International Journal of Research in Advanced Engineering and Technology
www.allengineeringjournal.in
ISSN: 2455-0876
Received: 20-07-2026, Accepted: 22-08-2026, Published: 07-09-2026
Volume 12, Issue 3, 2026, Page No. 28-33

An edge-based CNN-LSTM hybrid model for real-time intrusion detection in industrial internet of things (IIoT) environments

Yasameen A Muhsin
Department of Education, Al-Qurna Education Department, General Directorate of Education of Basra Governorate, Ministry of Education, Basra, Iraq
DOI: https://doi.org/10.66856/ijraet.2026.12.3.12024

Abstract
Industrial Internet of Things (IIoT) environments have witnessed increasing expansion due to the growing reliance on connected devices, sensors, and intelligent systems in industrial monitoring, control, and automation. Despite the advantages these environments offer, the proliferation of devices and protocols, along with the expansion of connectivity, leads to a wider attack surface and escalating security challenges. This underscores the importance of developing intrusion detection systems capable of effectively analyzing network traffic and detecting malicious activity. This study aims to design and evaluate a hybrid model based on Convolutional Neural Networks (CNNs) and Long Short-Term Memory Networks (LSTMs) for intrusion detection in IIoT environments. The model leverages CNNs' ability to extract patterns and LSTMs' ability to process temporal relationships in sequential data. The study utilized the DNN-EdgeIIoT-dataset associated with the Edge-IIoTset, a dataset specifically designed to evaluate cybersecurity technologies in IoT and IIoT applications. Data processing involved verifying timestamps and retaining valid records dating back to 2021. 43 network features were then selected for model building. After addressing missing values and standardizing features using StandardScaler, the records were sorted chronologically and divided into 70% for training, 15% for validation, and 15% for testing. The results indicate that the model has a high capacity for reducing false positives and maintaining high precision; however, low recall is the main limitation of its current performance. Therefore, improving the model's attack detection capabilities is a priority for future work. The inference speed also suggests the potential for studying the model in the context of edge applications, although future testing on actual edge devices is necessary to confirm this.

Keywords: Intrusion Detection, Deep Learning, CNN, LSTM, CNN-LSTM, Edge Computing, Edge-IIoTset

Introduction

Recent years have witnessed rapid development in the use of the Industrial Internet of Things (IIoT), with industrial devices, sensors, and systems becoming interconnected through digital communication networks. This has contributed to improved monitoring, control, and industrial automation processes, as well as providing vast amounts of data that can be used to support decision-making [1]. However, this increasing interconnection has simultaneously led to increased security risks due to the proliferation of devices, protocols, and connection points. This has made IIoT networks a potential target for a variety of cyberattacks [2, 3]. Intrusion detection systems are among the most important security mechanisms used to monitor network traffic and detect abnormal or malicious activity. As attack methods evolve, relying solely on static rules has become increasingly difficult, leading to a growing interest in using machine learning and deep learning techniques [3, 4]. Convolutional neural networks (CNNs) offer the ability to extract patterns from data, while LSTM excels at handling sequential data and temporal relationships. Therefore, combining these two techniques can provide a suitable model for analyzing network traffic in IIoT environments [5]. IIoT environments generate large and diverse amounts of network data, including both natural patterns and patterns associated with cyberattacks. The challenge lies in the fact that some attacks may not be revealed through a single record but rather through a series of chronologically sequential events. Furthermore, intrusion detection systems need to strike a balance between detecting the maximum number of attacks and minimizing false alarms. Accordingly, this study focuses on evaluating a CNN-LSTM model for detecting attacks in IIoT data while preserving the temporal dimension of the data, and measuring its performance using a variety of metrics.

Studies have shown that deep learning techniques have become important methods in developing intrusion detection systems due to their ability to extract complex representations from data [2, 3]. Scientific reviews have addressed a large number of methods that rely on CNNs, RNNs, LSTMs, and others in analyzing IoT data and detecting anomalous activity [4, 6, 7]. The Edge-IIoTset is an important dataset in the field of IoT and IIoT security. It was designed to provide realistic and relatively comprehensive data for evaluating intrusion detection techniques [1]. The set includes data related to a number of devices, protocols, and attack scenarios, making it suitable for evaluating machine learning and deep learning algorithms. The proposed evaluation framework in the current study utilizes the CNN-LSTM architecture to evaluate the learning of patterns and temporal dependencies in sequential IIoT traffic. The novelty of this study thus lies in the evaluation framework and the investigation of generalization under chronological traffic splitting, rather than in introducing a novel neural network architecture. Previous studies have explored the use of the CNN+LSTM model for intrusion detection in IIoT networks, supporting

the potential application of this type of architecture in the security field [5].

Despite the numerous previous studies, the varying data preparation, segmentation, and evaluation metrics make comparing models complex. Therefore, the current study focuses on a specific practical experiment using time-based data segmentation, with a comprehensive evaluation that includes performance metrics, a confusion matrix, and software-level inference speed. While CNN-LSTM approaches have already been studied for detecting intrusions in Industrial IoT systems, the problem of the generalization of such methods with chronological traffic splitting has not been explored sufficiently. Specifically, it is required to investigate whether the good results on validation can be preserved when the network is tested on the traffic which comes later and when there is not enough training data for the attack class. Consequently, this paper is devoted to closing the research gap connected with testing. This research aims to design and evaluate a CNN-LSTM hybrid model for intrusion detection in IIoT environments. Objectives of this study include: pre-processing the temporally sorted IIoT traffic; building CNN-LSTM models using 43 network features; assessing the model's binary intrusion detection effectiveness; examining the confusion matrix; measuring the inference throughput at the software level; and evaluating the limitations of the model, particularly in regard to attack recall.

Methodology

Data Used

The DNN-EdgeIIoT-dataset.csv file was used. The original number of records was: 2,219,201 records. After excluding invalid records or those not from 2021, the number became: 2,096,419 records. The final time range was: 2021-01-01 00:00:01.163427 – 2021-01-01 23:59:59.358821.

Features

Forty-three network features were used. The properties included ARP, ICMP, HTTP, TCP, UDP, DNS, MQTT, and Modbus/TCP protocols.

Data Processing

The features were converted to numerical values, and then missing values were processed using the median calculated from the training data only. After processing, training missing values = 0, validation missing values = 0, and test missing values = 0.

Time Division

After sorting chronologically, the valid data was split into training, validation, and testing sets. The training data had 1,467,493 records, which is 70% of the total data. On the other hand, the validation and testing data had 314,463 records each, totaling to 2,096,419 (Table 1).

Table 1: Chronological data partitioning

Partition	Records	Ratio
Training	1,467,493	70%
Validation	314,463	15%
Test	314,463	15%
Total	2,096,419	100%

Feature Scaling

Once the data had been divided by time, the input sequences were created separately within each split. The input sequences had ten data points each, and a stride of one data point was used for subsequent sequences. The input sequences were independently created in each time period, and no sequence spanned across the boundary between the training, validation, and test periods. None of the sequences spanned across the boundary between the splits for training, validation, and testing. The medians used for filling missing values and the parameters used for scaling the features were determined solely from the training set. The same parameters were then used for the validation and testing sets without fitting. This was done in order to prevent any data from the evaluation sets to affect the modeling process.

Building a CNN-LSTM Model

A hybrid model combining CNN and LSTM was built to extract patterns from the inputs and process the temporal relationships between successive records. The total number of parameters in the model contained 43,713 parameters.

Results

Training Results

The accuracies and losses of the CNN–LSTM model in training and validation phases throughout ten epochs are shown in (Fig. 1 & 2). As can be seen from (Fig. 1), the accuracy during training grew progressively from about 93.27% in the first epoch to 94.37% in the tenth epoch. The validation accuracy outperformed the training one most of the time, oscillating from about 94.07% to 94.59%, reaching the maximum value at the ninth epoch. The curves show quite a similar trend, which means the model is optimized in a relatively stable way. However, the drop of validation accuracy in the fifth epoch clearly indicates the lack of monotonicity in the validation phase. (Fig. 2) shows the training and validation loss curves of the model. The training loss reduced monotonically from about 0.169 in the first epoch to 0.139 in the tenth epoch. The validation loss reduced in the initial epochs, experienced fluctuations in the middle of training and reached its minimum value of about 0.143 in the ninth epoch. The validation loss was slightly larger than the training loss in the latter epochs but did not grow monotonically. This implies that the optimization process is rather stable, but there is no indication of overfitting yet. However, it is worth mentioning that the training curves alone do not prove satisfactory performance on new data.

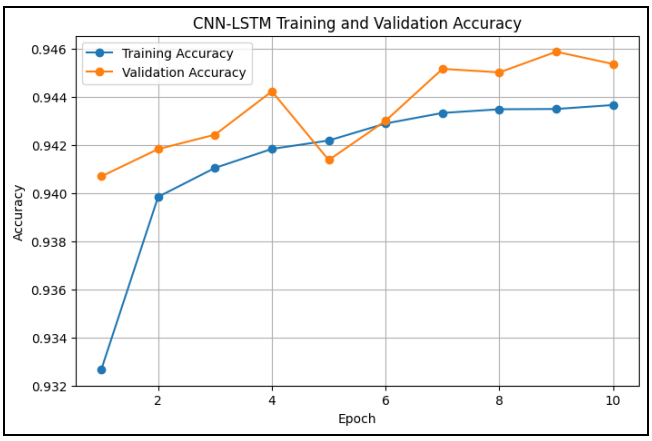

Fig 1: CNN-LSTM training and validation accuracy

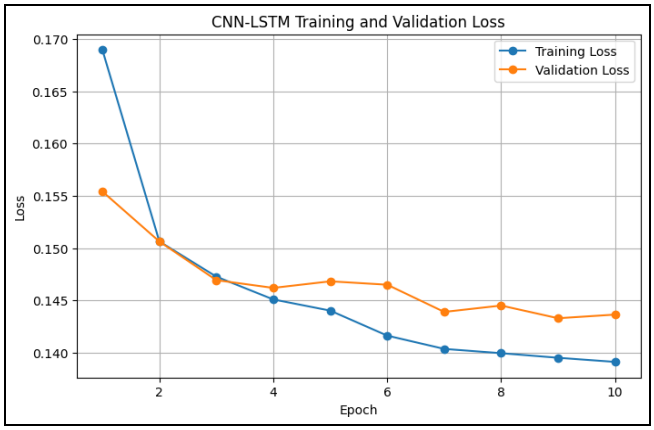

Fig 2: CNN-LSTM training and validation loss

The gap between the validation performance and the test performance needs further investigation. This difference may be attributed to temporal distribution shift, different attacks prevalence, differences in sequences creation, or low-represented traffic patterns during the training process. Difference between validation and testing performance must be seen as reflecting lack of temporal generalization, rather than proof of a particular cause. Possible causes could be shift in temporal distribution, changes in attack rate, selection of sequences, and insufficient sampling of particular traffic types; all of which need to be studied separately. Chronological splitting has its advantages for evaluation since it better mimics the situation when the model is deployed but it increases the difficulty of evaluation as well and necessitates explicit mentioning of the time interval and distribution of classes of each set.

Test Results

The results indicate high Precision and low Recall, demonstrating that the model achieves relatively high accuracy when issuing an alarm, but it does not detect a sufficient percentage of actual attacks. The analysis of the results obtained from testing shows great disparity between the capabilities of the model in preventing false positives and detecting attacks (Table 2). The achieved precision of 97.37% means that most of the sequences marked as attacks turned out to be actually malicious, and it is a good thing in the situation where unnecessary alarms put extra workload on the security specialists. The rate of 22.96% for recall indicates that there was significant misclassification of attacks as normal behavior. Therefore, the primary problem with the current model is that it does not provide sufficient coverage of attacks and not false positives. Therefore, the achieved accuracy of 60.24% cannot be used as the main indicator of performance, especially in case of imbalance of classes.

Table 2: Test-Set performance of the CNN–LSTM model

Matric	Value
Accuracy	60.2421 %
Precision	97.3709 %
Recall	22.9600 %
F1-score	37.1582 %
ROC-AUC	0.919359
PR-AUC	0.892100
False-positive rate	0.6503%
Attack detection rate	22.9600%

(Table 2) provides the performance measures that were calculated based on the results obtained at the level of prediction as shown in (Fig. 3). In this analysis, attack was considered to be the positive class while the values for precision, recall, F1-score, and false positives rate were determined from the confusion matrix.

Confusion Matrix

As shown in (Fig. 3), the confusion matrix at prediction level for the test sequences is provided. There were 152,471 accurate predictions for normal cases and 36,962 for attack cases. The number of false positives and false negatives was 998 and 124,022, respectively. The relatively small number of false positives is in accordance with the false positive rate of 0.6503% and the precision rate of 97.3709%. In contrast, a relatively large number of false negatives is associated with the recall rate of 22.9600%. These results show that the algorithm uses a cautious way of assigning an attack class. This makes the frequency of false alarm occurrences to decrease but at the same time makes a number of attacks go undetected. This implies that the algorithm cannot be seen as the only intrusion detection solution until the recall of attacks is improved.

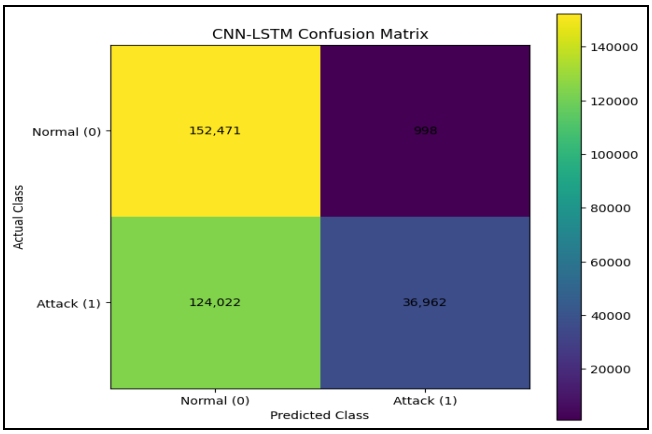

Fig 3: Prediction-Level confusion matrix of the CNN–LSTM model on the test sequences

Inference Speed

The analysis of the inference time reveals that the software-based solution is able to analyze a large amount of preprocessed sequences in the experimental conditions. This claimed measurement is applicable only for inferences on preprocessed sequences. This measurement does not include capturing packets, feature extraction, preprocessing, buffering of sequences, or alarm forwarding. Nevertheless, the throughput of the system does not guarantee its ability to operate in real-time regime in the IIoT edge setting. First of all, it is necessary to separate the average time per sequence from the latency, which also comprises the sequence capture, feature extraction, sorting timestamps, handling missing values, scaling, buffering sequences, inference, and sending alarms. Furthermore, the experiment was carried out in Google Colab environment instead of being conducted on resource-limited gateways or industrial controllers. Consequently, the edge viability of the solution should be determined based on the target hardware and involve such metrics as maximum and average memory usage, CPU usage, energy consumption, thermal properties, batch size, sequence length, and latency at the same time as other traffic is present. The model makes use of 43 features

and 43,713 parameters, suggesting a relatively small representation; however, the overall cost for feature collection and preprocessing is not dictated exclusively by the number of parameters. The total time to perform inference of 0.813158 seconds and the average time taken per sequence of 0.081316 seconds give an approximate rate of sequence processing of 12.30 sequences per second under the described experiment conditions.

Table 3: Software-Level inference performance on preprocessed test sequences

Index	Value
Total inference time	0.813158 s
Average time per sequence	0.081316 s
Sequences per second	12,3

Measurement results of software level inferences are shown in (Table 3) and are made on the Google Colab platform for preprocessed sequences. In this case, the computation times are measured excluding the time for capturing packets, features' extraction, missing values handling, scaling features, buffer of sequences, and sending alarms. Thus, the measurement results correspond to the cost of inference only and cannot be considered as real-time operations.

Discussion

The study examined CNN–LSTM hybrid architecture for binary intrusion detection in IIoT traffic with respect to the chronological order of observations. It is logically justified that convolutional layers can learn the relationships among the neighboring features of the input data while LSTM units are able to model dependencies between the sequential observations. Thus, the architecture of CNN–LSTM is aligned with the previous studies that focused on developing CNN–LSTM architectures for the purpose of detecting intrusions in industrial IoT [8]. On the other hand, it becomes clear that the suitability of the architecture does not equal to its effectiveness since the model provided the high precision and low false-positive rate but low recall at the selected threshold. Therefore, the main contribution of the experiment is the empirical characterization of the model that is conservative about generating alarms and needs to be optimized to cover attacks sufficiently. It is clear from these results that while the difficulty is not with the model's ability to avoid false alarms, the problem is rather with the model's poor ability to identify enough attack sequences, which is why it would be important to improve recall in the future.

In particular, the comparison between validation and test results is extremely important. Although 99% validation accuracy might seem good, 60.24% test accuracy and 22.96% recall are a more conservative estimate of the generalization performance. The chronological protocol is an advantage since it helps to avoid the optimism that is caused by random shuffling of the temporally adjacent records. Chronological evaluation, however, may reveal temporal drift, changes in the prevalence of attacks, and distributional differences in traffic conditions between training and test partition. The above factors should not be seen as reasons but rather as possibilities in this research because the issues of time drift and class prevalence changes have not been assessed using distributional analysis. As a result, the creation of the Edge-IIoTset was aimed at providing realistic evaluation of cybersecurity techniques for IoT and IIoT applications, but good performance in one dataset and one time window does not mean general robustness. Edge-IIoTset has been proposed as an aid for testing cybersecurity mechanisms in the context of IoT/IIoT [9]. However, conclusions made from one dataset in one time window cannot be considered proof of robustness in industrial networks. Therefore, a better design of the experiment will include repeated chronological splits, an independent dataset such as TON_IoT [10], or Bot-IoT [11], and comparison with classical machine-learning baseline and simple deep-learning models.

The low recall needs to be investigated primarily as an evaluation and data-distribution problem, rather than being automatically attributed to the CNN–LSTM architecture. Imbalanced classes and heterogeneity of attacks occur frequently in IIoT intrusion datasets [12]. Under such circumstances, the majority class may dominate the objective function while the minority attack patterns are not properly presented during training [13]. According to the established research in the area of imbalanced learning, it is recommended to use three approaches to the problem together rather than focusing only on accuracy. In the context of the current study, these experiments should include class-weighted binary cross-entropy, focal loss, careful oversampling of the training set, and hard example mining. Resampling must be done only in the training partition before sequence construction when needed, because otherwise, the estimation of the deployment performance will be distorted. The effects of each of the interventions need to be evaluated through recall, precision, F1-score, PR-AUC, and false-positive rate with confidence intervals over the repetitions.

ROCAUC and Precision-Recall AUC give discriminatory power across all decision thresholds that were assessed, but it is important to note that they cannot replace the confusion matrix at a selected operating point. ROC-AUC estimates the ordering of positive and negative examples at all thresholds, while PR-AUC is sensitive to the prevalence of the positive class and gives a more targeted assessment of attack detection in case of imbalanced classes [14]. Therefore, it can be concluded that the model may rank attacks higher than normal traffic despite the fact that the selected threshold provides a high precision and low recall [15]. This interpretation suggests carrying out a study of the threshold calibration on the validation set. The threshold must be frozen before test evaluation and the criterion of its selection has to be based on the expected industrial costs of the false positives and false negatives.

From a practical perspective, the current model may be used as the component of the detection pipeline. This low false positive rate means that very few non-attack sequences have been wrongly detected as attacks. To determine if this will lead to a reduction of work for analysts or can be used in a detection pipeline, this needs to be tested in a real-world environment. However, the high false-negative rate does not allow using the model as the only technique of blocking or clearing IIoT traffic. The detection pipeline can include the CNN-LSTM detector, rules for protocol awareness, statistical change detection, second-stage anomaly detector, and feedback from analysts. Thus, it will become possible to operate different thresholds for different protocols or assets, because the risk from unusual Modbus/TCP event is different from the risk from unusual HTTP event. As a result, the survey literature on IoT security also highlights

the problem of the inability of the single learning model to solve challenges related to heterogeneous devices, evolving attacks, label sparsity, and deployment limitations [16].

Finally, the inference measures look promising, but are preliminary. The current model has 43,713 parameters and shows high software-level throughput in the experimental environment, which encourages further edge evaluation. However, an edge claim should be supported by the full pipeline evaluation on specified hardware including feature extraction time, memory footprint, energy consumption, and latency under the stream of traffic. Moreover, it is important to evaluate how the compression techniques such as quantization or pruning affect the recall, especially, the recall of the minority attack categories. Thus, the next step in the research should be improving attack coverage, validation on independent and temporally different data, and real-world gateway performance measurement [17].

Limitation

The present research has multiple drawbacks. First, the experiments were performed using only one set of data and only one time window, thus not allowing for generalization of results to other industrial systems, other equipment, other protocols, and other attack vectors. Second, the analysis relied on binary classification and did not reveal if the model is capable of distinguishing between attack families and novel attacks. Third, the balance of normal and malicious data in every split, the length of the sequence, the stride, and the decision threshold need to be disclosed explicitly for replicability purposes. Fourth, the inference experiment was done in a software testing environment and not on an actual edge device; therefore, the current throughput cannot be considered as end-to-end industrial latency. Finally, the test recall rate is low, which means that the model should not be used for deployment at this point as a standalone protective measure.

Conclusion

The paper evaluates the CNN-LSTM design for the purposes of binary intrusion detection on IIoT traffic with chronological order. It shows the ability of the design to provide high precision and low false positive rate within the presented test scenario; however, the recall is limited, meaning the attack coverage is inadequate at the selected decision threshold. Inference tests at the software level suggest opportunities for edge device evaluation but fail to prove the real-time performance of the network on the industrial edge device. Therefore, the model in question can be regarded as experimental baseline requiring more validation with respect to class distribution, threshold adjustment, baselines, temporal analysis, and physical edge device evaluations.

Recommendations

Any future studies on this topic need to start by providing the class distribution along with the counts in the confusion matrix, followed by an analysis of the impact of class-weighted training and threshold tuning on validation data alone. Additionally, comparisons need to be drawn between the hybrid model and models that use CNN alone, LSTM alone, and classical machine learning techniques.

Future Work

The study can be further developed by integrating Attention mechanisms with CNN-LSTM, or by using modern architectures such as Transformer, in addition to exploring federated learning techniques in distributed environments. The model should also be tested on real Edge devices to assess response time and resource and power consumption, and to verify its suitability for real-world industrial applications. Future research must explore class weighted learning, focal loss, threshold tuning, multi-class attacks detection, repeated temporal splitting, and testing on a separate IIoT data set.

References

1. Ferrag MA, Friha O, Hamouda D, Maglaras L, Janicke H. Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning. IEEE Access. 2022;10:40281-40306. https://doi.org/10.1109/ACCESS.2022.3165809.
2. Al-Garadi MA, Mohamed A, Al-Ali AK, Du X, Ali I, Guizani M. A survey of machine and deep learning methods for Internet of Things (IoT) security. IEEE Commun Surv Tutor. 2020;22(3):1646-1685. https://doi.org/10.1109/COMST.2020.2988293.
3. Ferrag MA, Maglaras L, Moschoyiannis S, Janicke H. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. J Inf Secur Appl. 2020;50:102419. https://doi.org/10.1016/j.jisa.2019.102419.
4. Alsoufi MA, Razak S, Siraj MM, Nafea I, Ghaleb FA, Saeed F, Nasser M. Anomaly-based intrusion detection systems in IoT using deep learning: A systematic literature review. Appl Sci. 2021;11(18):8383. https://doi.org/10.3390/app11188383.
5. Altunay HC, Albayrak Z. A hybrid CNN+LSTM-based intrusion detection system for industrial IoT networks. Eng Sci Technol Int J. 2023;38:101322. https://doi.org/10.1016/j.jestch.2022.101322.
6. Khacha A, Aliouat Z, Harbi Y, Gherbi C, Saadouni R, Harous S. Landscape of learning techniques for intrusion detection system in IoT: A systematic literature review. Comput Electr Eng. 2024;120:109725. https://doi.org/10.1016/j.compeleceng.2024.109725.
7. Al-Haija QA, Droos A. A comprehensive survey on deep learning-based intrusion detection systems in Internet of Things (IoT). Expert Syst. 2025;42(2):e13726. https://doi.org/10.1111/exsy.13726.
8. de Elias EM, Carriel VS, de Oliveira GW, *et al*. A hybrid CNN-LSTM model for IIoT edge privacy-aware intrusion detection. In: 2022 IEEE Latin-American Conference on Communications (LATINCOM); 2022. https://doi.org/10.1109/LATINCOM56090.2022.10000468.
9. Alsaedi A, Moustafa N, Mohammadio A, *et al*. TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems. IEEE Access. 2020;8:165130-165150. https://doi.org/10.1109/ACCESS.2020.3022862.
10. Koroniotis N, Moustafa N, Sitnikova E, Turnbull B. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. Future Gener Comput Syst.

2019;100:779-796. https://doi.org/10.1016/j.future.2019.05.041.
11. Krawczyk B. Learning from imbalanced data: Open challenges and future directions. Prog Artif Intell. 2016;5:221-232. https://doi.org/10.1007/s13748-016-0094-0.
12. He H, Garcia EA. Learning from imbalanced data. IEEE Trans Knowl Data Eng. 2009;21(9):1263-1284. https://doi.org/10.1109/TKDE.2008.239.
13. Saito T, Rehmsmeier M. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PLoS One. 2015;10(3):e0118432. https://doi.org/10.1371/journal.pone.0118432.
14. Davis J, Goadrich M. The relationship between precision-recall and ROC curves. In: Proceedings of the 23rd International Conference on Machine Learning; 2006. p. 233-240. https://doi.org/10.1145/1143844.1143874.
15. Kikissagbe BR, Adda M. Machine learning-based intrusion detection methods in IoT systems: A comprehensive review. Electronics. 2024;13(18):3601. https://doi.org/10.3390/electronics13183601.
16. Rahman MM, Al Shakil S, Mustakim MMR. A survey on intrusion detection system in IoT networks. Cyber Secur Appl. 2025;3:100082. https://doi.org/10.1016/j.csa.2024.100082.
17. Cheng Y, Wang D, Zhou P, Zhang T. A survey of model compression and acceleration for deep neural networks. IEEE Signal Process Mag. 2018;35(1):126-136. https://doi.org/10.1109/MSP.2017.2765695.