

DEVELOPMENT OF AN IOT-BASED DEEP LEARNING SYSTEM FOR CYBERSECURITY ATTACK DETECTION IN SMART CITY NETWORKS

¹Agubosim Chuka C., ²Ezeibeonu Ozioma Stephanie

Department of Computer Science, Chukwuemeka Odumegwu Ojukwu University, Uli

¹cc.agubosim@coou.edu.ng, ²ezeibeonustephanie@gmail.com

Corresponding Author Email: cc.agubosim@coou.edu.ng

ABSTRACT

The growing use of Internet of Things (IoT) devices in smart cities has expanded the attack surface of cyber-physical systems and increased the need for data-driven intrusion detection. This study implemented and evaluated a hybrid convolutional neural network and long short-term memory (CNN-LSTM) model for cybersecurity attack detection in smart city networks using CICIOT2023 flow-level data. A controlled subset of 15,200 records was formed with 1,900 records each for benign traffic and seven grouped attack families. After cleaning, removal of identifying fields, encoding, and normalization, the subset was divided using a stratified 70% training, 15% validation, and 15% testing split. The CNN-LSTM model was trained with Adam at a learning rate of 0.001, a batch size of 64, and early stopping. On the independent test set, the model achieved 54.61% eight-class accuracy, 61.56% macro precision, 54.61% macro recall, and 54.67% macro F1-score. When the multiclass outputs were collapsed into attack versus benign decisions, 85.56% of malicious records were detected, but the benign false-positive rate was 49.47%. DDoS, DoS, Mirai, and Reconnaissance were detected more reliably than Brute Force, Spoofing, and Web-based attacks. Average batched inference latency was 0.0319 ms per record; this value represents processing speed after a flow record is available and does not measure elapsed time from attack onset to detection. The results therefore indicate preliminary potential for record-level malicious traffic screening, but the current multiclass performance and false-alarm rate are not sufficiently reliable for practical deployment. Further work should include standalone CNN and LSTM ablation tests, deployment-like class distributions, richer temporal context, feature engineering, calibrated decision thresholds, and alternative loss functions or architectures.

KEYWORDS: Internet of Things, Smart City, Cybersecurity, Deep Learning, CNN-LSTM

I. INTRODUCTION

The growth of smart cities leads to the extensive use of Internet of Things (IoT) technologies in transportation, healthcare, power systems, environment monitoring, security, smart buildings, and other areas. IoT enables ubiquitous sensing, communication, data exchange, and control to make cities safer and more efficient. However, interconnected and distributed IoT systems introduce new cybersecurity challenges because compromised sensors, communication channels, gateways, and services can create entry points for malicious actors. Rashid et al. (2020) showed that IoT-based smart cities are subject to various cyber-attacks, and it is critical to develop intelligent systems to differentiate between benign and malicious network traffic. Similarly, Rahman et al. (2020) stated that IoT-based smart cities require scalable intrusion detection systems to secure distributed and heterogeneous networks against emerging threats. The main challenge is that traditional and centralized cybersecurity solutions are ineffective against evolving and sophisticated distributed attacks since attackers can exploit

weaknesses in communication links and other components of the IoT infrastructure (Khan et al., 2021; Ezenwegbu et al., 2025).

Conventional intrusion detection systems rely on signature databases, handcrafted rules, or traditional machine learning algorithms to identify known cyber-attacks. However, such methods are often insufficient when dealing with complex and evolving attacks. As a result, researchers have turned to deep learning algorithms that can learn sophisticated patterns from large-scale traffic datasets. For example, Almiani et al. (2020) developed an intrusion detection framework based on a Recurrent Neural Network (RNN) with an application in IoT and fog environments. Similarly, Saba et al. (2022) proposed a CNN-based anomaly detection system for IoT networks, which can recognize traffic patterns associated with network intrusions. However, timely identification of malicious traffic is critical in IoT networks because attacks can disrupt real-time processes. Vishwakarma and Kesswani (2022) proposed a deep neural network model that can perform real-time intrusion detection in IoT environments.

Researchers have also focused on hybrid deep learning approaches that combine multiple network traffic analysis stages. For example, Convolutional Neural Networks (CNNs) can learn traffic features using filters, while Long Short-Term Memory networks (LSTMs) can process time-series data to learn temporal patterns. Altunay and Albayrak (2023) compared the performance of CNN, LSTM, and a hybrid CNN-LSTM architecture for intrusion detection in Industrial IoT environments. The authors found that a hybrid CNN-LSTM approach showed the best performance when detecting network attacks. Awajan (2023) proposed a deep fully connected network to identify malicious IoT traffic in real-time, aiming to improve security and service reliability. However, the performance of deep learning is highly dependent on the quality and size of the training dataset. In particular, some attack categories can have a much smaller representation than benign traffic, which can increase the likelihood of false-negative instances. Dina et al. (2023) applied a focal loss function to the deep learning model to address the class imbalance problem in IoT traffic classification.

It is evident that multiple deep learning approaches can be used to detect network intrusions in IoT-based smart cities. However, designing a secure and efficient cybersecurity system requires attention not only to classification performance, but also to false alarms, generalization, processing speed, and interpretability. Keshk et al. (2023) emphasized explainability in IoT intrusion detection, while Bakhsh et al. (2023) showed that feed-forward and LSTM-based architectures can identify traffic anomalies. Priyadarshini (2024) further evaluated distributed learning approaches for smart-city IoT cyberattack detection. Accordingly, this study implemented and evaluated an IoT-based CNN-LSTM solution for record-level cybersecurity attack detection using CIIoT2023 traffic features. The primary task was eight-class classification of benign traffic and seven attack families, with a secondary attack-versus-benign analysis and measurement of model inference speed. Because the experimental unit was an already formed flow-level record rather than a time window beginning at attack initiation, the study did not measure time-to-detection from attack onset. Inference latency was therefore interpreted only as computational processing latency and not as evidence of temporal early detection.

II. METHODOLOGY

The study employed an experimental system-development approach consisting of acquisition of a labelled IoT network-traffic dataset, data preprocessing, development of a deep-learning model, model training, testing, and performance evaluation. The labelled dataset contained legitimate network traffic and cybersecurity attack classes. The network-traffic data were examined, categorical targets were encoded, and numerical attributes were normalized to reduce the influence of differing feature scales. A hybrid CNN-LSTM model was implemented to classify network traffic. The CNN component extracted local patterns from the ordered

feature representation, while the LSTM component modeled dependencies within the transformed feature sequence. The model was trained on the training partition and monitored on the validation partition using fixed hyperparameters and early stopping. Final evaluation was performed on the independent test set using accuracy, precision, recall, F1-score, the confusion matrix, attack detection rate, false-positive rate, average batched inference latency, and inference throughput. The experimental objective was to evaluate the model under a controlled class-balanced setting while explicitly distinguishing classification quality from computational speed.

A. Data Collection

The study used the CICIOT2023 IoT network-traffic dataset collected by the Canadian Institute for Cybersecurity at the University of New Brunswick. CICIOT2023 was developed for intrusion-detection and network-security analytics in IoT environments. Unlike many earlier datasets collected from conventional computer networks, CICIOT2023 was generated using an IoT topology comprising 105 IoT devices, with a subset configured as attackers (Neto et al., 2023). The dataset contains malicious IoT traffic directed at other IoT devices and includes 33 attack types that are organized into seven principal attack families: DDoS, DoS, Reconnaissance, Web-based attacks, Brute Force, Spoofing, and Mirai, in addition to benign traffic (Neto et al., 2023). This structure supported the study objective of evaluating multiclass detection across diverse attack families. CICIOT2023 is distributed in packet-capture and extracted-feature forms. This study used the CSV-based flow-level representation for deep learning and machine learning. The available attributes describe characteristics such as flow duration, header length, protocol use, flow rate, TCP flags, packet count, packet size, inter-arrival behavior, variance, covariance, magnitude, radius, and related traffic statistics (Neto et al., 2023). The dataset includes attacks such as UDP, TCP, SYN, ICMP, and HTTP floods, Slowloris, port and vulnerability scanning, host scanning, ARP and DNS spoofing, SQL injection, command injection, cross-site scripting, dictionary brute force, and Mirai variants. For clarity, the study distinguished three dataset levels. First, the original CICIOT2023 corpus contains 33 attack types together with benign traffic. Second, the attack labels used in this experiment were consolidated into seven major attack families, with Benign retained as the eighth target class. Third, after preprocessing and removal of identifying fields, a controlled experimental subset of 15,200 records was formed by applying a fixed quota of 1,900 usable records per class. The equal-class quota was chosen to prevent high-volume classes from dominating optimization and to make class-wise evaluation directly comparable. The 15,200-record subset was then divided using stratified sampling into 10,640 training records, 2,280 validation records, and 2,280 independent test records. This quota-based sample does not preserve the natural prevalence of benign and malicious traffic in operational smart-city networks; therefore, the reported metrics should be interpreted as performance on a controlled balanced benchmark rather than as expected field prevalence.

B. Data Preprocessing

Data preprocessing prepared the flow-level records for training the hybrid CNN-LSTM model. The CSV data used for the experiment were combined into a unified dataset and examined to remove duplicates, irrelevant entries, infinite values, and missing values. The remaining records were separated into input features and target labels. Target categories were encoded as integers so that each traffic record was associated with either the benign class or one of the seven grouped attack families. The numerical attributes were normalized because CICIOT2023 features have different numerical ranges. For example, flow duration, header length, packet rate, inter-arrival measures, covariance, and binary indicators operate on different scales (Neto et al., 2023). Min-max normalization was applied according to Equation (1):

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

where the normalized value was obtained from the original feature value and the minimum and maximum values of the corresponding feature. Normalization was used to reduce the possibility that variables with larger numerical ranges would dominate neural-network optimization. The data were partitioned into training, validation, and testing sets using stratified sampling so that the eight target classes retained equal support in each subset. The training set was used to fit the CNN-LSTM model, the validation set was used for early-stopping decisions and model selection, and the test set was reserved for final performance evaluation. The preprocessed records were reshaped into the array format required by the CNN-LSTM implementation.

C. Implemented Hybrid CNN-LSTM Model

The implemented model used a hybrid CNN-LSTM architecture to classify IoT network traffic into eight classes. Each preprocessed observation contained 46 retained numerical traffic features and was reshaped to an input form of (46, 1). A Conv1D layer with 64 filters, kernel size 3, stride 1, same padding, and ReLU activation learned local interactions across the ordered feature representation. MaxPooling1D with pool size 2 reduced the feature-map dimension, and a dropout rate of 0.20 provided regularization. The pooled representation was passed to an LSTM layer with 64 units, followed by dropout of 0.30 and a dense layer with 32 ReLU units. The output layer contained eight neurons with softmax activation. The model was optimized using Adam with a learning rate of 0.001 and categorical cross-entropy loss, with a batch size of 64 and a maximum of 50 epochs. Early stopping monitored validation loss with patience of five epochs and restored the best weights. Because each model input represented one flow-level record, the LSTM did not observe a chronological sequence of consecutive network flows. It therefore modeled dependencies within the learned feature sequence rather than elapsed temporal behavior across attack onset. This distinction is important when interpreting the model as a record-level classifier rather than a temporal early-warning detector.

The hybrid architecture was selected because the two components provide complementary representation mechanisms. The convolutional stage can learn short-range feature interactions and suppress less informative variation through pooling, while the recurrent stage can retain dependencies across the resulting feature sequence. Altunay and Albayrak (2023) reported advantages for a hybrid CNN-LSTM architecture over individual deep-learning alternatives in an Industrial IoT setting, which provided prior motivation for the design choice. However, the present experiment did not include standalone CNN-only and LSTM-only models trained on the identical 15,200-record split. Therefore, this study does not claim empirical superiority of the hybrid architecture over either component alone. A same-split ablation comparison is required in follow-up work to quantify the contribution of each component.

Table 1: Architecture and Configuration of the Implemented Hybrid CNN-LSTM Model

Layer/Component	Specific Configuration	Output/Activation
Input Layer	46 retained input traffic features	Input shape: (46, 1)
Conv1D	64 filters; kernel size = 3; stride = 1; padding = same	ReLU
MaxPooling1D	Pool size = 2; stride = 2	Maximum pooling
Dropout 1	Rate = 0.20	20% neuron dropout
LSTM	64 hidden units; return_sequences = False	Tanh state activation; sigmoid gates
Dropout 2	Rate = 0.30	30% neuron dropout
Dense Layer	32 neurons	ReLU

Output Layer	8 neurons	Softmax
Loss Function	Categorical cross-entropy	—
Optimizer	Adam	Learning rate = 0.001
Batch Size	64 samples	—
Maximum Epochs	50 epochs	—
Early Stopping	Monitor = validation loss; patience = 5; restore best weights = True	—
Training Split	70%	—
Validation Split	15%	—
Testing Split	15%	—
Evaluation Metrics	Accuracy, precision, recall, F1-score, confusion matrix, detection rate, false- positive rate and average batched inference latency	—

D. Model Training

The CNN-LSTM model was trained using the preprocessed CICIoT2023 subset divided into 70% training, 15% validation, and 15% testing partitions. Training used the Adam optimizer with a learning rate of 0.001, a batch size of 64, and a maximum of 50 epochs. Categorical cross-entropy was used as the loss function, and early stopping with patience of five epochs was applied to reduce overfitting. The validation set monitored model performance during training, while the independent test set was used only for final evaluation. Reported metrics included accuracy, precision, recall, F1-score, the confusion matrix, attack detection rate, false-positive rate, average batched inference latency, and inference throughput.

Algorithm 1: Training Pseudocode

Input: Preprocessed CICIoT2023 dataset

1. Split dataset into 70% training, 15% validation, and 15% testing.
2. Reshape input data for CNN-LSTM.
3. Initialize CNN-LSTM model.
4. Set:
 - Learning rate = 0.001
 - Batch size = 64
 - Maximum epochs = 50
 - Patience = 5
5. Compile model using Adam optimizer and categorical cross-entropy.
6. FOR each epoch:
 - Train model using training data.
 - Evaluate using validation data.
 - IF validation loss improves:
 - Save model weights.
 - ELSE IF no improvement for 5 epochs:
 - Stop training.
7. Restore best model weights.
8. Evaluate model using test data.
9. Compute accuracy, precision, recall, F1-score,
detection rate, false-positive rate, and average batched inference latency.

Output: Trained CNN-LSTM model

E. System Implementation

The cybersecurity attack-detection system was implemented in Python using Keras for development and training of the CNN-LSTM model, while Pandas and NumPy were used for data handling and preprocessing. The preprocessed CICIOT2023 records were supplied to the model as 46-feature inputs. The CNN component extracted local feature interactions, and the LSTM component modeled dependencies in the transformed feature sequence before classification. The final softmax output classified each record into eight classes comprising benign traffic and seven attack families. The hyperparameters used in the implementation are presented in Table 2.

Table 2: Hyperparameters Used for the CNN-LSTM Model

Hyperparameter	Configuration
Conv1D Filters	64
Kernel Size	3
Conv1D Activation	ReLU
Max-Pooling Size	2
CNN Dropout Rate	0.20
LSTM Units	64
LSTM Activation	Tanh
LSTM Dropout Rate	0.30
Dense Layer Neurons	32
Dense Activation	ReLU
Output Neurons	8
Output Activation	Softmax
Optimizer	Adam
Learning Rate	0.001
Loss Function	Categorical Cross-Entropy
Batch Size	64
Maximum Epochs	50
Early-Stopping Patience	5
Training/Validation/Test Split	70% / 15% / 15%

III. SYSTEM RESULTS

The hybrid CNN-LSTM cybersecurity attack-detection model was implemented and evaluated using flow-level records derived from CICIOT2023. The original CICIOT2023 corpus contains 33 attack types organized into the seven major attack families used in this study, together with benign traffic. The experiment used the controlled 15,200-record subset described in Section 2, rather than the full CICIOT2023 class distribution. Experiments were carried out using Keras 3.13.2 with a JAX backend on an AMD EPYC CPU, using the architecture and hyperparameters specified in Section 2.

A. Experimental Dataset Distribution

The experimental subset was intentionally class-balanced. After preprocessing, the source IP address, destination IP address, flow ID, and timestamp were excluded from the predictors to reduce reliance on identifying or dataset-specific fields. Seventy-nine numerical fields were reduced to 46 retained features. A fixed quota of 1,900 usable records was included for each of the eight target classes, producing 15,200 records in total. This quota-based design equalized class support for controlled comparison, but it did not reproduce the prevalence of real smart-

city traffic. In particular, although the eight multiclass categories were equally represented, seven of the eight classes were malicious. Consequently, the attack-versus-benign view of the same subset had an implicit 7:1 malicious-to-benign class ratio, a factor that is relevant to interpretation of the high benign false-positive rate.

Table 3: Distribution of the Dataset Used for Model Implementation

Traffic Class	Records	Percentage (%)
Benign	1,900	12.50
Brute Force	1,900	12.50
DDoS	1,900	12.50
DoS	1,900	12.50
Mirai	1,900	12.50
Reconnaissance	1,900	12.50
Spoofing	1,900	12.50
Web-based	1,900	12.50
Total	15,200	100.00

The stratified 70:15:15 split contained 10,640 training instances, 2,280 validation instances, and 2,280 independent test records. Each partition retained equal representation of the eight target classes; therefore, the test set contained 285 unseen observations per class.

B. Model Training Results

The CNN-LSTM model had 35624 trainable parameters. The model was trained using Adam optimizer with a learning rate of 0.001 and a batch size of 64. The maximum number of epochs was set at 50, but the training process was stopped early at epoch 47. This was because the validation loss did not improve for five consecutive epochs after achieving the lowest loss of 1.2257 at epoch 42. The model had a validation accuracy of 55.31% at epoch 42. Thus, the weights from epoch 42 were used to generate the final predictions.

Table 4: Selected CNN-LSTM Training Results

Epoch	Training Loss	Training Accuracy (%)	Validation Loss	Validation Accuracy (%)
1	1.9056	24.08	1.8167	30.83
10	1.4233	47.01	1.3909	46.93
20	1.3195	50.21	1.3076	51.05
30	1.2601	52.55	1.2592	53.77
40	1.2198	54.13	1.2468	54.61
42	1.2091	54.02	1.2257	55.31
47	1.1996	54.48	1.2509	54.12

Training and validation loss generally decreased during learning, while training accuracy increased from 24.08% in the first epoch to approximately 54% near the end of training. Validation accuracy followed a similar pattern and reached its best value of 55.31% at epoch 42. The relatively small gap between the training and validation curves suggests that severe overfitting was not the dominant problem. Instead, the stabilization of both curves around 54% to 55% indicates limited discriminative capacity for the eight-class problem, consistent with overlap among several attack-family feature distributions and the restricted contextual information available to the model.

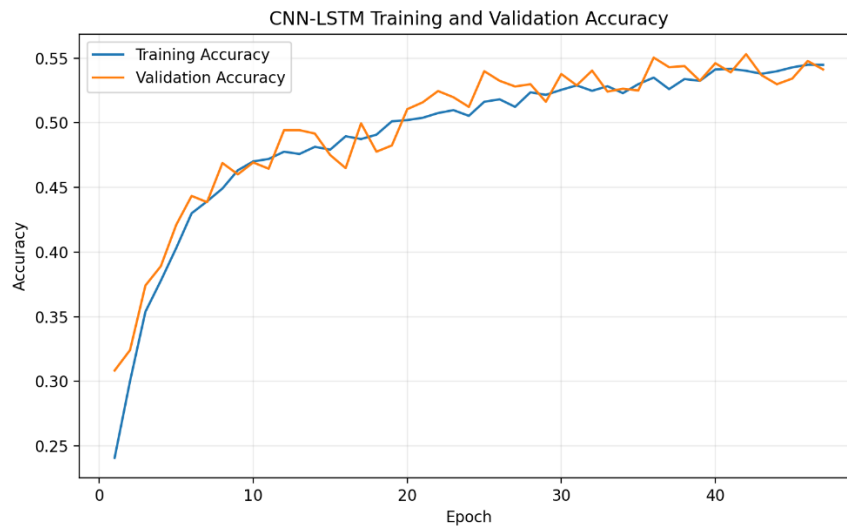


Figure 1: Training and Validation Accuracy of the CNN-LSTM Model

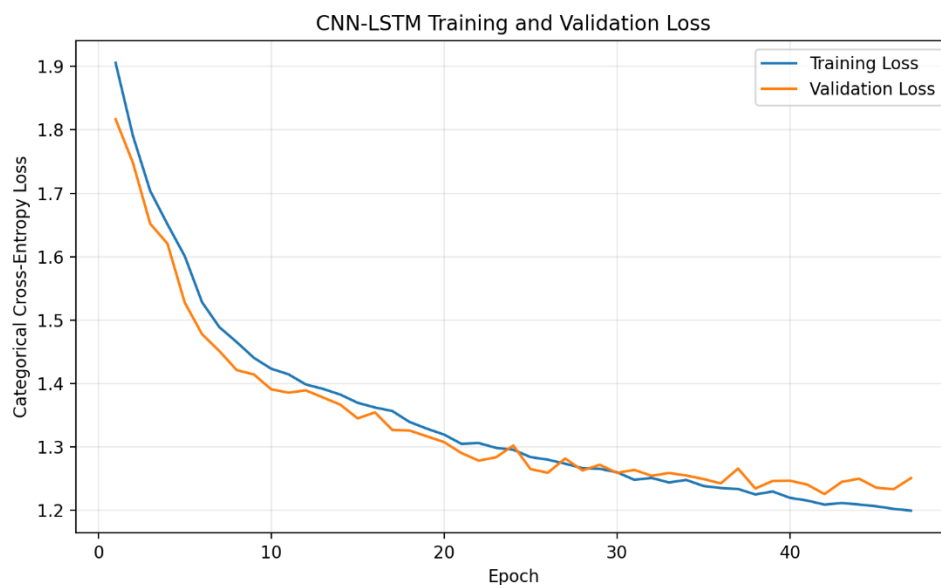


Figure 2: Training and Validation Loss of the CNN-LSTM Model

C. Overall Model Performance

The restored CNN-LSTM model was evaluated using the 2,280 observations in the test dataset that were not used during model training or validation. The model produced a test accuracy of 54.61%, macro precision of 61.56%, macro recall of 54.61%, and macro F1-score of 54.67%. The final categorical cross-entropy test loss was 1.2460. When the predictions were reduced to the fundamental intrusion-detection decision of attack versus benign traffic, the system detected 85.56% of malicious observations. However, the benign false-positive rate was 49.47%, indicating that the model generated a considerable number of false alarms.

Table 5: Overall Performance of the CNN-LSTM Model

Performance Metric	Experimental Result
Test Accuracy	54.61%
Macro Precision	61.56%
Macro Recall	54.61%

Macro F1-Score	54.67%
Weighted Precision	61.56%
Weighted Recall	54.61%
Weighted F1-Score	54.67%
Attack Detection Rate	85.56%
Benign False-Positive Rate	49.47%
Test Loss	1.2460
Best Validation Accuracy	55.31%
Best Epoch	42
Training Stopped	Epoch 47
Average Batched Inference Latency	0.0319 ms/record
Inference Throughput	≈31,337 records/s
Trainable Parameters	35,624

The gap between the multiclass accuracy of 54.61% and the attack detection rate of 85.56% is important. Multiclass accuracy required the model to identify the correct one of eight traffic classes, whereas the secondary attack-versus-benign analysis counted any predicted malicious family as a successful attack detection. The higher attack detection rate therefore indicates that the model was better at distinguishing many malicious records from benign traffic than at assigning the correct attack family. This result should not be interpreted as deployment readiness because almost half of the benign test records were also flagged as attacks.

D. Class-Wise Detection Performance

Table 6 below contains the precision, recall, and F1-score for each category. It can be seen that traffic of DDoS types produced the best results for all three metrics. Specifically, the precision was 97.79%, recall was 77.54%, and F1-score was 86.50%. The DoS types demonstrated a similar pattern, with the F1-score of 70.18%. Finally, reconnaissance had the best recall (80.35%) but a relatively low precision (47.12%).

Table 6: Class-Wise Performance of the CNN-LSTM

Traffic Class	Precision (%)	Recall/Detection Rate (%)	F1-Score (%)	Support
Benign	33.33	50.53	40.17	285
Brute Force	57.65	34.39	43.08	285
DDoS	97.79	77.54	86.50	285
DoS	93.57	56.14	70.18	285
Mirai	47.33	71.58	56.98	285
Reconnaissance	47.12	80.35	59.40	285
Spoofing	42.69	37.89	40.15	285
Web-based	72.97	28.42	40.91	285
Macro Average	61.56	54.61	54.67	2,280

Web-based attacks achieved relatively high precision of 72.97% but low recall of 28.42%, showing that positive Web-based predictions were often correct while most actual Web-based attacks were missed. Brute Force and Spoofing also had low recall values of 34.39% and 37.89%, respectively. By contrast, DDoS achieved the strongest F1-score, and Reconnaissance achieved the highest recall. The results indicate that the current feature representation separates high-volume or strongly patterned traffic more effectively than classes whose flow-level characteristics overlap with benign or other attack traffic.

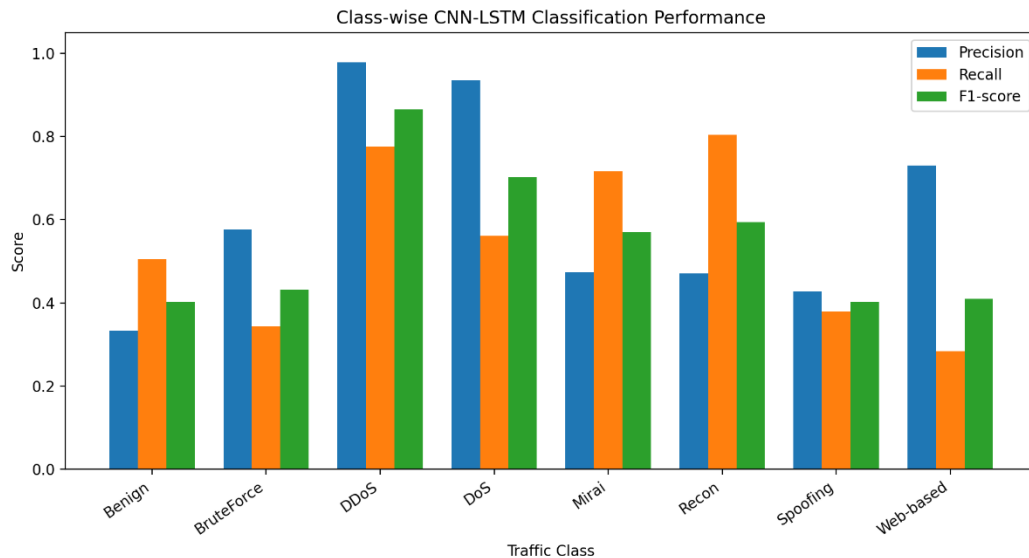


Figure 3: Class-Wise Precision, Recall and F1-Score of the CNN-LSTM Model

E. Confusion Matrix Analysis

The detailed confusion matrix obtained from the test dataset is presented in Figure 4. Each class contained 285 actual observations.

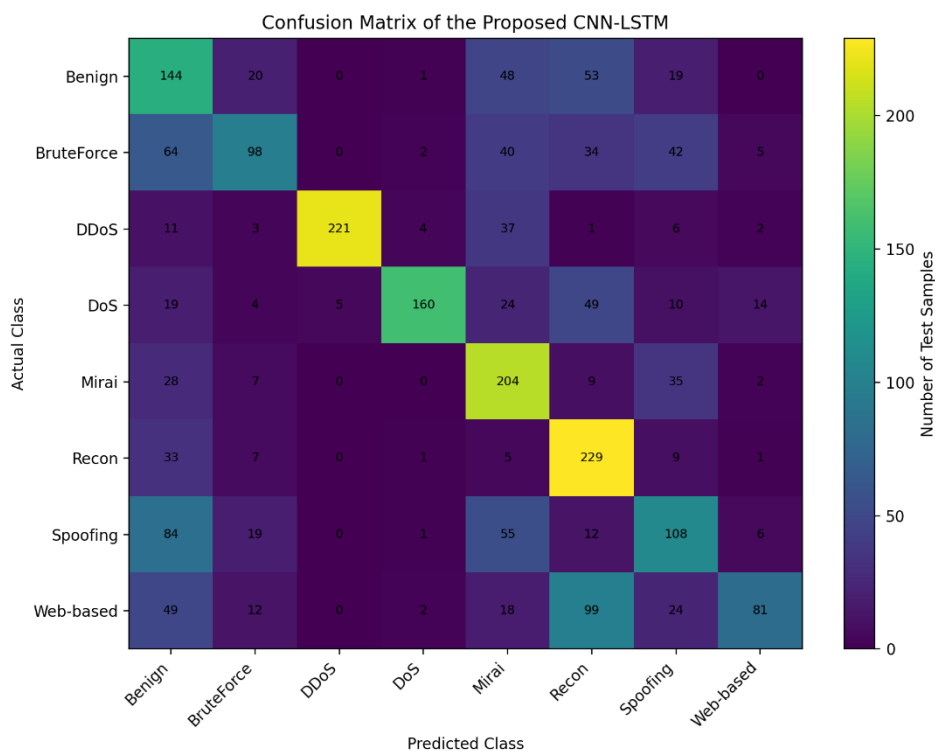


Figure 4: Confusion Matrix of the CNN-LSTM Model

The confusion matrix provides additional detail on class-specific errors. DDoS had 221 correct predictions out of 285 observations, Reconnaissance had 229, and Mirai had 204. In contrast, only 98 Brute Force, 108 Spoofing, and 81 Web-based observations were assigned to their correct classes. Web-based traffic was frequently confused with Reconnaissance, including 99 Web-based observations predicted as Reconnaissance. Spoofing was also confused with Benign traffic and Mirai, with 84 Spoofing observations predicted as Benign and 55 predicted

as Mirai. These errors explain the weak recall of the affected classes and show that the principal limitation is not uniform across all attack families.

These confusion patterns are consistent with limitations in both the selected predictors and the record-level architecture. Source and destination IP addresses, flow ID, and timestamp were removed, which reduced the risk that the model would memorize dataset-specific identifiers but also removed identity and timing context that can help distinguish scanning, spoofing, and Web-related behavior. In addition, each 46-feature flow record was classified independently; the LSTM did not receive a chronological sequence of neighboring flows. As a result, attacks that require contextual or temporal evidence may share similar aggregate statistics with other classes. The Web-based-to-Reconnaissance and Spoofing-to-Benign or Mirai errors therefore suggest a need for richer non-identifying temporal windows, protocol-aware features, and targeted feature engineering rather than simply increasing inference speed.

F. Attack-versus-Benign Detection and Inference Latency

For a secondary intrusion-screening analysis, all seven malicious classes were pooled as “attack” after the multiclass predictions had been generated. Of 1,995 malicious observations in the test set, 1,707 were predicted as one of the attack classes and 288 were predicted as Benign, producing an attack detection rate of 85.56%. Of the 285 benign observations, 144 were correctly predicted as Benign and 141 were incorrectly assigned to an attack class, producing a benign false-positive rate of 49.47%. This binary aggregation measures whether a completed flow record was assigned to any malicious class; it does not indicate how soon after attack initiation the record became detectable.

Table 7: Attack-versus-Benign Detection Results

Detection Outcome	Number of Records
Attacks correctly identified as malicious	1,707
Attacks incorrectly identified as benign	288
Benign traffic correctly identified	144
Benign traffic incorrectly flagged as attack	141
Attack Detection Rate	85.56%
False-Positive Rate	49.47%

Average batched inference time after model warm-up was approximately 0.0319 ms per record, corresponding to about 31,337 records per second in the reported execution environment. This measurement describes computational inference speed after a preprocessed flow record is available. It is not a time-to-detection measure from attack onset because the experiment did not construct chronological detection windows or retain timestamps for temporal onset analysis. Therefore, the latency result supports computational feasibility only and should not be used to claim temporal early detection. In practical operation, the 49.47% false-positive rate is the more serious limitation because it would generate a large number of unnecessary alerts. Taken together, the results show that the CNN-LSTM model learned useful distinctions in the controlled dataset but remained unreliable for eight-class attack-family identification. The model achieved 54.61% accuracy and a 54.67% macro F1-score, while the secondary attack-versus-benign aggregation detected 85.56% of malicious records. DDoS, DoS, Mirai, and Reconnaissance were easier to recognize than Benign, Brute Force, Spoofing, and Web-based traffic. The performance profile indicates preliminary value for record-level experimentation, but the false-positive rate and multiclass errors prevent a claim of operational effectiveness.

G. Practical Limitations and Targeted Improvement Directions

The 49.47% benign false-positive rate can plausibly be explained by several interacting factors. First, the balanced eight-class design gave equal support to each class, but seven classes were attacks and only one was benign. When viewed as a binary task, the training and test distributions therefore contained 87.5% attack records and 12.5% benign records, which can favor assignment to one of the malicious classes. Second, identifier and timestamp removal limited contextual cues, while classification of each flow independently prevented the model from observing neighboring-flow behavior. Third, a forced eight-way softmax decision provides no rejection option for uncertain records. These factors can produce false alarms even when multiclass class counts are balanced. Future improvement should therefore be error-driven. A two-stage detector is a strong candidate: the first stage can be trained specifically for benign-versus-attack discrimination with balanced binary support and calibrated thresholds, while the second stage can classify confirmed attacks into families. More diverse benign traffic and deployment-like class priors should be included in evaluation. Feature engineering should emphasize protocol behavior, rate and burst statistics, and non-identifying temporal windows. Focal loss may help concentrate learning on hard examples such as Brute Force, Spoofing, and Web-based attacks; ordinary inverse-frequency class weighting is less relevant to the present equal-support eight-class subset, but cost-sensitive weighting can be evaluated when the full imbalanced dataset is used. Attention-based or alternative sequence architectures can also be tested. Finally, standalone CNN and LSTM baselines should be trained on the identical split so that the incremental value of the hybrid design can be quantified rather than assumed.

IV. CONCLUSION

This study implemented and evaluated a hybrid CNN-LSTM model for cybersecurity attack detection in smart-city IoT network traffic using a controlled 15,200-record subset of CICIoT2023. The model classified benign traffic and seven grouped attack families from 46 retained flow-level features. On the independent test set, it achieved 54.61% multiclass accuracy, 61.56% macro precision, 54.61% macro recall, and 54.67% macro F1-score. Class-wise analysis showed stronger performance for DDoS, DoS, Mirai, and Reconnaissance than for Brute Force, Spoofing, Web-based, and Benign traffic. When predictions were collapsed into attack versus benign categories, 85.56% of malicious records were identified as attacks, but the benign false-positive rate was 49.47%. This combination indicates preliminary promise for record-level malicious traffic screening but is not sufficiently reliable for practical deployment. The measured 0.0319 ms per-record batched inference latency reflects processing speed only and does not establish temporal early detection from attack onset. The present study also did not include same-split CNN-only and LSTM-only ablation models, so no empirical superiority claim is made for the hybrid architecture. Future work should evaluate deployment-like traffic distributions, more representative benign traffic, calibrated or hierarchical decision strategies, richer temporal context, focal or cost-sensitive objectives, alternative architectures, and controlled ablation baselines. Reducing false alarms and improving weak attack-family recall are the primary requirements before operational use can be considered.

REFERENCES

- Almiani, M., AbuGhazleh, A., Al-Rahayfeh, A., Atiewi, S., & Razaque, A. (2020). Deep recurrent neural network for IoT intrusion detection system. *Simulation Modelling Practice and Theory*, 101, 102031. doi:10.1016/j.simpat.2019.102031.
- 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, 101322. doi:10.1016/j.jestech.2022.101322.
- Awajan, A. (2023). A novel deep learning-based intrusion detection system for IoT networks. *Computers*, 12(2), 34. doi:10.3390/computers12020034.

- Bakhsh, S. A., Khan, M. A., Ahmed, F., Alshehri, M. S., Ali, H., & Ahmad, J. (2023). Enhancing IoT network security through deep learning-powered intrusion detection system. *Internet of Things*, 24, 100936. doi:10.1016/j.iot.2023.100936.
- Dina, A. S., Siddique, A. B., & Manivannan, D. (2023). A deep learning approach for intrusion detection in Internet of Things using focal loss function. *Internet of Things*, 22, 100699. doi:10.1016/j.iot.2023.100699.
- Ezenwegbu, N. C., Gbasouzor, A. I., Akaho, A. A., Okeke, O. C., & Langat, C. E. (2025). Economic replacement of plants and equipment: A decision-making framework in engineering. *Advances in Science, Technology and Engineering Systems Journal*, 10(5), 20–32. <https://doi.org/10.25046/aj100503>
- Keshk, M., Koroniotis, N., Pham, N., Moustafa, N., Turnbull, B., & Zomaya, A. Y. (2023). An explainable deep learning-enabled intrusion detection framework in IoT networks. *Information Sciences*, 639, 119000. doi:10.1016/j.ins.2023.119000.
- Khan, M. A., Khan, M. A., Jan, S. U., Ahmad, J., Jamal, S. S., Shah, A. A., Pitropakis, N., & Buchanan, W. J. (2021). A deep learning-based intrusion detection system for MQTT enabled IoT. *Sensors*, 21(21), 7016. doi:10.3390/s21217016.
- Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., & Ghorbani, A. A. (2023). CICIOT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. *Sensors*, 23(13), 5941. <https://doi.org/10.3390/s23135941>.
- Priyadarshini, I. (2024). Anomaly detection of IoT cyberattacks in smart cities using federated learning and split learning. *Big Data and Cognitive Computing*, 8(3), 21. doi:10.3390/bdcc8030021.
- Rahman, M. A., Asyhari, A. T., Leong, L. S., Satrya, G. B., Tao, M. H., & Zolkipli, M. F. (2020). Scalable machine learning-based intrusion detection system for IoT-enabled smart cities. *Sustainable Cities and Society*, 61, 102324. doi:10.1016/j.scs.2020.102324.
- Rashid, M. M., Kamruzzaman, J., Hassan, M. M., Imam, T., & Gordon, S. (2020). Cyberattacks detection in IoT-based smart city applications using machine learning techniques. *International Journal of Environmental Research and Public Health*, 17(24), 9347. doi:10.3390/ijerph17249347.
- Saba, T., Rehman, A., Sadad, T., Kolivand, H., & Bahaj, S. A. (2022). Anomaly-based intrusion detection system for IoT networks through deep learning model. *Computers & Electrical Engineering*, 99, 107810. doi:10.1016/j.compeleceng.2022.107810.
- Vishwakarma, M., & Kesswani, N. (2022). DIDS: A deep neural network based real-time intrusion detection system for IoT. *Decision Analytics Journal*, 5, 100142. doi:10.1016/j.dajour.2022.100142.