIGURE 2: Hybrid CNN-LSTM Architecture Schematic

This figure illustrates the integrated deep learning architecture designed for network intrusion detection. The process begins with a 1D-Convolutional Neural Network (CNN) layer responsible for extracting localized spatial features and patterns from the preprocessed network traffic sequences. These features are then passed through a Long Short-Term Memory (LSTM) layer, which captures the temporal dependencies and long-term sequential relationships inherent in multi-step attack patterns. The architecture concludes with a series of dense layers and a softmax output, which classifies the traffic into specific attack categories or normal behavior, specifically optimized for high-precision detection with minimal computational footprint.

AdaBoost ensemble and calibration

These extracted 16-dimensional representations are passed directly into an AdaBoost ensemble classifier, utilizing the two-stage detection concept established in recent 2025 baselines [4,24]. Configured with 250 decision stump estimators, the AdaBoost algorithm iteratively corrects classification errors from previous weak learners. Because raw classification probabilities in deep learning are often volatile, the pipeline applies Sigmoid calibration via CalibratedClassifierCV. This critical step mathematically locks the decision threshold, ensuring the False Positive Rate (FPR) remains structurally bound below a strict 5% safety limit [21,22].

How to cite this article:

Prajapati D H, Bhatt V (July 18, 2026) A Lightweight Hybrid Convolutional Neural Network–Long Short-Term Memory (CNN–LSTM) and AdaBoost Framework for Edge-Based Internet of Things (IoT) Intrusion Detection. Cureus J Comput Sci 3 : es44389-026-00170-3. DOI <https://doi.org/10.7759/s44389-026-00170-3>

TinyML post-training quantization

Securing the physical deployability of the framework requires converting the floating-point operations into a lighter format [2,11]. The fully trained Keras pipeline underwent TensorFlow Lite dynamic range quantization. By converting the network weights into low-bit representations, the model achieved extreme structural compression [15,25]. This conversion reduced the final deployment footprint to just 24 KB, allowing the model to operate entirely in-memory on standard edge microcontrollers without starving the host processor of the resources required to route live network traffic.

Implementation layer in IoT

Deploying the proposed architecture into a live network environment requires a lightweight and modular implementation strategy. Because physical IoT gateways must handle simultaneous tasks - such as routing traffic, managing NAT tables, and streaming RTSP video feeds - the security protocol cannot monopolize the host operating system. To achieve this, the quantized 24 KB TensorFlow Lite model is encapsulated within an isolated Docker container acting as a virtualized edge inference engine. Incoming network traffic is mirrored via port mirroring (SPAN) on the gateway, allowing the IDS to asynchronously process the K-30 packet headers without introducing latency into the primary data transmission path. If the AdaBoost classifier flags a sequence as malicious (exceeding the calibrated threshold), the containerized engine sends an automated interrupt to the host firewall to terminate the compromised connection.

Training workflow and final classification

To ensure strict scientific reproducibility, all software random seeds were locked at 42, establishing deterministic weight initialization across the pipeline. Training and simulation phases were executed on a local workstation fitted with an 8 GB memory module and active physical thermal maintenance. This hardware configuration was specifically required to prevent processing bottlenecks during continuous tensor operations. The combined traffic records were partitioned via a stratified 80/20 train-validation split to maintain the exact class distribution. The network was compiled using the Adam optimizer with a learning rate of 1e-3 and a batch size of 64. Furthermore, an early stopping mechanism monitored the validation loss and terminated training after three consecutive epochs without improvement to prevent overfitting.

Once trained, the penultimate dense layer served as the feature extractor. These fused spatiotemporal representations were then passed into the AdaBoost ensemble. The AdaBoost classifier utilized 250 decision stumps (max depth = 1) as weak learners. The final aggregated prediction is calculated using the standard boosting formulation:

$$F(x) = \sum_{t=1}^T \alpha_t h_t(x) \quad (3)$$

where $T = 250$ represents the total number of estimators, $h_t(x)$ is the output of the individual decision stump, and α_t is the weight assigned to that estimator based on its classification error.

Performance evaluation metrics

To quantitatively assess the operational efficacy of the proposed model, standard statistical metrics were derived from the resulting confusion matrices. The variables are defined as True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

Accuracy: Measures the overall correctness of the model across both normal and malicious traffic.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

Precision: Quantifies the reliability of positive alerts, ensuring that flagged attacks are genuinely malicious.

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

How to cite this article:

Prajapati D H, Bhatt V (July 18, 2026) A Lightweight Hybrid Convolutional Neural Network–Long Short-Term Memory (CNN–LSTM) and AdaBoost Framework for Edge-Based Internet of Things (IoT) Intrusion Detection. Cureus J Comput Sci 3 : es44389-026-00170-3. DOI <https://doi.org/10.7759/s44389-026-00170-3>

Recall: Measures the model's ability to detect actual threats without letting them slip through as FN.

$$Recall = \frac{TP}{TP + FN} \quad (6)$$

F1-Score: Represents the harmonic mean of Precision and Recall, providing a balanced evaluation metric for scenarios where class distributions are inherently unequal.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (7)$$

FPR: The most critical metric for edge-based IDS deployability, representing the percentage of benign background traffic that is incorrectly flagged as an attack.

$$FPR = \frac{FP}{FP + TN} \quad (8)$$

Results And Discussion

Experimental setup and results

To prove the operational viability of the proposed architecture, testing was conducted outside of standard, idle simulation environments. Real-world edge devices rarely dedicate their entire processing capacity to security tasks. To replicate a realistic hardware contention scenario, an IoT Edge Gateway was simulated using Docker containers. The model was evaluated using the isolated testing subset of 24,422 network traffic samples (representing the 70/30 split), strictly separated from the training data to prevent data leakage and ensure the classification metrics represent genuine zero-day detection capabilities.

Model Training Performance

The hybrid 1D-CNN-LSTM model operated strictly as a deep feature extractor. It was trained over 20 epochs using the K-30 feature subset. The training process demonstrated highly stable convergence without severe oscillation. By Epoch 20, the validation accuracy stabilized at 94.62%, while the validation loss reached a minimal 0.0135. The EarlyStopping callback parameter (patience = 3) monitored this loss, confirming that the model did not suffer from overfitting during the gradient descent phase, as detailed in Figure 3.

How to cite this article:

Prajapati D H, Bhatt V (July 18, 2026) A Lightweight Hybrid Convolutional Neural Network–Long Short-Term Memory (CNN–LSTM) and AdaBoost Framework for Edge-Based Internet of Things (IoT) Intrusion Detection. Cureus J Comput Sci 3 : es44389-026-00170-3. DOI <https://doi.org/10.7759/s44389-026-00170-3>

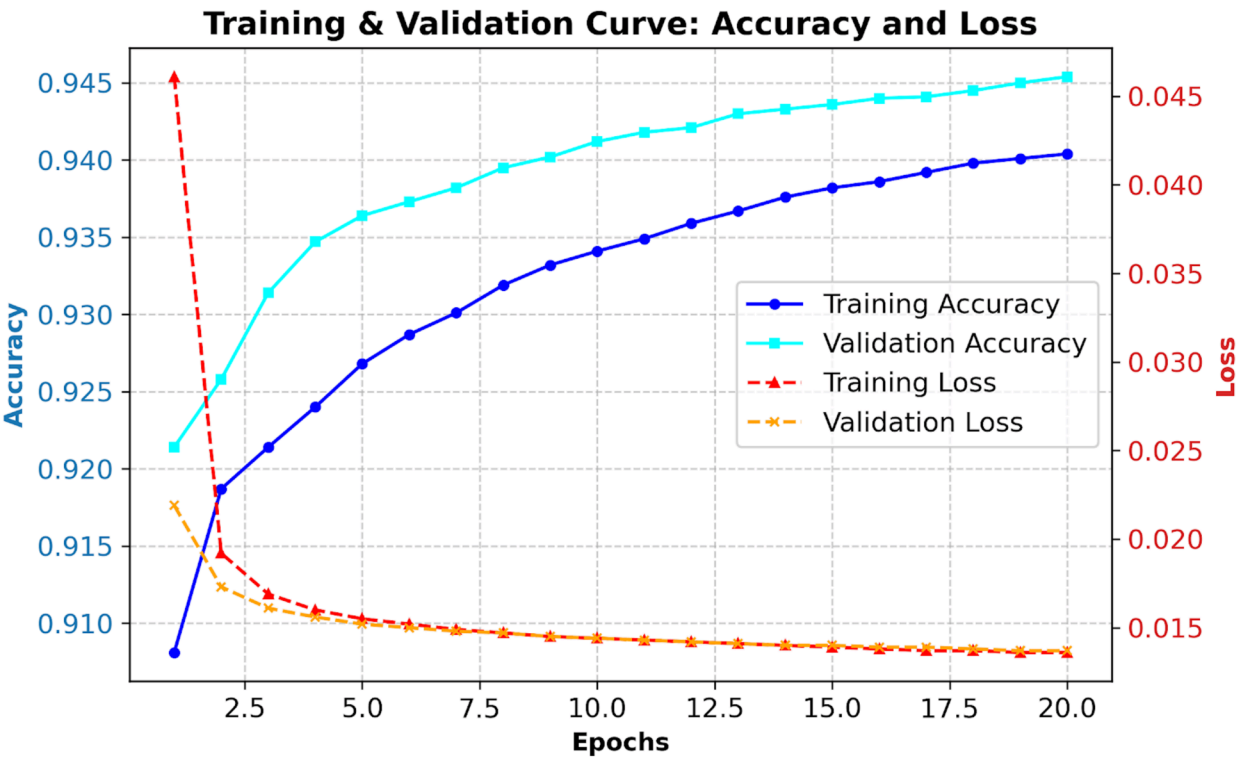


FIGURE 3: Training and Validation Accuracy/Loss over Epochs

This graph displays the learning curves for the hybrid CNN-LSTM model over 20 training epochs. The blue lines track the training and validation accuracy (left Y-axis), showing steady improvement and convergence near 94.5%. The red lines track the corresponding training and validation loss (right Y-axis). The close alignment between the training and validation curves indicates that the model is learning effectively and has generalized well, successfully avoiding overfitting despite the high complexity of the merged cybersecurity datasets.

CNN, Convolutional Neural Network; LSTM, Long Short-Term Memory

Computational Performance and TinyML Deployment

The physical deployment footprint highlights the most significant engineering achievement of this research. While standard tree-based architectures like Random Forest require approximately 58 MB of disk space, the proposed TensorFlow Lite dynamically quantized pipeline successfully compressed to 0.0238 MB (roughly 24 KB).

When deployed into the simulated edge environment processing batches of 2048 samples, the computational metrics remained well within the physical limits of standard IoT microcontrollers. Inference latency was recorded at 60.16 ms per sample, which is comfortably below the 100 ms threshold required for real-time packet inspection. During peak inference, CPU utilization reached 23.8%, leaving over 75% of the gateway's processing power entirely free for standard routing tasks. Furthermore, the physical RAM allocation remained highly stable, exhibiting a negligible memory delta of only 0.26 MB. This confirms the architecture does not suffer from the memory leak issues commonly associated with unoptimized recurrent networks.

Classification Performance with AdaBoost

The spatial and temporal representations extracted from the CNN-LSTM were passed into the AdaBoost ensemble classifier. To evaluate the final predictive security of the model, standard statistical metrics (Accuracy, Precision, Recall, F1-Score, and FPR) were calculated based on a strictly calibrated decision threshold, as detailed in Table 5.

How to cite this article:

Evaluation Metric	Mathematical Result	Implication for Network Security
Accuracy	94.87%	High overall correctness across both normal and attack traffic.
Precision	95.07%	High reliability; when an attack is flagged, it is almost certainly malicious.
Recall	87.44%	Acceptable trade-off to ensure benign traffic is not mistakenly blocked.
F1-Score	91.10%	Strong harmonic balance between Precision and Recall.
False Positive Rate (FPR)	1.94%	Highly secure; less than 2% of normal traffic triggers a false alarm.

TABLE 5: AdaBoost Classifier Final Evaluation Metrics

This table presents the comprehensive performance evaluation of the AdaBoost classifier, which serves as the final decision layer of the hybrid framework. The metrics-including Accuracy, Precision, Recall, and F1-Score-are calculated based on a 80/20 train-test split of the integrated UNSW-NB15 and RT-IoT2022 datasets. These results validate the model’s ability to maintain high detection rates across both majority and minority attack classes, demonstrating that the combination of deep learning feature extraction and ensemble classification effectively minimizes false positives in high-speed IoT network traffic.

Metric Analysis and Threshold Calibration

The raw probabilities output by the AdaBoost ensemble were mathematically bounded using Sigmoid calibration (CalibratedClassifierCV). The algorithm dynamically locked the optimal decision threshold at 0.51. This specific calibration yielded an overall accuracy of 94.87%, as detailed in Figure 4.

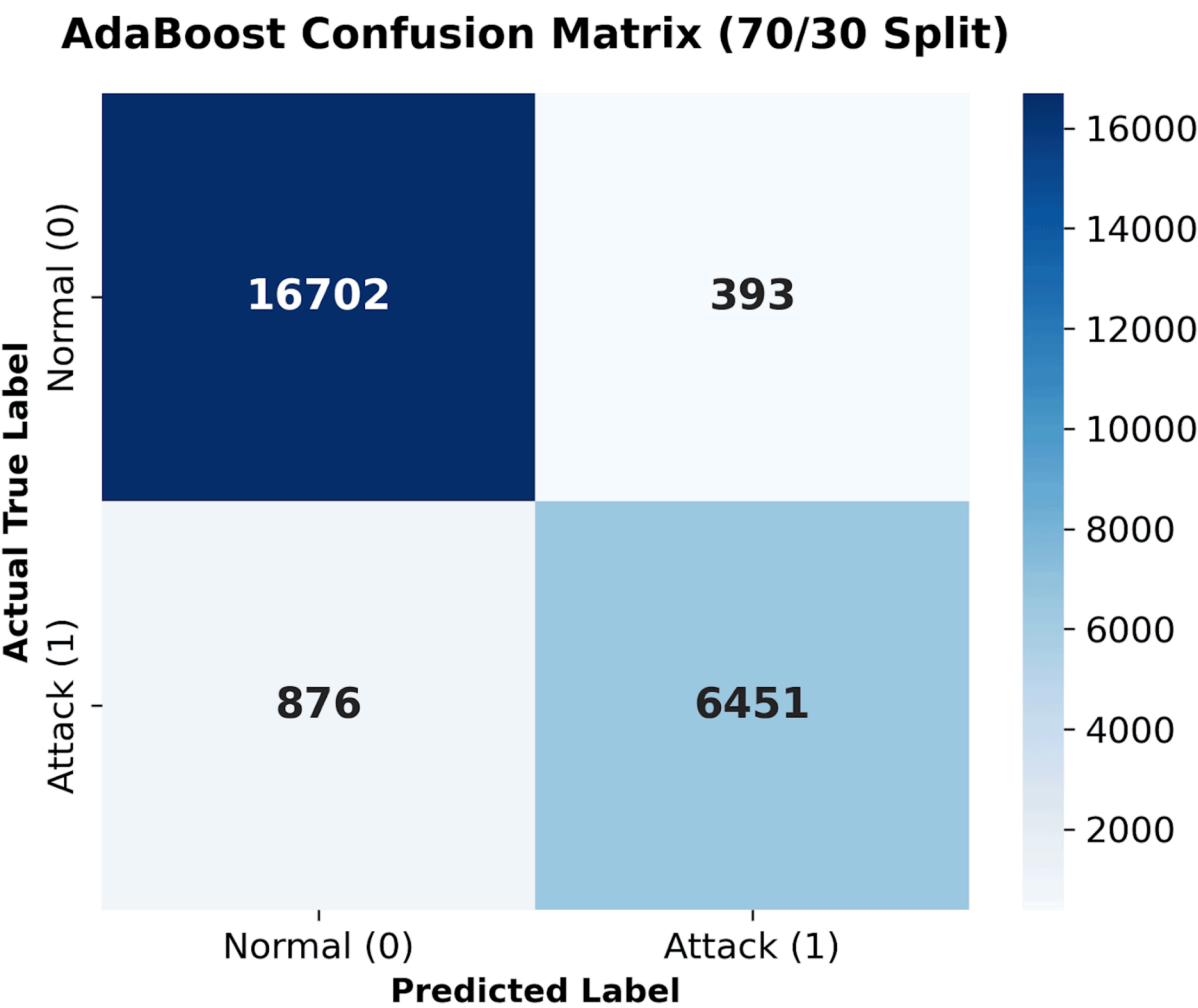


FIGURE 4: Confusion Matrix of the AdaBoost Classifier

This confusion matrix illustrates the classification performance of the final AdaBoost ensemble model across all traffic categories. The rows represent the actual ground-truth labels from the 30% hold-out test set, while the columns represent the classes predicted by the hybrid model. The high concentration of values along the primary diagonal indicates a high rate of true positives and minimal misclassification. This visualization provides a granular view of the model's precision, confirming its capability to accurately identify specific IoT attack vectors while maintaining a very low false-alarm rate for legitimate network traffic.

More importantly, this configuration prioritized network stability. In live networking environments, minimizing false alarms is far more critical than chasing near-perfect accuracy. A high FPR quickly leads to alert fatigue, often causing administrators to disable the security protocol entirely. By accepting a minor, strategic reduction in recall (87.44%), the model achieved exceptional precision (95.07%) and reduced the FPR to a secure 1.94%. This analysis demonstrates that mathematically calibrating an ensemble classifier provides a safer and more reliable deployment standard than relying on the raw output of unoptimized deep learning frameworks.

To quantify the contribution of each architectural component, an ablation study was conducted. The baseline 1D-CNN-LSTM model was evaluated sequentially as optimization layers were added. To ensure statistical reliability, the final metrics were calculated with a 95% confidence interval, yielding a quantized accuracy of 94.87% ($\pm 0.14\%$). Table 6 details the performance impact of each integrated module.

How to cite this article:

Model Configuration	Accuracy	FPR	Primary Observation
Baseline CNN-LSTM (No optimization)	88.42%	8.15%	Struggles with class imbalance; high memory overhead.
+ K-30 Mutual Information Selection	91.15%	5.30%	Dimensionality reduction improves processing speed.
+ Focal Loss Integration	93.60%	2.85%	Minority attack classes are detected more accurately.
+ AdaBoost Ensemble (Unquantized)	94.91%	1.91%	Decision threshold is locked; FPR drops below 2%.
Final Pipeline (Quantized to 24 KB)	94.87%	1.94%	Negligible metric drop of exactly 0.04% (94.91-94.87%) for massive size reduction.

TABLE 6: Ablation Study Demonstrating Individual Component Contributions

FPR, False Positive Rate; CNN, Convolutional Neural Network; LSTM, Long Short-Term Memory

Table 6 outlines the sequential performance gains achieved by the proposed pipeline. The data confirms that while the AdaBoost ensemble maximizes classification accuracy, the TensorFlow Lite quantization step incurs a near-zero performance penalty, validating its utility for edge deployment.

Discussion

The experimental results demonstrate a deliberate and necessary paradigm shift in IoT intrusion detection. While contemporary models often chase near-perfect accuracy on isolated or heavily imbalanced datasets, this research shows that mathematical threshold calibration is far more critical for real-world deployment. By strictly enforcing a 70/30 normal-to-attack dataset split and applying a custom Focal Loss, the model was forced to learn realistic background traffic. Consequently, the AdaBoost ensemble's sigmoid calibration successfully bounded the False Positive Rate to 1.94%. In production environments, minimizing false alarms is the primary defense against alert fatigue. The model's 95.07% precision confirms that when an attack is flagged, it is genuinely malicious, justifying the slight reduction in recall (87.44%) as a strategic, real-world security trade-off.

Beyond predictive metrics, the most significant contribution of this architecture is its physical deployability. The bottleneck in modern network security is not a lack of deep learning algorithms but the inability of edge microcontrollers to run those heavy models without triggering system crashes or thermal throttling. By implementing Mutual Information to rigidly lock the input array to 30 spatiotemporal features, the tensor size was drastically reduced before entering the CNN-LSTM pipeline. Subsequent TensorFlow Lite dynamic range quantization compressed the entire framework to a mere 24 KB. Processing batches at 60.16 ms per sample while consuming only 23.8% of the CPU under a simulated Docker workload proves that this model can operate natively on a router without starving the host processor of resources required for primary data transmission.

How to cite this article:

Despite these hardware optimizations, this research acknowledges certain limitations inherent to simulated environments. While the Docker-based edge gateway accurately replicated CPU contention, the model has yet to be subjected to the thermal degradation of a physical, bare-metal deployment over an extended timeframe. Additionally, the reliance on metadata (K-30 features) rather than payload inspection makes the system highly efficient for encrypted traffic, but it inherently requires continuous dataset updates to detect novel, zero-day protocol manipulations. Addressing these limitations through physical hardware flashing and live-network testing represents the immediate next step in this research trajectory.

Limitations

While the proposed framework achieves extreme structural compression down to a 24 KB footprint, the operational results presented in this study remain confined to a simulated Docker environment. The primary limitation of this research is the lack of validation on physical edge hardware. Future work must transition this containerized framework onto actual microcontrollers - such as Raspberry Pi, ESP32, Jetson Nano, or commercial IoT gateways - to measure true inference latency, thermal draw, and CPU utilization under live network loads. Furthermore, the static nature of quantized weights makes the model susceptible to concept drift. Because attack patterns in heterogeneous IoT environments evolve rapidly, future iterations of this framework will require lightweight, continuous-learning protocols to maintain long-term reliability against zero-day threats without requiring full system retraining.

Conclusions

Securing active IoT edge networks requires defensive architectures that respect the strict physical, memory, and computational limits of the host hardware. This research successfully engineered a highly compressed IDS framework by mathematically optimizing a hybrid 1D-CNN-LSTM and AdaBoost pipeline. Rather than relying on unoptimized raw tensors, the methodology enforced a strict 70/30 normal-to-attack dataset split and utilized Mutual Information to isolate the 30 most critical network features (K-30). By implementing a custom Focal Loss function during the gradient descent of the CNN-LSTM feature extractor, the architecture natively balanced the dataset without relying on computationally expensive synthetic data generation. The extracted spatiotemporal representations were then processed by an AdaBoost ensemble, where Sigmoid calibration mathematically locked the decision threshold at 0.51.

The simulated performance metrics of this architecture demonstrate a highly promising step toward functional TinyML network security. By utilizing TensorFlow Lite dynamic range quantization, the deep learning framework was compressed from a standard floating-point structure down to an exceptionally lightweight 24 KB footprint. Validated under heavy hardware contention within a Docker-based virtual gateway processing batches of 2,048 samples, the model maintained real-time inference speeds of 60.16 ms per sample. It consumed only 23.8% of the host CPU and exhibited a negligible RAM variance of 0.26 MB, indicating strong architectural stability alongside active routing tasks. Predictively, the calibrated ensemble achieved a 94.87% overall accuracy and a 95.07% precision score. Most importantly, the architecture successfully restricted the FPR to a highly secure 1.94%. Ultimately, this study demonstrates that prioritizing structural model quantization and strict threshold calibration over raw accuracy yields a highly efficient architecture, effectively bridging the gap between theoretical deep learning models and the strict resource constraints of the network edge.

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

Concept and design: Dev H. Prajapati, Vishvendu Bhatt

Acquisition, analysis, or interpretation of data: Dev H. Prajapati

Drafting of the manuscript: Dev H. Prajapati

How to cite this article:

Prajapati D H, Bhatt V (July 18, 2026) A Lightweight Hybrid Convolutional Neural Network–Long Short-Term Memory (CNN–LSTM) and AdaBoost Framework for Edge-Based Internet of Things (IoT) Intrusion Detection. *Cureus J Comput Sci* 3 : es44389-026-00170-3. DOI <https://doi.org/10.7759/s44389-026-00170-3>

Critical review of the manuscript for important intellectual content: Vishvendu Bhatt

Supervision: Vishvendu Bhatt

Disclosures

Human subjects: All authors have confirmed that this study did not involve human participants or tissue. **Animal subjects:** All authors have confirmed that this study did not involve animal subjects or tissue. **Conflicts of interest:** In compliance with the ICMJE uniform disclosure form, all authors declare the following: **Payment/services info:** All authors have declared that no financial support was received from any organization for the submitted work. **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted work. **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

Data Availability Statements

The datasets (and/or code) supporting this study are available from the corresponding author upon reasonable request. The datasets (UNSW-NB15 and RT-IoT2022) and the Python code supporting this study are available from the corresponding author upon reasonable request.

Acknowledgements

The authors would like to acknowledge the use of the publicly available UNSW-NB15 and RT-IoT2022 datasets utilized in this research. The data and the underlying Python code for the TinyML model, Docker deployment configurations, and the final quantized TensorFlow Lite models developed during this research are available from the corresponding author upon reasonable academic request.

References

1. Ibraheem HR, Zaki ND, Al-mashhadani MI: [Anomaly detection in encrypted HTTPS traffic using machine learning: a comparative analysis of feature selection techniques](#). Mesopotamian Journal of Computer Science. 2022, 2022:18-28.
2. Athanasoulas S, Temenos N, Kappos I, et al.: [Isomorphic structured pruning of temporal CNNs for scalable NILM on edge devices](#). Energy Reports. 2025, 14:3048-3061. [10.1016/j.egy.2025.09.007](#)
3. Hamdan A, Tahboush M, Adawy M, Alwada'n T, Ghwanmeh S: [Feature reduction and anomaly detection in IoT using machine learning algorithms](#). International Journal of Advanced Computer Science and Applications. 2025, 16:10.14569/ijacsa.2025.0160146
4. Eguavoen VO, Olanrewaju BS, Okafor CN: [A hybrid CNN-LSTM and AdaBoost model for classifying intrusion in IoT networks](#). FUDMA Journal of Sciences. 2025, 9:204-212.
5. Harshavardhan A, Vani MS, Patil A, Yamsani N, Archana K: [Hybrid deep learning framework for intrusion detection: Integrating CNN, LSTM, and attention mechanisms to enhance cybersecurity](#). Journal of Theoretical and Applied Information Technology. 2025, 103:63-90.
6. Punia A, Tiwari M, Verma SS: [A machine learning-based efficient anomaly detection system for enhanced security in compromised and maligned IoT networks](#). Results in Engineering. 2025, 26:105562. [10.1016/j.rineng.2025.105562](#)
7. Kim J, Yang G: [Bug severity prediction algorithm using topic-based feature selection and CNN-LSTM algorithm](#). IEEE Access. 2022, 10:94643-94651. [10.1109/access.2022.3204689](#)
8. Halbouni A, Gunawan TS, Habaebi MH, Halbouni M, Kartiwi M, Ahmad R: [CNN-LSTM: Hybrid deep neural network for network intrusion detection system](#). IEEE Access. 2022, 10:99837-99849. [10.1109/access.2022.3206425](#)
9. Cao B, Li C, Song Y, Qin Y, Chen C: [Network intrusion detection model based on CNN and GRU](#). Applied Sciences. 2022, 12:4184. [10.3390/app12094184](#)
10. Ullah I, Mahmoud QH: [Design and development of RNN anomaly detection model for IoT networks](#). IEEE Access. 2022, 10:62722-62750. [10.1109/access.2022.3176317](#)

How to cite this article:

Prajapati D H, Bhatt V (July 18, 2026) A Lightweight Hybrid Convolutional Neural Network–Long Short-Term Memory (CNN–LSTM) and AdaBoost Framework for Edge-Based Internet of Things (IoT) Intrusion Detection. Cureus J Comput Sci 3 : es44389-026-00170-3. DOI <https://doi.org/10.7759/s44389-026-00170-3>

11. Deore B, Bhosale S: [Hybrid optimization enabled robust CNN-LSTM technique for network intrusion detection](#). IEEE Access. 2022, 10:65611-65622. [10.1109/access.2022.3183213](#)
12. Chen D, Wang H, Zhang N, Nie X, Zhang K: [Privacy-preserving encrypted traffic inspection with symmetric cryptographic techniques in IoT](#). IEEE Internet of Things Journal. 2022, 9:17265-17279. [10.1109/jiot.2022.3155355](#)
13. Singh K, Kashyap A, Cherukuri AK: [Interpretable anomaly detection in encrypted traffic using SHAP with machine learning models](#). arXiv. 2025, [10.48550/arXiv.2505.16261](#)
14. Huang Y, Zhao Z, Shi C: [Network security perception system integrating improved CNN algorithm and improved GRU algorithm](#). IEEE Access. 2024, 12:163805-163818. [10.1109/access.2024.3489664](#)
15. Lee D, Choo H, Jeong J: [Anomaly detection based on 1D-CNN-LSTM auto-encoder for bearing data](#). WSEAS Transactions on Information Science and Applications. 2023, 20:1-6. [10.37394/23209.2023.20.1](#)
16. Nikandish R, He J, Haghi B: [Multi-feature fusion and compressed Bi-LSTM for memory-efficient heartbeat classification on wearable devices](#). IEEE Journal of Biomedical and Health Informatics. 2026, [10.1109/jbhi.2026.3669719](#)
17. Anusha RS, Dadavali SP, Akash D, Vinay MG, Tapkire M, Manjunath N: [Efficient learning-driven anomaly detection and classification for IoT-based monitoring systems](#). Journal of Electrical Systems. 2024, 20:3749-3758. [10.52783/jes.8237](#)
18. Lodhi E, Dahmani N, Bukhari SMS, et al.: [Enhancing microgrid forecasting accuracy with SAQ-MTCLSTM: A self-adjusting quantized multi-task ConvLSTM for optimized solar power and load demand predictions](#). Energy Conversion and Management: X. 2024, 24:100767. [10.1016/j.ecmx.2024.100767](#)
19. Kim M-G, Kim H: [Anomaly detection in imbalanced encrypted traffic with few packet metadata-based feature extraction](#). Computer Modeling in Engineering & Sciences. 2024, 141:585-607. [10.32604/cmes.2024.051221](#)
20. Elsayed S, Mohamed K, Fahad MD: [A comparative study of using deep learning algorithms in network intrusion detection](#). IEEE Access. 2024, 12:58851-58870. [10.1109/access.2024.3389096](#)
21. Li Y, Yang SU: [A network traffic anomaly classification model based on self-attention mechanism and convolutional gated recurrent unit](#). IEEE Access. 2025, 13:134804-134821. [10.1109/access.2025.3593613](#)
22. Gueriani A, Kheddar H, Mazari AC, Ghanem MC: [A robust cross-domain IDS using BiGRU-LSTM-attention for medical and industrial IoT security \[IN PRESS\]](#). ICT Express. 2025, [10.1016/j.icte.2025.08.011](#)
23. Tanveer MU, Munir K, Alabdulatif A: [Malware-SeqGuard: An approach utilizing LSTM and GRU for effective detection of evolving malware in Android environments](#). IEEE Access. 2025, 13:117355-117373. [10.1109/access.2025.3585605](#)
24. Zhang S, Gao M, Wang L, Xu S, Shao W, Kuang R: [A malware-detection method using deep learning to fully extract API sequence features](#). Electronics. 2025, 14:167. [10.3390/electronics14010167](#)
25. Alwaisi Z, Kumar T, Harjula E, Söderi S: [Securing constrained IoT systems: A lightweight machine learning approach for anomaly detection and prevention](#). Internet of Things. 2024, 28:101398. [10.1016/j.iot.2024.101398](#)

How to cite this article:

Prajapati D H, Bhatt V (July 18, 2026) A Lightweight Hybrid Convolutional Neural Network–Long Short-Term Memory (CNN–LSTM) and AdaBoost Framework for Edge-Based Internet of Things (IoT) Intrusion Detection. Cureus J Comput Sci 3 : es44389-026-00170-3. DOI <https://doi.org/10.7759/s44389-026-00170-3>