International Journal of Innovative Information Systems & Technology Research 14(3):314-327, July-Sept, 2026

© SEAHI PUBLICATIONS, 2026 www.seahipublications.org ISSN: 2467-8562

DOI:10.5281/zenodo.22806979

Major Model for Network Intrusion Detection System in Edge Computing Environments

[1] P. H. Aliezi *[2] C. G. Igiri *[3] V. T. Emmah

[1,2&3]Department of Computer Science, Rivers State University, Port-Harcourt, Nigeria
aliezi.prettyfavour@rsu.edu.ng, igiri.chima@ust.edu.ng, victor.emmah@ust.edu.ng

Abstract

Edge computing places computation close to users, gateways, local servers, and Internet of Things devices, reducing delay but also widening the points at which abnormal traffic, unauthorized access, and data tampering can occur. This paper presents a hybrid Convolutional Neural Network and Long Short-Term Memory model for network intrusion detection in an edge-computing environment. The method combines local traffic collection, preprocessing, convolutional feature extraction, temporal sequence learning, classification, alert generation, event logging, and a supporting SHA-256 file-integrity audit component. Model training and testing were carried out with CICIDS2017 network traffic data. A selected subset of 10,000 records was preprocessed through data cleaning, categorical encoding, normalization, scaling, and an 80 percent training and 20 percent testing split. The objective was to improve intrusion detection accuracy and response speed for edge-based security monitoring while reducing dependence on centralized cloud analysis. Accuracy, recall, precision, F1-score, latency, and response time were used for evaluation, with standalone Convolutional Neural Network and Long Short-Term Memory models serving as baseline comparisons. The hybrid model achieved 96.78 percent accuracy, 97.44 percent recall, 96.09 percent precision, and 96.76 percent F1-score, with 25 ms latency and 35 ms response time. The supporting audit module detected a modified Portable Document Format file through differences in hash value, page count, and word count. The findings suggest that convolutional feature extraction and temporal sequence learning can support fast, reliable intrusion detection at the edge.

Keywords: Network intrusion detection; edge computing; Convolutional Neural Network Long Short-Term Memory; deep learning; cybersecurity; file integrity; anomaly detection; Internet of Things security

1.0 Introduction

The rapid growth of cloud computing, Internet of Things services, mobile platforms, and distributed digital applications has increased the amount of data generated and processed close to users and devices. Edge computing addresses some limitations of centralized cloud processing by placing computation and storage near routers, gateways, sensors, local servers, and user devices (Abbas et al., 2018). Although this reduces bandwidth consumption and improves response time, it also creates a wider and more distributed security surface (Roman et al., 2018). Unlike a centralized cloud environment, an edge environment may contain many resource-limited nodes that process traffic locally, exchange data frequently, and operate close to physical users and devices. As a result, attacks can appear at several entry points before traffic reaches a central security system.

For this reason, intrusion detection in edge computing must identify malicious activity close to the point where it occurs. Intrusion detection systems identify activities such as unauthorized access, denial-of-service attacks, port scanning, malware propagation, abnormal traffic flows, and policy violations (Garcia-Teodoro et al.,

2009). Traditional signature-based methods can detect known attack patterns, but they are weak against new and unknown attacks. Anomaly-based methods can detect unfamiliar behaviour, but they often generate high false alarms when legitimate traffic patterns change. These weaknesses become more serious at the edge, where delayed detection can affect local devices, services, and data before cloud-side analysis is completed.

Machine learning and deep learning have improved intrusion detection by learning attack patterns from network traffic data (Berman et al., 2019; Buczak & Guven, 2016). Deep learning is useful for edge intrusion detection because traffic records often contain feature relationships and sequential behaviour that are difficult to define manually before training (Gamage & Samarabandu, 2020). In a hybrid model, Convolutional Neural Network layers can extract local feature patterns from traffic attributes, while Long Short-Term Memory layers can learn time-dependent relationships in traffic sequences (Altunay & Albayrak, 2023; Khan et al., 2019). The combination is useful because an intrusion may first appear as an unusual feature pattern and become clearer only after repeated activity is observed over time.

The problem addressed in this work is that many existing intrusion detection systems are still designed mainly for centralized environments and do not fully support fast, lightweight, and scalable security at the edge. Cloud-only detection can increase bandwidth use and latency when large volumes of traffic from Internet of Things devices and local servers are forwarded for remote analysis. Signature-based systems also struggle with zero-day attacks, while anomaly-based systems may produce excessive false positives. In addition, edge environments often handle local file uploads and document exchange, which creates a need for integrity checking so that tampering and unauthorized modification can be detected quickly.

The objective is to develop and evaluate a hybrid Convolutional Neural Network and Long Short-Term Memory model for distinguishing normal and malicious network activities in an edge-computing environment. The work contributes an edge-oriented intrusion detection workflow that combines local preprocessing, convolutional feature extraction, temporal sequence learning, classification, alert generation, and event logging. It also compares the proposed hybrid model with standalone Convolutional Neural Network and Long Short-Term Memory baselines and includes a supporting file-integrity audit module for detecting altered documents in the edge-security workflow.

2.0 Related Work

Garcia-Teodoro et al. (2009) reviewed anomaly-based network intrusion detection techniques and grouped detection methods according to statistical, knowledge-based, and machine-learning approaches. Although the review did not propose a new deep-learning model, it showed that anomaly-based intrusion detection can identify previously unseen attacks when normal behaviour is properly modelled. A persistent weakness was false alarm generation when legitimate network behaviour changes. That problem is relevant to edge computing because traffic can vary across gateways, Internet of Things devices, and local services, making false-positive control necessary for practical deployment.

Buczak and Guven (2016) examined data mining and machine-learning methods for cybersecurity intrusion detection, including decision trees, support vector machines, Bayesian methods, artificial neural networks, and ensemble models. Their survey showed that machine-learning approaches can improve detection compared with rigid rule-based systems, using performance measures such as accuracy, detection rate, and false-alarm rate across benchmark datasets. However, many of the reviewed methods depended on manual feature engineering and were not designed specifically for resource-constrained edge nodes. This gap supports the need for deep models that can learn useful representations from traffic data while still being evaluated for speed.

Berman et al. (2019) surveyed deep-learning applications in cybersecurity, including intrusion detection, malware analysis, phishing detection, and threat classification. The review discussed model structures such as deep neural networks, convolutional neural networks, recurrent neural networks, autoencoders, and generative models. It showed that deep learning can improve automatic feature learning, while also drawing attention to computational cost, explainability, adversarial vulnerability, and dependence on large training datasets. These limitations are directly relevant to edge intrusion detection because edge environments require accurate models that can also operate under latency and resource constraints.

Ferrag et al. (2020) reviewed deep-learning approaches, intrusion detection datasets, and comparative experimental findings for cybersecurity intrusion detection. Their review covered recurrent neural networks, convolutional neural networks, deep belief networks, restricted Boltzmann machines, autoencoders, and hybrid models. The study is relevant because it shows that dataset choice and model structure strongly influence IDS performance. However, it remained more focused on deep-learning IDS broadly than on edge-level deployment constraints.

Tsimenidis et al. (2022) reviewed deep learning in Internet of Things intrusion detection and discussed model families such as multilayer perceptrons, recurrent neural networks, convolutional neural networks, autoencoders, and generative models. Their review emphasized that deep learning can improve detection in IoT environments but also faces challenges such as computational cost, model deployment, and distributed operation. This supports this work because edge-based IDS must balance detection quality with deployability.

Gamage and Samarabandu (2020) compared deep-learning methods for network intrusion detection and discussed common experimental weaknesses in the field. Their review examined models such as deep neural networks, convolutional neural networks, recurrent neural networks, and autoencoder-based systems, with evaluation based on metrics such as accuracy, precision, recall, F1-score, and false-positive rate. They noted that reported results often depend strongly on dataset selection, train-test split strategy, and reproducibility details. Since the paper was a broad review rather than an edge-specific implementation, it also points to the need to compare the proposed hybrid model with standalone baseline models under the same experimental conditions.

Liu and Lang (2019) surveyed machine-learning and deep-learning approaches for intrusion detection systems and emphasized that detection performance depends on traffic type, attack class, dataset structure, and deployment environment. Their review covered classical classifiers and neural-network models, including deep belief networks, recurrent neural networks, and convolutional approaches. Accuracy, detection rate, and false-alarm rate were highlighted as major evaluation criteria. However, many reviewed approaches were still assessed mainly in centralized benchmark settings. For the present work, that gap matters because edge computing requires attention not only to classification performance but also to fast local response.

da Costa et al. (2019) surveyed machine-learning-based intrusion detection approaches for Internet of Things environments. The review examined intelligent techniques used for IDS and highlighted the growing need for security mechanisms that can cope with IoT scale, heterogeneity, and resource limits. Its limitation for this study is that it focused mainly on machine-learning methods rather than hybrid deep-learning architectures, but it helps justify the edge and IoT direction of this work.

Diro and Chilamkurti (2018) proposed a distributed attack detection scheme using deep learning for Internet of Things environments. Their work compared distributed fog-level detection with centralized cloud-based detection and showed that distributed detection can improve scalability for IoT security. The study is important here because it directly supports the argument that intrusion detection should move closer to edge or fog nodes instead of relying only on remote cloud analysis.

Javaid et al. (2016) proposed a deep-learning intrusion detection method based on self-taught learning with a sparse autoencoder on the NSL-KDD dataset. Their methodology used unsupervised feature learning followed by supervised classification, and the model was evaluated with accuracy, precision, recall, and F-measure. The results demonstrated that deep representation learning could support intrusion detection better than manually designed features. However, the model was not designed for edge computing and used an older benchmark dataset, so its behaviour under modern edge traffic and latency requirements was not established.

Khan et al. (2019) proposed a scalable hybrid intrusion detection system based on Spark machine learning and a Convolutional-LSTM network. The model used a two-stage structure: an anomaly-detection module based on Spark machine learning and a misuse-detection module based on Convolutional-LSTM learning. Evaluated on the ISCX-UNB dataset, the hybrid system achieved 97.29 percent accuracy during 10-fold cross-validation. That work is close to this research because it combines convolutional and temporal learning for intrusion detection. Its focus, however, was large-scale processing rather than edge-level latency, local alerting, and file-integrity support.

Jullian et al. (2023) implemented a distributed deep-learning framework for detecting cyber-attacks in Internet of Things networks. The framework evaluated feed-forward neural network and Long Short-Term Memory models on NSL-KDD and BoT-IoT datasets, reporting accuracy as high as 99.95 percent across some experimental setups. The study supports distributed IDS design for IoT, although it did not focus on the same Convolutional Neural Network and Long Short-Term Memory feature-fusion structure or supporting file-integrity audit used in this work.

Altunay and Albayrak (2023) developed Convolutional Neural Network, Long Short-Term Memory, and hybrid Convolutional Neural Network plus Long Short-Term Memory models for industrial Internet of Things intrusion detection. Their methodology used the UNSW-NB15 and X-IIoTID datasets for binary and multi-class classification. The hybrid model achieved 93.21 percent binary accuracy and 92.90 percent multi-class accuracy on UNSW-NB15, and 99.84 percent binary accuracy and 99.80 percent multi-class accuracy on X-IIoTID. Their findings support the use of hybrid spatial-temporal learning in distributed industrial networks. A remaining limitation is that the evaluation focused more on classification accuracy than on full edge deployment behaviour such as latency, logging, and response time.

Vinayakumar et al. (2019) investigated deep-learning models for intelligent intrusion detection using benchmark intrusion datasets. Their work used deep neural network structures and evaluated performance with classification metrics such as accuracy, precision, recall, and F1-score. The results showed that deep models can learn useful representations for attack detection when trained on representative network traffic. Computational cost and deployment feasibility were not the central focus, leaving open the question of how such models perform when pushed closer to edge devices.

Khraisat et al. (2019) surveyed intrusion detection systems, datasets, techniques, and challenges, with attention to signature-based, anomaly-based, and hybrid methods. The review discussed detection accuracy, false-positive rate, zero-day detection, benchmark quality, and attack coverage as important measurement criteria. Its main contribution was to show that benchmark selection and evaluation design strongly affect IDS results. Because it was not an implementation study, it did not solve the deployment issue for edge computing. Nevertheless, it justifies using a labelled benchmark dataset and comparing model structures under the same conditions.

Sharafaldin et al. (2018) introduced the CICIDS2017 dataset to provide realistic benign and attack traffic for intrusion detection evaluation. The methodology generated traffic flows from benign user profiles and attack scenarios, then extracted labelled flow features for supervised learning. The dataset includes several attack categories and is widely used because it supports evaluation with accuracy, precision, recall, F1-score, and confusion-matrix-based analysis. Its limitation is that benchmark traffic cannot fully represent every real edge deployment, especially where device type, bandwidth, and local traffic patterns differ. In the present work, CICIDS2017 provides the traffic source for evaluating the proposed model, while the selected 10,000-record subset remains a limitation.

Moustafa and Slay (2015) introduced UNSW-NB15 as a comprehensive dataset for network intrusion detection research. The dataset was designed to represent modern network traffic and attack behaviours more realistically than older benchmarks. Although this experiment uses CICIDS2017, UNSW-NB15 remains important because several related Convolutional Neural Network, Long Short-Term Memory, and hybrid IDS studies use it for comparison.

Koroniotis et al. (2019) developed the Bot-IoT dataset for Internet of Things network forensic analytics and botnet attack evaluation. The dataset is important to edge and IoT intrusion detection because botnet and distributed denial-of-service attacks are common threats in connected-device environments. Its relevance to this article is that it shows the need to test IDS models on IoT-specific traffic beyond general network datasets.

Farhan et al. (2025) developed a network-based intrusion detection approach using a Sequential Deep Neural Network with Extra Tree-based feature selection on the UNSW-NB15 dataset. Their methodology combined feature selection with deep classification and reported strong binary classification performance using standard detection metrics. The work shows that feature preparation remains important even when deep-learning models are used. However, it is not a direct benchmark for the present work because it used a different dataset

and did not focus on a Convolutional Neural Network and Long Short-Term Memory architecture for edge intrusion detection.

Tamuka et al. (2026) presented a systematic review of intrusion detection in fog computing using PRISMA-guided search and bibliometric analysis. The review grouped fog intrusion detection systems into traditional, machine-learning, deep-learning, and hybrid or ensemble categories. It reported that hybrid and deep-learning approaches commonly achieved accuracy values between 95 and 99 percent on benchmark datasets such as NSL-KDD, UNSW-NB15, CICIDS2017, KDD99, and BoT-IoT. The review also identified major limitations, including computational load, false positives under concept drift, weak generalization to unseen attacks, privacy risks from centralizing data, and limited real-world validation. Those concerns support the edge focus of the present work because high benchmark accuracy still needs deployment-aware evaluation.

Rezapour et al. (2021) conducted a systematic review of security issues, challenges, and solutions in fog computing. Their review identified security concerns related to reliability, access control, attacks, secure connection, privacy, and other fog-specific cases. This source strengthens the edge-computing framing of the article because fog and edge systems share the need for low-latency processing close to end devices while still protecting distributed data and services.

Roy et al. (2022) proposed a lightweight supervised intrusion detection mechanism for IoT networks. Their model used optimization steps such as multicollinearity removal, sampling, and dimensionality reduction to reduce computational burden while maintaining strong detection performance on CICIDS2017 and NSL-KDD. The work is useful for this study because it shows why edge IDS evaluation should consider both detection quality and computational efficiency.

Islam et al. (2026) reviewed lightweight intrusion detection systems for Internet of Things edge environments using a PRISMA-ScR method. The review synthesized 78 peer-reviewed studies from 2017 to March 2026 and evaluated deployability evidence such as model compactness, preprocessing burden, system placement, hardware measurements, operational robustness, and reproducibility. It found that no single IDS family dominates all deployment scenarios, because rankings change when hardware constraints or operational robustness are prioritized. For edge-based intrusion detection, this point reinforces the need to report operational measures such as latency and response time alongside accuracy.

Taken together, the reviewed studies show that deep learning and hybrid architectures can improve intrusion detection performance, especially when models combine feature extraction and sequence learning. However, much of the literature still emphasizes benchmark accuracy more than edge-specific deployment needs. The present work addresses this gap by developing a hybrid Convolutional Neural Network and Long Short-Term Memory model, comparing it with standalone baselines, reporting classification and operational metrics, and including a supporting file-integrity audit component for edge-security monitoring.

3.0 MATERIALS AND METHODS

This section discusses the methodology used, dataset, dataset structure and architecture of the model.

3.1 Methodology Used

The model was developed for edge-level intrusion detection, where traffic can be inspected near gateways, routers, Internet of Things devices, and local edge servers before every record is forwarded to a central cloud service. The workflow covered data collection, preprocessing, sequence construction, model training, model testing, performance evaluation, alert generation, event logging, and supporting file-integrity auditing.

Network traffic records were first collected from the CICIDS2017 dataset and converted into numerical form for model training. Preprocessing involved data cleaning, categorical encoding, normalization, scaling, and dataset partitioning. The processed records were then reshaped into sequential input samples so that the model could learn both feature-level patterns and time-dependent behaviour. Python was used for implementation, with TensorFlow and Keras supporting deep-learning model development, Pandas and NumPy supporting data handling, and Scikit-learn supporting preprocessing, train-test splitting, and metric calculation.

The intrusion-detection component was based on a hybrid Convolutional Neural Network and Long Short-Term Memory model. The Convolutional Neural Network component extracted local feature patterns from each prepared traffic sequence, while the Long Short-Term Memory component learned temporal

dependencies across the sequence. The final classification layer produced a binary decision indicating whether a traffic sequence was normal or intrusion-related. After classification, the system generated alerts and event logs containing information such as detected event type, timestamp, source and destination information, confidence value, and status.

A separate file-integrity audit module was included as a supporting edge-security component. This module was not used to calculate the Convolutional Neural Network and Long Short-Term Memory classification metrics. Instead, it compared an original Portable Document Format file with a test file using SHA-256 hash values, page count, and word count. Keeping the two outputs separate made it possible to interpret network classification results and document-integrity results independently.

3.2 Dataset and Dataset Structure

The experiment used CICIDS2017 network traffic data. CICIDS2017 contains labelled benign and attack traffic flows and is widely used for evaluating supervised intrusion detection models (Sharafaldin et al., 2018). A selected subset of 10,000 records was used in this study. The records were cleaned, encoded, normalized, scaled, and divided into 80 percent training data and 20 percent testing data. This produced 8,000 training records for model fitting and 2,000 testing records for unseen evaluation.

The model input was organized as a sequence of 10 time steps with 7 features at each time step. These features represented selected traffic indicators such as packet statistics, flow duration, protocol behaviour, and related activity attributes after preprocessing. The target label was treated as a binary class, where records were classified as either normal traffic or intrusion-related traffic.

Table 1: Dataset Structure Used for Model Development

Dataset item	Description
Dataset source	CICIDS2017 network traffic dataset
Selected records	10,000 records
Training split	80% of selected records, equal to 8,000 records
Testing split	20% of selected records, equal to 2,000 records
Input sequence length	10 time steps
Features per time step	7 selected and scaled features
Model input shape	10 x 7 per sequence
Target label	Binary class label
Class meaning	Normal traffic or intrusion-related traffic
Preprocessing operations	Cleaning, encoding, normalization, scaling, and train-test splitting

3.3 Architecture of the Model

The proposed model used a hybrid Convolutional Neural Network and Long Short-Term Memory architecture. The input layer accepted traffic sequences of shape 10 x 7, where 10 represented the number of time steps and 7 represented the selected features at each time step. With this input structure, the model treated traffic as a short sequence rather than as isolated records.

The Convolutional Neural Network branch performed feature extraction on the prepared input sequence. A one-dimensional convolutional layer with 64 filters and a kernel size of 3 scanned the input sequence and produced feature maps representing local relationships among adjacent time steps and traffic attributes. These feature maps helped the model identify compact intrusion-related patterns, such as unusual packet statistics, abnormal flow behaviour, and repeated access indicators. A MaxPooling1D operation with pool size 2 reduced the feature-map dimension while retaining dominant patterns, and the resulting representation was flattened for combination with temporal features.

The Long Short-Term Memory branch performed temporal learning. The first Long Short-Term Memory layer used 64 units and returned sequences so that time-dependent information could be passed to the next recurrent layer. The second Long Short-Term Memory layer used 32 units and condensed the sequence into a fixed-length temporal representation. That structure helped the model capture repeated or gradual activity such as port scanning, denial-of-service traffic, repeated failed access attempts, and gradual tampering behaviour.

The extracted Convolutional Neural Network and Long Short-Term Memory representations were joined in dense fusion layers. The Convolutional Neural Network feature extraction stage produced a 32-dimensional spatial representation, while the Long Short-Term Memory stage produced a 32-dimensional temporal representation. These outputs were concatenated into a 64-dimensional vector and passed through dense layers for final classification. The final sigmoid output layer produced a probability score for binary classification.

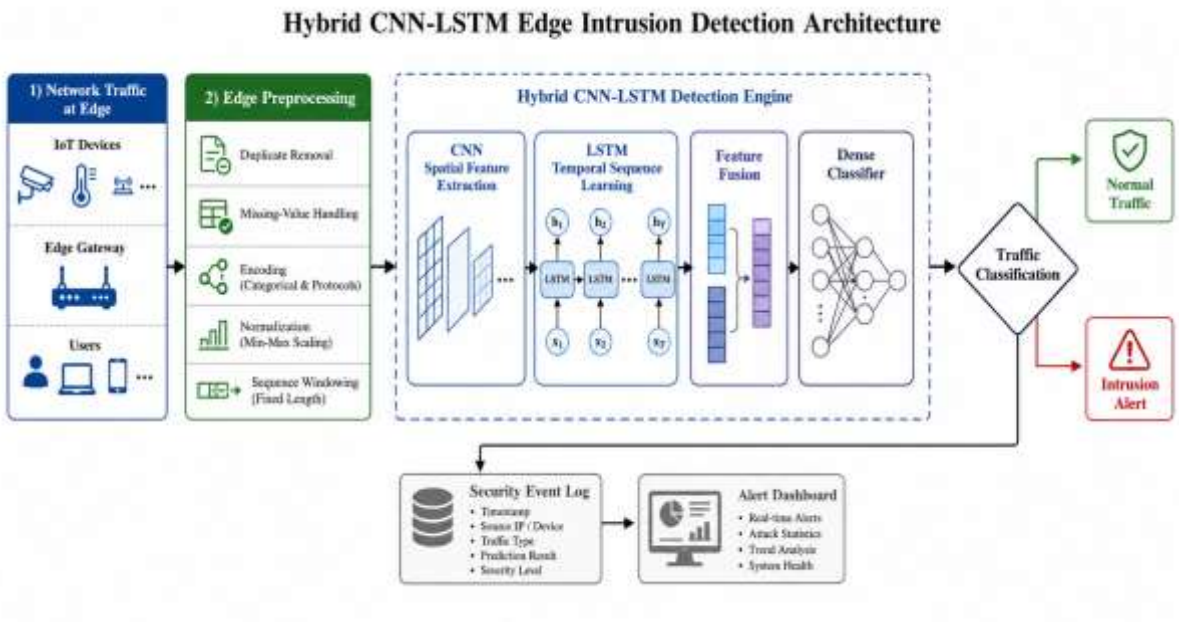

Figure 1: System Architecture

Table 2: Proposed CNN-LSTM Model Structure

Stage	Layer or operation	Main setting	Output or purpose
Input	Sequence input	10 time steps x 7 features	Prepared traffic sequence
CNN branch	Conv1D	64 filters, kernel size 3	Local feature maps from traffic sequences
CNN branch	MaxPooling1D	Pool size 2	Reduced feature-map dimension
CNN branch	Flatten and dense transformation	Dense representation	32-dimensional spatial feature vector
LSTM branch	First Long Short-Term Memory layer	64 units, returns sequences	Temporal sequence representation
LSTM branch	Second Long Short-Term Memory layer	32 units	32-dimensional temporal feature vector
Fusion	Concatenation	32 spatial + 32 temporal features	64-dimensional combined vector
Classification	Dense layers and Softmax output	Two-class classification	Normal or intrusion-related output

Table 3: Model Parameters and Hyperparameters

Parameter	Value
Input shape	10 x 7
Convolution filters	64
Convolution kernel size	3
Pooling size	2
First LSTM units	64
Second LSTM units	32
CNN spatial representation	32 dimensions
LSTM temporal representation	32 dimensions

Concatenated representation	64 dimensions
Dense fusion units	64 and 32
Output activation	Softmax
Optimizer	Adam
Loss function	Categorical cross-entropy
Number of epochs	30
Batch size	32
Learning rate	0.001
Dropout	0.3
Total trainable parameters	38,848

Standalone Convolutional Neural Network and standalone Long Short-Term Memory models were also trained under the same data split and evaluation conditions. The standalone Convolutional Neural Network model used the same preprocessed input to learn feature-level traffic patterns without recurrent temporal modelling. The standalone Long Short-Term Memory model used the same sequence input to learn temporal behaviour without the convolutional feature-extraction branch. These baselines allowed the hybrid architecture to be compared with its two main component structures.

Table 4: Baseline Model Structures Used for Comparison

Baseline model	Structure	Purpose of comparison
Standalone Convolutional Neural Network	Input sequence, convolutional feature extraction, pooling, flattening, dense classification	Tests the effect of feature extraction without Long Short-Term Memory sequence learning
Standalone Long Short-Term Memory	Input sequence, recurrent temporal learning, dense classification	Tests the effect of sequence learning without Convolutional Neural Network feature extraction
Hybrid Convolutional Neural Network and Long Short-Term Memory	Parallel or fused spatial feature extraction and temporal learning, dense classification	Tests whether combined spatial-temporal learning improves detection

Model performance was evaluated using accuracy, precision, recall, F1-score, latency, and response time. Accuracy measured the proportion of correct classifications, precision measured the reliability of predicted intrusion labels, recall measured the proportion of real intrusions detected, and F1-score balanced precision and recall. Latency and response time were used to assess whether the proposed model was suitable for fast edge-level detection.

4.0 Results

The results are presented under three categories: classification performance, operational performance, and file-integrity audit output. Classification performance was measured using accuracy, recall, precision, and F1-score. Operational performance was measured using latency and response time. The file-integrity audit result was reported separately because it belonged to the supporting document-verification component rather than the network-traffic classifier.

4.1 Classification Results

Table 5: Experimental Results of the CNN-LSTM, CNN, and LSTM Models

Model	Accuracy (%)	Recall (%)	Precision (%)	F1-score (%)
CNN-LSTM	96.78	97.44	96.09	96.76
CNN	94.12	95.76	94.96	95.36
LSTM	95.10	95.45	94.70	95.07

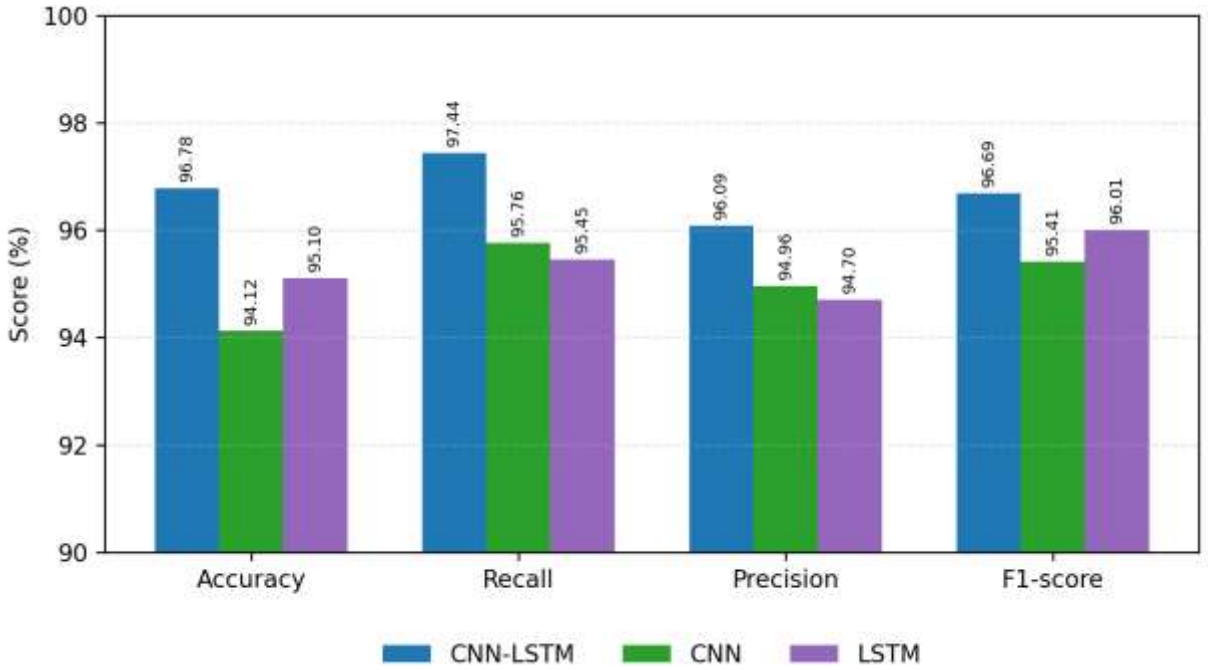

Figure 2: Classification Performance of CNN-LSTM, CNN, and LSTM Models

The CNN-LSTM model recorded 96.78 percent accuracy, 97.44 percent recall, 96.09 percent precision, and 96.76 percent F1-score. The standalone CNN model recorded 94.12 percent accuracy, 95.76 percent recall, 94.96 percent precision, and 95.36 percent F1-score. The standalone LSTM model recorded 95.10 percent accuracy, 95.45 percent recall, 94.70 percent precision, and 95.07 percent F1-score.

4.2 Operational Performance Results

Table 6: Operational Performance Results

Metric	CNN-LSTM Edge Model
Latency (ms)	25
Response time (ms)	35

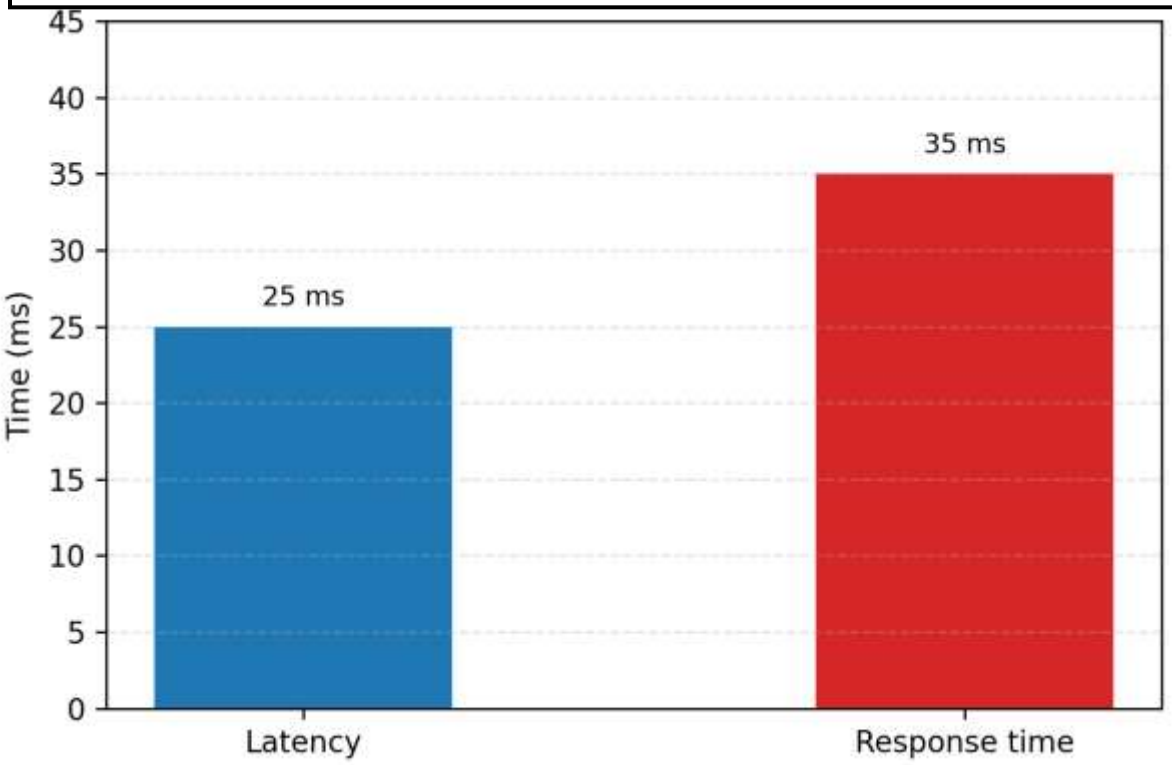

Figure 3: Operational Performance of the CNN-LSTM Edge Model

The operational evaluation recorded 25 ms latency and 35 ms response time in the experimental environment. The implementation also generated intrusion and security event logs containing detected events, source and destination information, timestamps, attack classes, confidence values, and status information.

4.3 File-Integrity Audit Results

The file-integrity audit component was tested with an original Portable Document Format file and a modified test file. The audit output showed different SHA-256 hash values, different page counts, and different total word counts.

Table 7: File-Integrity Audit Output

Audit Item	Original PDF	Test PDF	Result
SHA-256 hash	58d31cfe…e14d0e26	9714d8dc…ed37edd3	Different
Page count	8	1	Different
Total word count	1,319	280	Different
Integrity status	Verified source	Altered file detected	Compromised

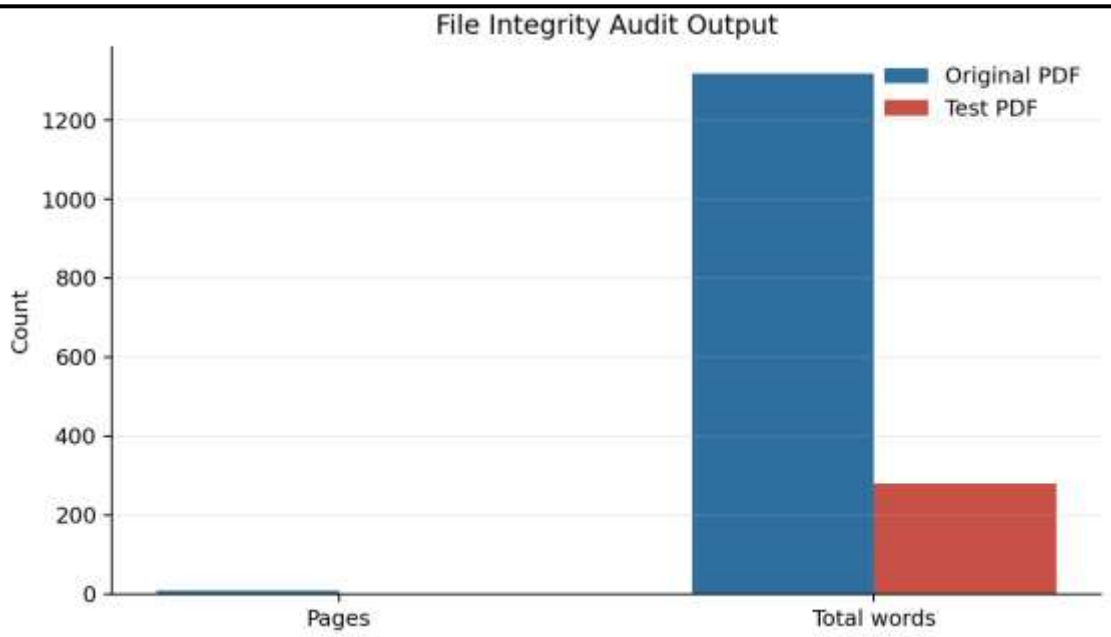

Figure 4: File-Integrity Audit Output

5.0 DISCUSSION

The hybrid CNN-LSTM model performed better than the standalone CNN and LSTM models under the same experimental conditions. The hybrid model achieved 96.78 percent accuracy, 97.44 percent recall, 96.09 percent precision, and 96.76 percent F1-score. In comparison, the standalone CNN model achieved 94.12 percent accuracy and 95.36 percent F1-score, while the standalone LSTM model achieved 95.10 percent accuracy and 95.07 percent F1-score. The comparison suggests that convolutional feature extraction and temporal sequence learning produced a more balanced detector when used together than when either structure was used alone.

The performance gain is linked to the different roles of the two model components. The CNN branch extracted local feature patterns from the prepared traffic sequence, such as relationships among packet statistics, flow duration, protocol behaviour, and other selected traffic indicators. These feature maps helped the model identify compact intrusion-related patterns within the input. The LSTM branch then examined how those patterns changed across time steps, making it useful for activities that unfold gradually, such as port scanning, repeated failed access, denial-of-service behaviour, and progressive tampering. A standalone CNN can miss time-dependent behaviour, while a standalone LSTM may not extract local feature relationships as strongly before sequence modelling. The hybrid structure reduced these individual weaknesses by combining both forms of learning.

Recall is especially important in intrusion detection because a false negative means that an actual attack is missed. The CNN-LSTM model recorded the highest recall value, 97.44 percent, which indicates that it detected a larger proportion of intrusion-related records than the baseline models. Precision was also high at 96.09 percent, showing that the model did not achieve high recall merely by labelling too many records as attacks. The resulting F1-score of 96.76 percent shows that the model maintained a strong balance between detecting attacks and controlling false alerts.

The operational results also support the edge-computing focus of this work. The model recorded 25 ms latency and 35 ms response time in the experimental environment. These values are important because an edge-based intrusion detection system must make decisions close to gateways, routers, local servers, and Internet of Things devices without waiting for every traffic record to be transferred to a central cloud service. Fast local processing can reduce bandwidth pressure and help administrators respond earlier to suspicious activity.

These findings are consistent with earlier studies that reported benefits from hybrid deep-learning architectures. Khan et al. (2019) showed that a Convolutional-LSTM intrusion detection architecture could improve detection by combining convolutional and temporal learning. Altunay and Albayrak (2023) also reported strong results for hybrid Convolutional Neural Network and Long Short-Term Memory models in industrial Internet of Things intrusion detection. The present work follows the same spatial-temporal learning direction, but frames the model around edge-level operation by including latency, response time, alert generation, event logging, and a supporting file-integrity audit component.

The file-integrity audit should be interpreted as an additional edge-security function rather than as part of the CNN-LSTM classifier. Its purpose was to verify whether a document had been altered by comparing SHA-256 hash values, page count, and word count. The audit result showed that the modified test file differed from the original file and was correctly flagged as compromised. That result supports the wider edge-security workflow because edge environments may handle both network traffic and local file uploads. However, the audit result should not be mixed with the network classification metrics.

The results should still be read with some limitations in mind. First, the experiment used a selected subset of 10,000 CICIDS2017 records, so the findings may not fully represent all attack types, traffic volumes, or device behaviours in real edge networks. Second, the evaluation was performed in an experimental software environment rather than on physical edge hardware. As a result, the latency and response-time values should be interpreted as experimental measurements, not final production guarantees. Third, the current evaluation does not yet include broader stress testing under live traffic, adversarial examples, different edge-device capacities, or changing traffic distributions. Those issues should be addressed before full operational deployment.

6.0 CONCLUSION

The work addressed the need for fast and reliable network intrusion detection in edge-computing environments, where gateways, local servers, and Internet of Things devices must respond to suspicious traffic without depending entirely on centralized cloud analysis. A hybrid Convolutional Neural Network and Long Short-Term Memory model was developed to classify normal and malicious traffic by combining convolutional feature extraction with temporal sequence learning.

The model was trained and tested with a selected 10,000-record subset of CICIDS2017 using an 80 percent training and 20 percent testing split. The hybrid model achieved 96.78 percent accuracy, 97.44 percent recall, 96.09 percent precision, and 96.76 percent F1-score, outperforming the standalone Convolutional Neural Network and Long Short-Term Memory baselines. The operational evaluation also recorded 25 ms latency and 35 ms response time, indicating that the approach can support quick local detection in an edge-oriented workflow.

The main contribution is a spatial-temporal intrusion detection design supported by baseline comparison, operational performance reporting, alert and event logging, and a file-integrity audit module. The audit module flagged an altered Portable Document Format file through differences in hash value, page count, and word count, showing how document-integrity checking can complement network-based intrusion detection in an edge-security setting.

The main limitations are the use of a selected CICIDS2017 subset and an experimental software environment rather than a full deployment on physical edge devices. Future work should evaluate the model with larger and more diverse datasets, include more attack classes, test the system on real edge hardware, measure computational cost under live traffic, and assess robustness against adversarial attacks. Overall, the findings show that a hybrid Convolutional Neural Network and Long Short-Term Memory approach can strengthen edge-computing security by supporting accurate, low-latency intrusion detection close to the source of network activity.

REFERENCES

Abbas, N., Zhang, Y., Taherkordi, A., & Skeie, T. (2018). Mobile edge computing: A survey. *IEEE Internet of Things Journal*, *5*(1), 450-465. https://doi.org/10.1109/JIOT.2017.2750180

Altunay, H. C., & Albayrak, Z. (2023). A hybrid CNN+LSTM-based intrusion detection system for industrial IoT networks. *Engineering Science and Technology, an International Journal*, *38*, Article 101322. https://doi.org/10.1016/j.jestch.2022.101322

Berman, D. S., Buczak, A. L., Chavis, J. S., & Corbett, C. L. (2019). A survey of deep learning methods for cyber security. *Information*, *10*(4), Article 122. https://doi.org/10.3390/info10040122

Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, *18*(2), 1153-1176. https://doi.org/10.1109/COMST.2015.2494502

da Costa, K. A. P., Papa, J. P., Lisboa, C. O., Munoz, R., & de Albuquerque, V. H. C. (2019). Internet of Things: A survey on machine learning-based intrusion detection approaches. *Computer Networks*, *151*, 147-157. https://doi.org/10.1016/j.comnet.2019.01.023

Diro, A. A., & Chilamkurti, N. (2018). Distributed attack detection scheme using deep learning approach for Internet of Things. *Future Generation Computer Systems*, *82*, 761-768. https://doi.org/10.1016/j.future.2017.08.043

Farhan, M., Waheed ud din, H., Ullah, S., Khan, M. A., Mazhar, T., Khattak, U. F., Jaghdam, I. H., & Hussain, M. S. (2025). Network-based intrusion detection using deep learning technique. *Scientific Reports*, *15*, Article 25550. https://doi.org/10.1038/s41598-025-08770-0

Ferrag, M. A., Maglaras, L., Moschoyiannis, S., & He, Y. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, *50*, Article 102419. https://doi.org/10.1016/j.jisa.2019.102419

Gamage, S., & Samarabandu, J. (2020). Deep learning methods in network intrusion detection: A survey and an objective comparison. *Journal of Network and Computer Applications*, *169*, Article 102767. https://doi.org/10.1016/j.jnca.2020.102767

Garcia-Teodoro, P., Diaz-Verdejo, J., Macia-Fernandez, G., & Vazquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, *28*(1-2), 18-28. https://doi.org/10.1016/j.cose.2008.08.003

Islam, M. M., Salsabil, U., Nurmamatov, M., & Hossain, S. (2026). Lightweight intrusion detection systems for IoT-edge environments: A PRISMA-ScR systematic review of deployability evidence and a unified assessment framework. *Future Internet*, *18*(6), Article 300. https://doi.org/10.3390/fi18060300

Javaid, A., Niyaz, Q., Sun, W., & Alam, M. (2016). A deep learning approach for network intrusion detection system. *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies*, 21-26. https://doi.org/10.4108/eai.3-12-2015.2262516

Jullian, O., Otero, B., Rodriguez, E., Gutierrez, N., Antona, H., & Canal, R. (2023). Deep-learning based detection for cyber-attacks in IoT networks: A distributed attack detection framework. *Journal of Network and Systems Management*, *31*, Article 33. https://doi.org/10.1007/s10922-023-09722-7

Khan, M. A., Karim, M. R., & Kim, Y. (2019). A scalable and hybrid intrusion detection system based on the Convolutional-LSTM network. *Symmetry*, *11*(4), Article 583. https://doi.org/10.3390/sym11040583

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity*, *2*, Article 20. https://doi.org/10.1186/s42400-019-0038-7

Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. *Future Generation Computer Systems*, *100*, 779-796. https://doi.org/10.1016/j.future.2019.05.041

Liu, H., & Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. *Applied Sciences*, *9*(20), Article 4396. https://doi.org/10.3390/app9204396

Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *2015 Military Communications and Information Systems Conference (MilCIS)*. https://doi.org/10.1109/MilCIS.2015.7348942

Rezapour, R., Asghari, P., Javadi, H. H. S., & Ghanbari, S. (2021). Security in fog computing: A systematic review on issues, challenges and solutions. *Computer Science Review*, *41*, Article 100421. https://doi.org/10.1016/j.cosrev.2021.100421

Roman, R., Lopez, J., & Mambo, M. (2018). Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems*, *78*, 680-698. https://doi.org/10.1016/j.future.2016.11.009

Roy, S., Li, J., Choi, B.-J., & Bai, Y. (2022). A lightweight supervised intrusion detection mechanism for IoT networks. *Future Generation Computer Systems*, *127*, 276-285. https://doi.org/10.1016/j.future.2021.09.027

Sharafaldin, I., Habibi Lashkari, A., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 108-116. https://doi.org/10.5220/0006639801080116

Tamuka, N., Mathonsi, T. E., Olwal, T. O., Maswikaneng, S., Muchenje, T., & Tshilongamulenzhe, T. M. (2026). Intrusion detection in fog computing: A systematic review of security advances and challenges. *Computers*, *15*(3), Article 169. https://doi.org/10.3390/computers15030169

Tsimenidis, S., Lagkas, T., & Rantos, K. (2022). Deep learning in IoT intrusion detection. *Journal of Network and Systems Management*, *30*(1), Article 8. https://doi.org/10.1007/s10922-021-09621-9

Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2019). Deep learning approach for intelligent intrusion detection system. *IEEE Access*, *7*, 41525-41550. https://doi.org/10.1109/ACCESS.2019.2895334