Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.054203

ARTICLE

A Lightweight Intrusion Detection System Using Convolutional Neural Network and Long Short-Term Memory in Fog Computing

Hawazen Alzahrani[1], Tarek Sheltami[1], Abdulaziz Barnawi[2], Muhammad Imam[2,*] and Ansar Yaser[3]

[1]Computer Engineering Department, Interdisciplinary Research Center of Smart Mobility and Logistics, King Fahd University of Petroleum and Minerals, Dhahran, 31261, Saudi Arabia

[2]Computer Engineering Department, Interdisciplinary Research Center for Intelligent Secure Systems, King Fahd University of Petroleum and Minerals, Dhahran, 31261, Saudi Arabia

[3]Transportation Research Institute (IMOB), Hasselt University, Hasselt, 3500, Belgium

*Corresponding Author: Muhammad Imam. Email: mimam@kfupm.edu.sa

Received: 21 May 2024 Accepted: 02 August 2024 Published: 12 September 2024

ABSTRACT

The Internet of Things (IoT) links various devices to digital services and significantly improves the quality of our lives. However, as IoT connectivity is growing rapidly, so do the risks of network vulnerabilities and threats. Many interesting Intrusion Detection Systems (IDSs) are presented based on machine learning (ML) techniques to overcome this problem. Given the resource limitations of fog computing environments, a lightweight IDS is essential. This paper introduces a hybrid deep learning (DL) method that combines convolutional neural networks (CNN) and long short-term memory (LSTM) to build an energy-aware, anomaly-based IDS. We test this system on a recent dataset, focusing on reducing overhead while maintaining high accuracy and a low false alarm rate. We compare CICIoT2023, KDD-99 and NSL-KDD datasets to evaluate the performance of the proposed IDS model based on key metrics, including latency, energy consumption, false alarm rate and detection rate metrics. Our findings show an accuracy rate over 92% and a false alarm rate below 0.38%. These results demonstrate that our system provides strong security without excessive resource use. The practicality of deploying IDS with limited resources is demonstrated by the successful implementation of IDS functionality on a Raspberry Pi acting as a Fog node. The proposed lightweight model, with a maximum power consumption of 6.12 W, demonstrates its potential to operate effectively on energy-limited devices such as low-power fog nodes or edge devices. We prioritize energy efficiency while maintaining high accuracy, distinguishing our scheme from existing approaches. Extensive experiments demonstrate a significant reduction in false positives, ensuring accurate identification of genuine security threats while minimizing unnecessary alerts.

KEYWORDS

Intrusion detection; fog computing; CNN; LSTM; energy consumption

1 Introduction

The Internet of Things (IoT) is expanding daily as new devices are connected to the network. It provides ubiquitous connectivity for devices, services, and systems and is widely used in our daily

routines. We experienced an increase in linked gadgets and IoT technologies, ranging from smart homes, drones, and even autonomous vehicles. While IoT continues to grow in popularity, security threats are also increasing [1]. Due to their small size and low power, these devices cannot be equipped with advanced defense mechanisms. It is critical to ensure network security and information protection in IoT applications as it could lead to increased network vulnerabilities and threats. It is important to transport the created data to a device with more computational capability to carry out the storage, processing, and analysis. However, the move to cloud computing is impeded by these devices' high traffic volume. Fog computing is a distributed computing paradigm that works nearer to the network's edge, allowing it to pre-process data before sending it to the cloud. It can thus provide faster handling and response for IoT devices [2].

By bringing computation and storage closer to the data source, fog computing reduces latency and bandwidth usage, making it an ideal solution for real-time applications in IoT environments. Despite its advantages, fog computing still faces challenges, particularly in terms of security and resource constraints. Fog devices are deployed in areas that lack adequate protection, making them susceptible to various cyberattacks. To address these challenges, IDSs have been implemented in fog computing to significantly mitigate the security risks posed by attacks on the fog infrastructure. IDSs are effective methods for detecting the presence of attacks or abnormal behaviors in the fog layer by monitoring the software and hardware within a network. They assist enterprises in detecting and mitigating a wide range of attacks, including intrusions, unauthorized access attempts, and malicious software infections [3]. IDS are deployed in two different methods: at the host level (HIDS) on a node to monitor system operations on its system application files or the OS that is running on the node. Second, at the network level (NIDS) on a border router or gateway, where it analyzes traffic flows on the network [4]. Anomaly-based NIDS is very effective at identifying previously unknown or novel assaults and zero-day attacks. This proactive strategy allows for rapid detection and response to security problems, limiting possible harm or illegal access. The signature-based IDS detects known assaults successfully due to its working technique. However, it does not detect unknown attacks and is insufficient for the limited resources of the IoT. Anomaly-based IDS is thought to be appropriate for IoT, it can identify unknown attacks. However, anomaly-based IDS may cause false alerts by wrongly categorizing regular traffic as abnormal traffic [5]. The defense cannot anticipate that there will be a clear separation between an attack and the regular usage of resources by a legitimate user, which presents another difficult situation. Instead, there will likely be some overlap, which could result in an undesirable circumstance. ML algorithms have been implemented to assist in these situations. ML in NIDS has emerged as a promising approach to the problem of network security in fog computing. Through intelligent algorithms that can learn from different datasets, it can help identify patterns and anomalies in data that may indicate potential security threats. ML-based IDS on fog computing can also leverage the distributed nature of the computing paradigm to implement more efficient and effective intrusion detection mechanisms.

Many interesting IDSs have been presented based on ML and DL approaches, such as support vector machine (SVM) [6], auto-encoder (AE) [7], and artificial neural networks (ANN) [8]. These methods have demonstrated encouraging results in identifying significant features within IoT traffic and efficiently detecting network intrusions and attacks. However, implementing IDSs based on IoT technology on fog nodes with limited capacity and low power can be complex and demanding. Employing a lightweight IDS can potentially offer a beneficial solution for IoT devices. These lightweight IDSs utilize machine learning algorithms to effectively detect intrusion attempts without consuming the energy of the fog node [9].

This research aims to build a lightweight IDS based on hybrid ML models for IoT systems. This improves security and reduces overhead at fog nodes while providing services to the IoT layer. This approach involves leveraging the capabilities of the fog layer by moving the first line of defense closer to the IoT layer and reducing networking and computation costs from the cloud layer. To the best of our knowledge, most existing FC-IDSs that use the hybrid approach of CNN and LSTM have achieved high accuracy [10,11]. However, none of these studies considered energy consumption. The use of old datasets and deploying the model to the cloud layer are the limitations of the previous work in lightweight IDSs [12]. The CICIoT2023 dataset, which includes multiple attacks not available in other IoT datasets, was used to IoT experts develop new security analytics solutions [13]. Fig. 1 illustrates the proposed lightweight IDS structure for the fog computing environment. The fog computing environment is comprised of three major components: end devices, fog nodes, and cloud infrastructure. Fog computing connects the IoT and cloud layers by providing processing, storage, and network services.

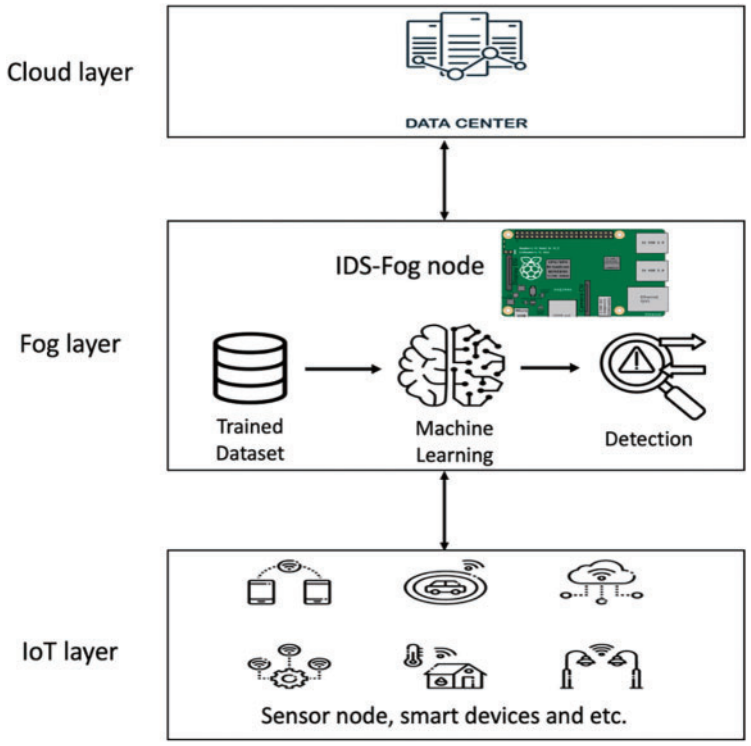

Figure 1: The structure of the proposed system

The objective of this research is to address the challenges of building energy-aware IDS in the fog layer, with a focus on reducing and mitigating overhead. The problem at hand encompasses several key aspects: existing IDS models often rely on outdated datasets that fail to capture the vast majority of modern attacks. To overcome this limitation, this study proposes the use of a modern dataset, CICIoT2023, which includes a comprehensive range of contemporary attack scenarios. By leveraging this dataset, the proposed IDS model aims to enhance detection accuracy and ensure its relevance in the context of emerging threats. To achieve effective attack classification, a hybrid CNN and LSTM machine learning model will be employed. By optimizing the energy efficiency of the IDS model, the research seeks to minimize resource consumption and improve the sustainability

of fog computing environments. Furthermore, the study will explore the practicality of deploying IDSs using limited resources, specifically utilizing the Raspberry Pi as a Fog node. This investigation aims to demonstrate the feasibility of implementing IDS functionality in resource-constrained fog computing environments, thereby expanding the applicability of IDS solutions. The performance evaluation of the proposed IDS model will be conducted based on key metrics, including latency, energy consumption, and detection rate. By assessing these metrics, the research aims to quantify the efficiency and effectiveness of the model in comparison to existing techniques, with a particular emphasis on improving accuracy and reducing the false positive rate.

The key contributions of this work are:

1. Building an energy-aware, IDS in the Fog Computing layer, which reduces and mitigates the overhead.
2. Using a modern dataset, CICIoT2023, instead of the outdated datasets that do not include most modern attacks.
3. Using a hybrid CNN and LSTM machine learning model for binary-class and multi-class attack classification, considering the energy consumption of the model.
4. Exploring IDS using limited resources, specifically the Raspberry Pi 3 as a Fog node.
5. Evaluating the performance of the model based on latency, energy consumption, and detection rate.
6. Improving the accuracy compared to the existing techniques and reducing the false positive rate.

The rest of the paper is organized as follows: the next section is a comprehensive literature review of IDS in fog computing. Section 3 discusses the motivation behind investigating research in the fog layer. Section 4 provides an overview of the methodology of the proposed work. Section 5 presents the evaluation of the results and comparison with previous studies. Finally, the conclusion and future work are presented in Section 6.

2 Literature Review

These studies aim to develop novel techniques, algorithms, or architectures that optimize IDS performance while minimizing energy and CPU consumption. We explore various approaches, such as energy-aware algorithms, and lightweight ML models, that contribute to more sustainable and energy-efficient IDS solutions in fog computing.

Khater et al. [12] proposed a lightweight IDS for a fog computing environment using a Multilayer Perceptron (MLP) model with a Raspberry Pi serving as a fog node. It was used to detect different types of attacks on two datasets: ADFA-LD, which had 94% accuracy, 95% recall, and 92% F1 score, and ADFA-WD, which had 74% accuracy, 74% recall, and 74% F1 score. Their studies showed an average testing time of about 750 microseconds, which is reasonable for many IoT applications. However, their model still needs improvement in detection accuracy and computing efficiency, as it cannot detect present IoT network assaults effectively. Authors in [14] used the ADFA-LD dataset to classify data using an MLP model and a modified Vector Space Representation (MVSR) approach known as N-gram. They also used the Linear Correlation Coefficient (LCC) to address test data containing missing N-grams. Principal Component Analysis (PCA) and Mutual Information (MI) were used for feature selection. The simulation results showed a 5% False Positive Rate (FPR) and 96% accuracy. It achieved low energy consumption of 8.809 mj and low CPU time utilization of 4.404 ms. However, to identify the most recent IoT network threats, their model needs an enhanced dataset. Aliyu et al. [15] suggested IDS

nodes that identify deviations from typical network activity, classifying them as malicious and isolating the suspected node. They incorporated Exponentially Weighted Moving Average (EWMA) into the system to reduce noise from social media discussions. After adding EWMA, accuracy increased from 80% to 95%, and it could identify intrusions 0.25–0.5 s faster. However, using EWMA results in a minimum 0.75–1.3 s increase in latency. The suggested system's application to other attack types or domains was not examined, focusing exclusively on man-in-the-middle (MITM) assaults in fog computing for social media. Aliyu et al. [16] introduced a unique intrusion detection technique modeled after the human immune system in fog computing using Artificial Neural Networks (ANN). Reduced resource usage and increased efficiency can be attained by utilizing a distributed design that makes use of fog nodes and cloud computing. The suggested solution reduced energy usage by 10% compared to using a neural network on the fog node, achieving accuracy up to 98.8% in KDD-99 and 96.7% in NSL-KDD datasets. As the duty cycle increases, the system's accuracy is seen to increase. However, this also results in increased latency and energy consumption. Roy et al. [17] devised B-Stacking, a unique IoT intrusion detection approach. Using the CICIDS2017 and NSL-KDD datasets, it applies improved machine learning algorithms to obtain high accuracies of about 99% and 98%, respectively. The model includes optimization techniques like sampling, data scaling, dimensionality reduction, and multicollinearity removal. These processes reduce computing complexity by effectively identifying critical aspects and drastically reducing the number of features and data points needed for abnormality identification. According to the experimental results, B-Stacking is a potential intrusion detection technique for IoT environments due to its high detection rate and very low false alarm rate. However, the research does not evaluate power consumption through simulation or implementation. Wardana et al. [18] implemented federated learning (FL) in an edge-fog-cloud architecture to address latency concerns by distributing resources closer to the data source or end users, leading to shorter round-trip times and better application responsiveness. The proposed technique was validated using the CICIoT2023 dataset. FL adds communication and synchronization overhead and had higher average memory usage than centralized learning.

The studies primarily focus on IDS functionality without explicitly considering energy constraints. Samy et al. [19] compared six distinct DL models to determine the most effective one. These models were evaluated using five distinct datasets, each with a unique attack type. The experiments demonstrated that the LSTM model outperforms the other DL models in terms of accuracy and detection rate. De Souza et al. [20] proposed a binary Extra Tree (ET) classifier for traffic analysis, while Rai [21] utilized ensemble techniques such as Distributed Random Forest (DRF) and Gradient Boosting Machine (GBM). Sadaf et al. [22] introduced the Auto-IF method using Autoencoder (AE) and Isolation Forest (IF). Abdussami [23] developed an Incremental Deep Neural Network (DNN) for intrusion detection. Souza et al. [24] combined Deep Neural Networks (DNN) and k-Nearest Neighbor (kNN) in a hybrid binary classification technique called DNN-KNN. Sun et al. [25] emphasized the achievements of DL-IDS, including the usage of CNN-LSTM hybrid networks and category weight optimization. Attique et al. [26] proposed the Cu-DNNGRU system based on CUDA-deep neural network gated recurrent units. Tuli et al. [27] presented a lightweight automatic object identification system using deep learning and the Aneka framework. Çavuşoğlu et al. [28] utilized deep learning and transfer learning for network traffic classification. Chang et al. [29] introduced fast Fourier transform (FFT) and DL approaches for extracting high-level information. Jiang et al. [30] proposed a model with algorithms to maximize accuracy and minimize latency. Gudla et al. [31] compared DL models with traditional ML models for DDoS attack detection. Singh et al. [32] combined Naive Bayesian-based IDS with a Markov Model and Virtual Honey Pot Device. Roopak et al. [33] explored deep learning models such as CNN and LSTM for DDoS attack

detection. Various other methods and frameworks were also introduced by different authors to address DDoS attacks in fog and cloud computing settings, IoT networks, and VPN servers. Diro et al. [34] found that distributed threat detection at the fog level is more scalable than centralized cloud-based methods. The DAE-BILSTM model, as proposed by Selim et al. [35], combines a Bidirectional Long Short-Term Memory (BiLSTM) with a Deep AutoEncoder (DAE) to provide effective feature extraction, noise reduction, and network traffic analysis. Alzahrani et al. [36] extracted features from the data using a correlation-based method to eliminate unnecessary information, producing a model with no computational overhead. Onah et al. [37] combined a Genetic Algorithm Wrapper-Based feature selection approach with a Naive Bayes Classifier. Kumar et al. [38] suggested using interplanetary file systems (IPFS) for off-chain storage in blockchain networks and AI for real-time analysis in blockchain-IoT systems. Almaiah et al. [39] examined and analyzed the behavior and propagation of Shamoon assaults in the fog computing edges. Xu et al. [40] underlined the requirement for a fully functional infrastructure backed by cloud computing and fog computing, in addition to a data-driven core engine and semi-supervised techniques. Rahman et al. [41] proposed techniques to overcome resource limitations in IoT devices while achieving detection accuracy comparable to centralized IDS. Tu et al. [42] used physical layer security (PLS) to identify and prevent impersonation attacks in fog computing networks. The authors [43] proposed a method involving two stages of classification for improved accuracy, calling for various base learners and ensemble approaches. Moustafa et al. [44] presented the outlier Dirichlet mixture-based ADS (ODM-ADS) technique, allowing for self-adaptation to data poisoning attempts, creating a valid profile, and using an outlier function to identify anomalies. Gazdar [45] offered FDeep, an IDS deployed in the Fog Layer of the IoT system in smart home environments. To prevent degradation of detection skills, the DL model in FDeep is re-trained and updated periodically. The results showed that the LSTM model using CNN surpassed other models in terms of detection accuracy. The authors [46] tested several DL models, including DNMLP, LSTM, BiLSTM, GRU, CNN + LSTM, and HEM. The results revealed that the LSTMDL model surpassed DNMLP in properly predicting attacks, although it took longer to detect activity (CBDT) than other models. Authors in [47] developed a hybrid deep learning system to detect DDoS attacks by combining CNN and LSTM models. Utilizing the CICIoT2023 dataset, their system achieved a high attack detection rate of 99.995% and an attack type detection rate of 99.96%. However, the analysis revealed certain drawbacks and areas for development. One issue is that achieving high accuracy requires a vast amount of data. The use of large datasets increases training and testing timeframes, which can be inefficient in real-time detection settings where immediate response is required. Gad et al. [48] examined different machine learning techniques in the context of ToN-IoT dataset partitions, assessing their performance in binary and multi-class classification tasks. The authors emphasized how much better the XGBoost method is in the suggested model compared to other ML algorithms. According to Labiod et al. [49], an intrusion detection system that uses a two-layered fog architecture operates by placing an attack identification module in the cloud and an anomaly detector inside fog nodes. This approach restricts connection between fog nodes and the cloud, allowing real-time data processing while simultaneously lowering latency. de Araujo-Filho et al. [50] presented a novel method for intrusion detection using generative adversarial networks (GANs) and a fog-based architecture. The proposed method effectively addresses latency and lack of labeled data, as demonstrated by the experimental findings. Meng et al. [51] offered a plan of defense for mobile fog computing (MFC) networks against clever attackers, integrating the DQL algorithm, prospect theory, and static subjective game modeling. Several studies [52,53] presented approaches to DDoS defense for fog and cloud computing settings.

3 Motivations

In recent years, the proliferation of interconnected devices and the growing adoption of fog computing have introduced new security challenges in the Fog layer. IDS play a crucial role in detecting and mitigating potential security threats in these environments. However, building energy-aware, anomaly-based IDS in the Fog layer poses significant challenges that need to be addressed.

One of the primary limitations of existing IDS models is their reliance on outdated datasets that fail to capture most modern attacks. This research is motivated by the need to overcome this limitation by leveraging a modern dataset, CICIoT2023, which includes a comprehensive range of contemporary attack scenarios. By incorporating this dataset, the proposed IDS model aims to enhance detection accuracy and ensure its relevance in the face of emerging threats. Effective attack classification is another critical aspect that needs to be addressed. To achieve this, a hybrid CNN and LSTM machine learning model will be employed. This hybrid architecture is capable of handling both binary-class and multi-class attack classification tasks while considering the energy consumption of the model. By optimizing the energy efficiency of the IDS model, this research aims to minimize resource consumption and improve the sustainability of fog computing environments. To the best of our knowledge, most existing IDSs that use CNN and LSTM have achieved high accuracy, however none of these studies considered the energy consumption.

Furthermore, the practicality of deploying IDSs using limited resources is an important consideration. Utilizing Raspberry Pi as a Fog node, this study explores the feasibility of implementing IDS functionality in resource-constrained fog computing environments. By demonstrating the viability of IDS solutions in such environments, the applicability and effectiveness of IDS can be expanded. The proposed IDS model will be evaluated based on key performance metrics, including latency, energy consumption, and detection rate. Through this evaluation, the efficiency and effectiveness of the model will be quantified and compared to existing techniques. Special emphasis will be placed on improving accuracy and reducing the false positive rate, addressing the pressing need for reliable and efficient IDS solutions in the Fog layer.

Overall, this research is motivated by the need to address the challenges of energy-aware, anomaly-based IDS in the Fog layer, with a focus on reducing and mitigating overhead. By leveraging a modern dataset, employing a hybrid CNN-LSTM model, and considering resource constraints, this study aims to contribute to the development of more efficient and effective IDS solutions for fog computing environments.

4 Methodology and Analysis

This study evaluates an energy-efficient Internet of Things IDS model using various datasets, including CICIoT2023, a modern dataset featuring contemporary attack scenarios. The model employs a hybrid CNN and LSTM machine learning model. The study explores the practicality of deploying IDSs using limited-resources devices like Raspberry Pi as a Fog node, a method consistent with previous research. The study aims to identify areas for improvement and assess the model's generalization capabilities. Papers [12,14] shared a similar methodology with this study. The authors tested the model performance using the Raspberry Pi. Fig. 2 illustrates the proposed IDS methodology.

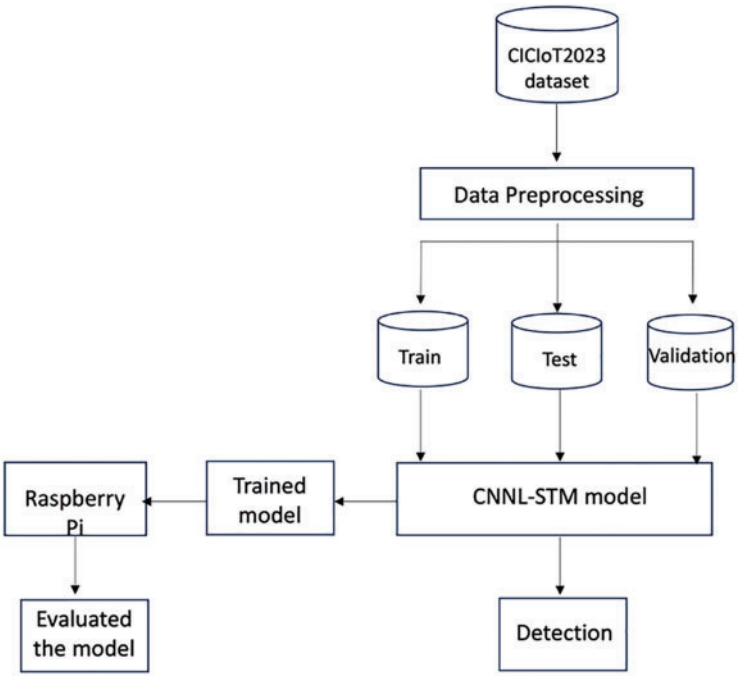

Figure 2: The proposed methodology

4.1 Dataset Processing

The CICIoT2023 dataset is a comprehensive collection of IoT attack data that provides valuable support for developing security analytics applications in real-world IoT operations. It offers opportunities to optimize ML models, explore the impact of features on models, and gain insights into classification interpretations. The dataset was categorized into binary and multi-class categories for classification tasks. Binary classification divides instances into normal and abnormal classes, while multi-class classification identifies different types of attacks, such as DDoS, DoS, Mirai, Benign Traffic, Spoofing, and Recon.

The preprocessing phase includes an attack labeling step to ensure proper labeling. This labeled dataset serves as the foundation for the CNN-LSTM model's supervised training, allowing the model to learn patterns and relationships between network traffic data and attack types. Class imbalance can occur if there is a significant difference in the number of instances across attack labels or classes. In binary classification, the model performs well on both classes, while in multi-class classification, the model learns to differentiate between different types of attacks and normal traffic.

Normalization is a crucial process in data preprocessing, translating numerical data to a common scale to prevent bias in machine learning applications. It aims to maintain similar scales for features, preventing dominance or bias. Standardization transforms data to have a mean of 0 and a standard deviation of 1. The Label Encoder function converts categorical labels into numerical representations, ensuring each unique label is consistently mapped to a unique numerical value. This helps in efficient computation during the training and prediction phases of machine learning models. The encoding method depends on the features of the categorical variables and the dataset. One-hot encoding can increase dataset dimensionality, while attack labeling allocates labels to relevant classes, preserving category distinctiveness and eliminating unintended ordinality.

One-hot encoding is a binary representation strategy used in binary classification, where two classes are present. It creates a binary feature for each distinct label, indicating its presence or absence. This is suitable for binary classification problems as it allows machine learning models to interpret labels in a binary format. In multi-class classification, label binarization is used to encode category labels, converting the binary matrix representation into a binary column. This reduces imbalanced class distributions and allows the model to learn independently for each class label, potentially improving minority class representation and prediction.

The dataset is split into a training set and a testing set using an 80% to 20% ratio. The training set is used to train the CNN-LSTM model, while the testing set is used for the final evaluation. The testing set is further divided into a final testing set and a validation set using a 50%:50% ratio. This secondary split allows for fine-tuning and hyperparameter adjustment, providing an unbiased estimate of the model's performance when applied to unseen data.

4.2 Anomaly Detection Using Machine Learning Models

In the fields of ML and DL, the most advanced and high-performing models typically require a significant number of resources. For this effort, it is essential to conduct research and apply light-ML models in order to effectively implement them on low-power devices. The suggested IDS model uses CNN and LSTM as hybrid machine learning methods to identify normal and abnormal traffic. The architecture of CNN-LSTM shown in Fig. 3.

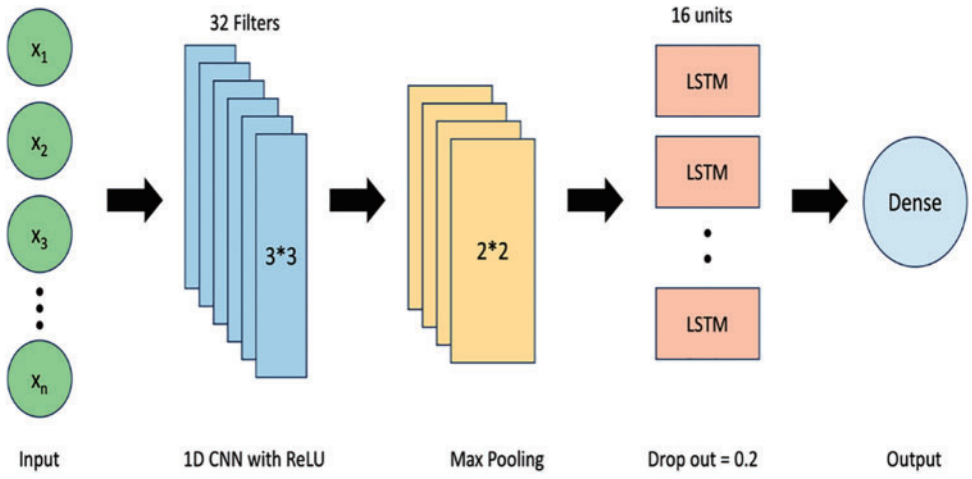

Figure 3: CNN-LSTM architecture

The structure of the model with the function of each layer is as follows:

- Conv1D Layer: This layer performs 1-dimensional convolutions on the input. It has 32 filters, each with a kernel size of 9. The activation function used in this layer is ReLU (Rectified Linear Unit), which introduces non-linearity.

$$ReLU\,(x) = max\,(0, x) \tag{1}$$

where x is the output of the first layer Conv1D, then passed into the second layer which is MaxPooling1D layer to reduce the feature map.

- MaxPooling1D Layer: This layer performs max pooling on the output of the Conv1D layer. It reduces the spatial dimensions by selecting the maximum value within a window of size 2 while retaining the most important features. The purpose of this layer is to down sample the feature maps, reducing computational complexity and controlling overfitting.
- LSTM Layer: This layer utilizes LSTM units for sequential processing. It has 16 units specifies the number of LSTM cells or memory units in the layer, which capture dependencies and patterns in the input sequence. This layer only returns the output at the last time step, rather than the full sequence of outputs. The dropout rate is set to 0.2, which means that 20% of the LSTM units' inputs will be randomly set to zero during training to prevent overfitting.
- Dense Layer: This layer is a fully connected layer with 1 unit indicates that the layer has a single neuron. It serves as the output layer of the model. The activation function used in this layer is sigmoid, which squashes the output value between 0 and 1, representing the probability of the positive class for binary classification. The activation function that used for multi-class classification is SoftMax, which calculates the probabilities of each class.

Additionally, the model is compiled with the following settings:

- Loss Function: A function for measuring the performance of an algorithm: if the predictions are incorrect, the loss function will return a greater value, whereas if the model makes correct predictions, the loss function will return a lower value. Cross-entropy, which evaluates the difference between two probability distributions for a given random variable or series of events, is commonly employed as a loss function when optimizing classification models. Binary-Crossentropy loss function is commonly used for binary classification problems. Categorical-Crossentropy, which is appropriate for multi-class classification problems.

$$Binary - loss = -(\mathrm{y} * \log(\mathrm{p}) + (1 - \mathrm{y}) * \log(1 - \mathrm{p})) \tag{2}$$

where y represents the true label of the sample (either 0 or 1). p represents the predicted probability of the positive class (class 1).

$$Categorical - Crossentropyloss = -\sum (y * log(p)) \tag{3}$$

where $\sum$ represents the summation over all classes, y is a one-hot encoded vector representing the true label of the sample. It has a value of 1 for the true class and 0 for all other classes. The loss is computed by taking the negative sum of the element-wise multiplication of y and $log(p)$.

- Optimizer: Adam. It is an optimization algorithm that adapts the learning rate during training to improve convergence. It adjusts the parameters based on the gradients of the loss function with respect to the parameters computed during backpropagation using the chain rule.

By integrating CNN and LSTM layers, the model achieves a better balance between feature extraction and capturing long-term dependencies compared to using only CNN layers. This approach reduces the need for a very deep network with many convolutional layers, addressing the challenge of modeling long-term dependencies in sequences and potentially alleviating the requirement for an excessively deep architecture. Throughout the fine-tuning and optimization process, model architecture modifications, such as adding or removing layers and adjusting the number of units or filters in each layer, have been made to enhance its performance.

While previous studies have demonstrated high detection performance, understanding the energy consumption implications of CNN-LSTM models is crucial for practical deployment and resource

management. Table 1 presents the final model parameters adopted for better results. Increasing the kernel size and filter count in the convolutional layer allows the model to capture more diverse and complex features. Additionally, reducing the number of LSTM units helps mitigate overfitting while still preserving the model's ability to capture relevant temporal information. Classifying the attack labels into six classes mitigates data imbalance and improves training efficiency.

Table 1: CNN-LSTM parameters

CNN-LSTM	Layers	Kernel/Neurons	Activation/loss	Optimizer	Epoch	Batch size
Binary-class	Conv1D	(9/32)	ReLU	Adam	20	128
	MaxPooling	2	–			
	LSTM	16	–			
	Dense	1	Sigmoid/binary _crossentropy			
Multi-class	Conv1D	(9/32)	ReLU	Adam	20	128
	MaxPooling	2	–			
	LSTM	16	–			
	Dense	8	Sofmax/categorical _crossentropy			

To identify the combination of epoch and batch size that provides the best balance between training time and model performance, and to consider the stability and overall performance on the validation set, the optimized version involves iterating through the entire dataset 20 times (epochs) while using a batch size of 128. The goal is to adapt the architecture to better capture the underlying patterns in the data and improve the trade-off between energy efficiency and detection rate.

4.3 Evaluation Metrics

When evaluating the CNN-LSTM model, several metrics can be used to assess its performance. Here are some commonly used evaluation metrics along with their equations:

- True Positive (TP) indicates the number of attack traffic detected as attack traffic.
- True Negative (TN) indicates the number of normal traffic detected as normal traffic.
- False Positive (FP) indicates the number of attack traffic detected as normal traffic.
- False Negative (FN) indicates the number of normal traffic detected as attack traffic.

Accuracy (ACC): Accuracy measures the overall correctness of the model's predictions.

$$Accuracy = (TP + TN) / (TP + TN + FP + FN) \tag{4}$$

Precision: Precision measures the proportion of correctly predicted positive instances out of all instances predicted as positive.

$$Precision = TP / (TP + FP) \tag{5}$$

Recall: measures the proportion of correctly predicted positive instances out of all actual positive instances.

$$Recall = \mathrm{TP} / (\mathrm{TP} + \mathrm{FN}) \tag{6}$$

$F1$ score: The $F1$ score is the harmonic mean of precision and recall, providing a balanced evaluation metric.

$$F1\ score = 2 * (Precision * Recall) / (Precision + Recall) \tag{7}$$

Area Under the Receiver Operating Characteristic Curve (AUC-ROC): AUC-ROC measures the model's ability to distinguish between classes by plotting the true positive rate (TPR) against the false positive rate (FPR). AUC-ROC represents the area under the ROC curve, which ranges from 0 to 1. A value closer to 1 indicates better classification performance. False Positive (FP): False positive is a metric used in binary classification problems, representing the number of negative instances incorrectly classified as positive by the model. It measures the model's propensity to generate false alarms or false positives.

$$False\ Positive = FP / (FP + TN) \tag{8}$$

4.4 Evaluation Environment

The ML models are implemented using Python 3.9, leveraging libraries such as Numpy, Scikit-learn, and Pandas for data manipulation and preprocessing. Matplotlib and Seaborn are used for data visualization and analysis. TensorFlow's Keras provides the classes and functions necessary for defining and training neural network models. The experiment was tested on a Raspberry Pi 3 Model B+, which features a 64-bit quad-core ARM Cortex-A53 CPU and 1 GB LPDDR2 SDRAM main memory. The Raspberry Pi has been deployed as a fog node in various IoT applications, as referenced in [12,33]. The power consumption and latency at different phases of the model simulation were investigated using the Raspberry Pi in conjunction with an Arduino Uno. Table 2 presents the evaluation environment specifications.

$$P = \mathrm{I} * \mathrm{V} \tag{9}$$

Table 2: Evaluation environment specifications

CPU	64-bit quad-core ARM cortex-A53
RAM	1 GB
TensorFlow	2.7
Python	3.9
Arduino	Uno

This equation calculates the power consumption (P) of the Raspberry Pi by multiplying the current (I) passing through the shunt resistor by the voltage across the Raspberry Pi. The current (I) passing through the 1-ohm shunt resistor (R) is the same as the current passing through the Raspberry Pi since they are connected in series. The voltage across the Raspberry Pi (V_RPi) is measured by subtracting the voltage measured with the Arduino Uno from the power supply voltage. Power consumption was sampled every 2000 ms.

The main phases of running the code included importing necessary packages, loading the dataset sample, preprocessing the dataset, and making predictions. The latency for each step was recorded, and the code was executed multiple times to ensure precision and avoid errors. The experimental setup, including the Raspberry Pi, Arduino, and resistor shunt, is illustrated in Fig. 4. This setup allowed for the detailed measurement and analysis of power consumption during the execution of the ML models.

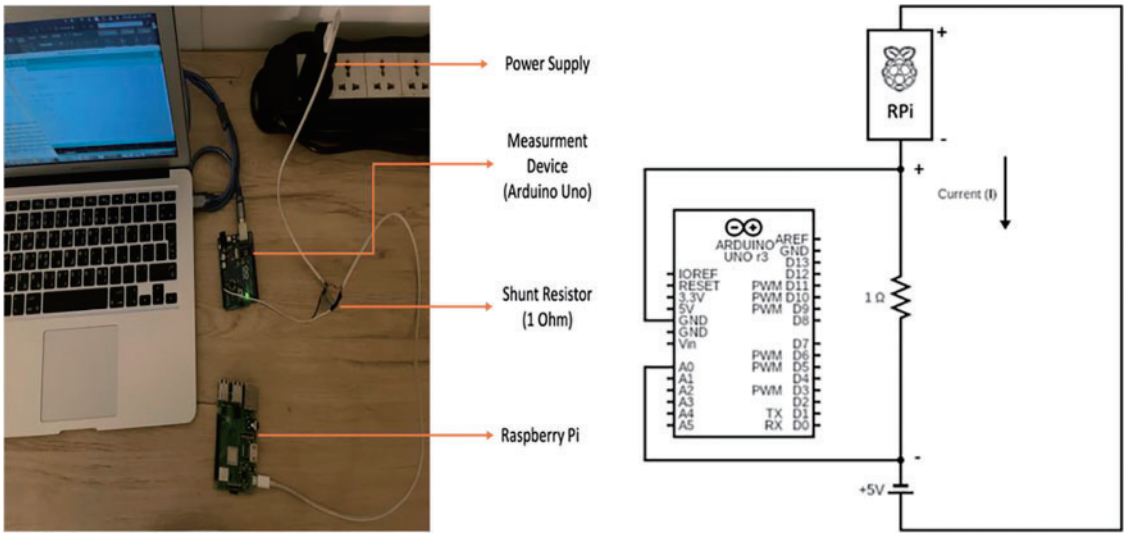

Figure 4: Experiment setup and circuit diagram

4.5 Experiment Setup

The training and validation steps of the ML models were carried out on a PC with a 1.8 GHz Dual-Core Intel Core i5 processor. The classifiers were implemented in Python 3.9, and the trained models were subsequently transferred to a Raspberry Pi 3 Model B+ for further experimentation. The experiment platform for the lightweight IDS utilized the Raspberry Pi 3 Model B+, with the trained ML model and datasets, each with a sample size of 1000, loaded onto the Raspberry Pi.

Energy consumption and latency were investigated to evaluate the efficiency of the model simulation for both binary-class and multi-class classifications during testing. The main phases of running the code were recorded, including importing necessary packages, loading the dataset sample, preprocessing the dataset, and making predictions. Each code execution was manually performed multiple times to ensure precision and avoid errors.

Fig. 4 illustrates the experimental setup, which included a Raspberry Pi, Arduino, and resistor shunt to measure power consumption. The circuit diagram details the experimental configuration for measuring power supplied to the Raspberry Pi. The ground (GND) terminals of the Raspberry Pi, Arduino Uno, and power supply were connected together to ensure a common reference point for the circuit. A 1-ohm shunt resistor was inserted between the positive terminal of the power supply and the positive terminal of the Raspberry Pi. This resistor allows for the measurement of current flowing through the circuit. The A0 (analog input) pin of the Arduino Uno was connected to a point between the shunt resistor and the positive terminal of the Raspberry Pi. This setup enables the Arduino Uno to measure the voltage drop across the shunt resistor, which is proportional to the current flowing through the circuit.

The code was run through an SSH connection with the Raspberry Pi, and the latency for each step was recorded. Eq. (9) was used to calculate the power consumed by the Raspberry Pi.

5 Results and Discussion

In this section, we conduct a detailed performance analysis and explain the findings of the ML models mentioned in the preceding paragraph. The collected results are compared and examined with the goal of documenting and assessing model behavior on recent datasets. Finally, this work is compared to earlier investigations to highlight improvements and differences in performance.

5.1 Model Performance Evaluation

We first evaluate the detection quality of the proposed method for distinguishing between normal and anomalous network traffic in the CICIoT2023 dataset. As shown in Fig. 5, the training and validation accuracy values are reported to be approximately 99.17% and 99.21%, respectively. It is desirable to have a model that performs well on both the training and validation datasets without a substantial difference between the accuracy values. The training accuracy line measures the model's performance on the training dataset, indicating its ability to classify training examples accurately. The validation accuracy line represents the model's performance on unseen data, indicating its ability to detect attacks in real-world fog computing environments. The fact that the validation accuracy is slightly higher than the training accuracy suggests that the model is performing well and generalizing effectively without overfitting to the training data. Fig. 6 shows the training and validation losses, further illustrating the model's performance.

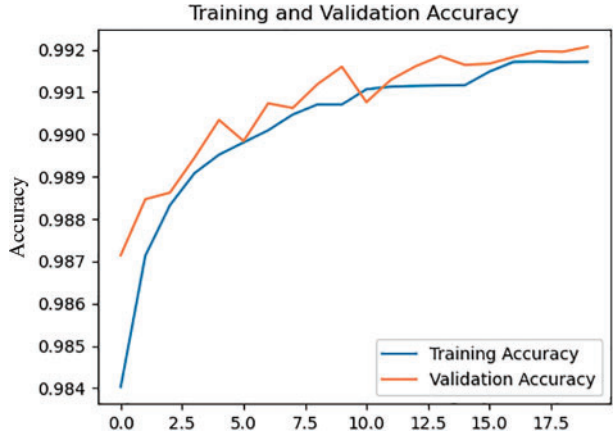

Figure 5: Accuracy of CICIoT2023 binary classification

In ROC analysis, the FPR (False Positive Rate) and TPR (True Positive Rate) are used to plot the ROC curve, which visualizes the performance of a binary classification model across different classification thresholds. In Fig. 7, the ROC curve illustrates the trade-off between the true positive rate and the false positive rate. A score of 99.79% indicates that the model has excellent discrimination ability, achieving a high TPR while keeping the FPR low. This demonstrates that the model is effective in correctly classifying positive instances and avoiding false positives.

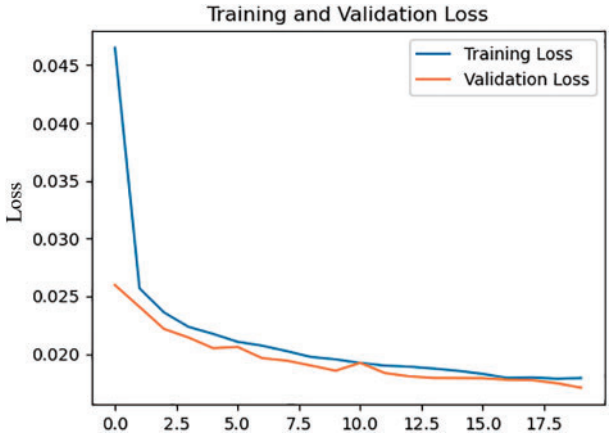

Figure 6: Loss of CICIoT2023 binary classification

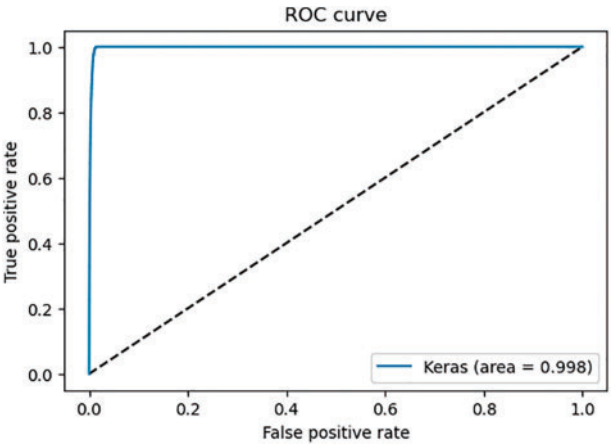

Figure 7: ROC curve of CICIoT2023 binary classification

To evaluate the detection quality of the proposed method for distinguishing between different classes of the CICIoT2023 dataset, we labeled the data for DDoS, DoS, Mirai, Recon, Spoofing, and Benign traffic. Fig. 8 presents the training and validation accuracy values for the multi-class model, which are approximately 92.37% and 92.76%, respectively. The small difference between training and validation results suggests that the model is generalizing well and learning meaningful patterns. The graph indicates that the model is neither overfitting nor underfitting, as the training and validation accuracy lines are relatively close to each other. This demonstrates that the model effectively learns from the training data and adjusts its parameters, enabling it to make more accurate predictions over time. Fig. 9 shows the training and validation losses for the CICIoT2023 dataset, further illustrating the model's performance.

Fig. 10 represents the ROC AUC with a score of 0.95, indicating excellent performance. Random guessing corresponds to a diagonal line from the bottom-left corner to the top-right corner. The model's FPR (False Positive Rate) and TPR (True Positive Rate) show moderate misclassification rates for the attack classes. The detection performance of the CNN-LSTM model for each class of the test dataset is shown in Table 3. It is evident that some classes are well-classified, particularly those with a high number of occurrences in the dataset. For example, Benign Traffic, DoS, and Mirai have

very low misclassification rates, followed by DDoS, Recon, and Spoofing. Due to the similarity in data patterns, the model makes mistakes when distinguishing between DDoS, Recon, and Spoofing. However, the classification is generally successful.

Figure 8: Accuracy of CICIoT2023 multi classification

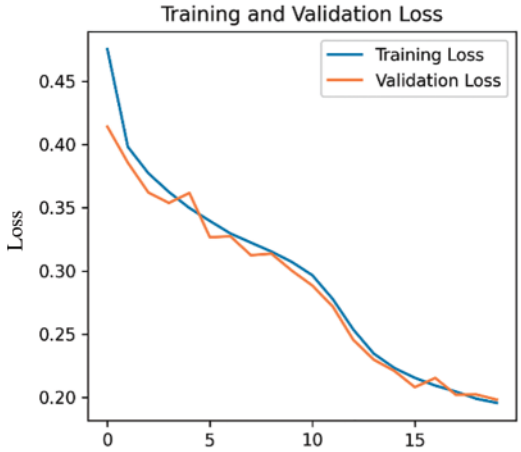

Figure 9: Loss of CICIoT2023 multi classification

Fig. 11 shows the performance of the CNN-LSTM model on the binary and multi-class CICIoT2023, NSL-KDD, and KDD-99 datasets. Evaluating the generalization ability of a model to unseen data involves using a separate test set. The results show high accuracy and consistent performance on the test set, indicating better generalization. Recall measures the proportion of actual positive instances correctly identified by the model. In this model, a recall score of 94% suggests a high ability to correctly identify positive instances. The F1 score provides a balance between precision and recall, considering both false positives (precision) and false negatives (recall). In the binary-class model, an F1 score of 92% suggests a reasonable balance between precision and recall. A higher precision score indicates that the model has a low false positive rate. In our case, a precision score of 90% suggests the model's ability to limit false positive predictions. The evaluation metrics for multi-classification—precision, recall, and F1 score—are 94%, 92%, and 93%, respectively. These metrics demonstrate the performance of the multi-class model on the NSL and KDD-99 datasets. The model

achieved high accuracy rates on both datasets, with a slightly higher accuracy of 99.94% on the KDD-99 dataset. The false alarm rates are relatively low for both the NSL and KDD-99 datasets, indicating a good ability to avoid false positives. The multi-classification of CICIoT2023 achieved a lower false alarm rate compared to other datasets, while KDD-99 had the highest accuracy. In general, the models provide excellent results across multiple datasets and evaluation metrics.

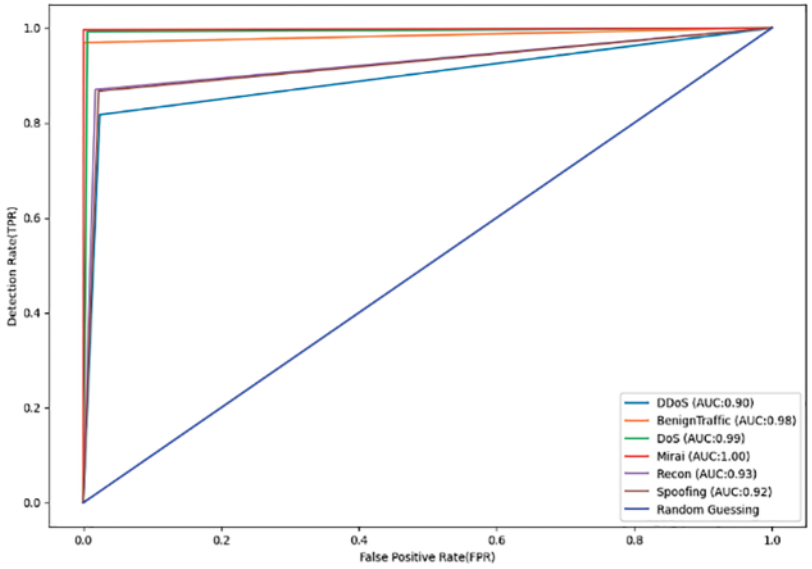

Figure 10: ROC AUC performance

Table 3: Detection performance of the attack classes

Labels	DDoS	Benign traffic	DoS	Mirai	Recon	Spoofing
DDoS	90.1%	0	0	0	7.45%	9.91%
Benign traffic	0.01%	99.01%	2.95%	0.07%	0.13%	0
DoS	0	0.44%	98.99%	0.12%	0.17%	0.01%
Mirai	0	0.03%	0.08%	99.47%	0.19%	0.02%
Recon	6.82%	0	0	0	92.99%	4.55%
Spoofing	8.2%	0	0	0	3.51%	91.29%

The variation in performance across the datasets can be attributed to several factors. NSL is a new version of the KDD-99 dataset, and the difference in performance between the NSL and KDD-99 datasets is because KDD-99 contains many redundant records, which are extremely easy for classifiers to identify, resulting in high performance. In contrast, NSL does not include redundant records in the training set, preventing classifiers from being biased towards more frequent records. The CICIoT2023 dataset encompasses both network traffic and IoT data, making it unique compared to the NSL and KDD-99 datasets. The inclusion of IoT data introduces additional challenges, as IoT devices have distinct communication patterns and may exhibit different behaviors compared to traditional network traffic. The CICIoT2023 dataset, compared to NSL and KDD-99, is larger and includes a

more diverse range of attack types, providing the model with more opportunities to learn and improve its performance in terms of multi-classification.

Figure 11: Performance evaluation of the model for CICIoT2023, NSL, and KDD-99 datasets

The Raspberry Pi is an appropriate platform for assessing the performance of the model due to its limited processing resources. The power consumption of the Raspberry Pi, which was utilized to mimic a fog node, peaks at 3.9 W during boot. In idle mode, consumption drops to 3 W. Fig. 12 displays the energy consumption recorded for each step of employing CNN-LSTM intrusion detection on the Raspberry Pi with various datasets. However, the energy consumption of the cloud and the IoT layer are beyond the scope of this work. For the dataset in testing, the maximum power consumption of 6.12 W was recorded during the prediction step in evaluating the multi-class CICIoT2023 dataset. The KDD-99 dataset recorded the lowest power consumption in every single run compared with other datasets.

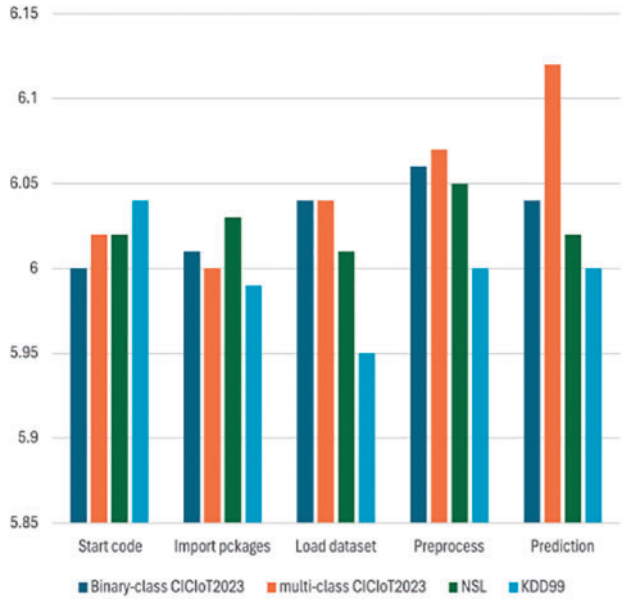

Figure 12: Energy consumption of the model steps

We also investigated the latency of the proposed model. Latency is defined as the period between when an input is supplied to the model and when an output classifying the packet is detected. Fig. 13

displays the model latency for CICIoT2023, KDD-99, and NSL-KDD on the Raspberry Pi. The results were obtained using a 1000-instance sample size, selected uniformly at random. This helps us to acquire statistically accurate latency values. Additionally, comparing the results of the training and validation sets with a separate test sample enhances the reliability of the findings. Larger sample sizes contribute to more precise estimates and increase statistical power. However, it is important to consider the impact on energy consumption and the time required for data processing.

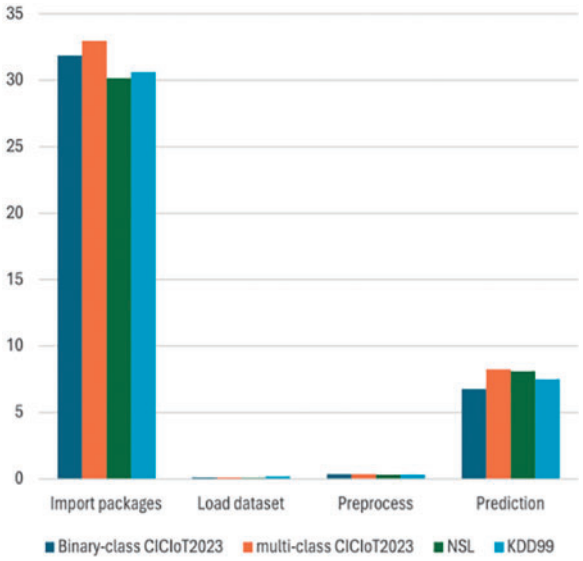

Figure 13: Latency of the model steps

To prevent experimental mistakes, the model code was manually executed 12 times for each dataset. The latency for each step was embedded in the code testing the model. To get the worst-case scenario for CNN-LSTM and different dataset loading and processing, the maximum result was presented. It's worth noting that the presented results of energy and latency for each dataset are from a single run. Each phase of code execution has different latency and time consumption; for example, importing the necessary packages to run the model has a different cost than uploading a sample dataset for inference, as importing packages takes more time. High latency during the importing packages step is due to loading and initializing libraries and packages for ML model execution. Some packages require configuration steps, like loading model weights. However, with each iteration, latency decreases due to caching mechanisms, reducing overhead and improving overall execution speed, while the energy consumption doesn't show significant differences or variations across iterations. This is attributed to various factors. Establishing benchmark datasets and performance metrics that include latency measurements would facilitate better evaluation and comparison of IDS solutions. This study encourages future researchers to consider latency as an essential metric when evaluating IDS performance. The detection and response time can directly impact the system's effectiveness in identifying and mitigating security threats. Minimizing detection and response time is crucial to reducing potential damage caused by intrusions.

In terms of reducing and mitigating overhead of the IDS, during the load dataset phase, the model consumes 6.04 W, while before increasing the kernel size and filter count of the CNN layer and reducing the number of LSTM units and classifying the attack labels into 6 classes, the model consumes 6.05 W. In the preprocess phase, the model consumes 6.07 W of power, whereas the model

consumed 6.13 W before optimization. Finally, in the prediction phase, the model consumes 6.12 W of power, while it consumed 6.25 W before optimization. The model, after optimization, showcases better power consumption efficiency across all phases, indicating that it can perform the required tasks while utilizing fewer resources.

5.2 Comparison with Related Work

First, we review studies that proposed hybrid classification models combining CNN with LSTM [10,11,19,45,46] and studies that tested on the CICIoT2023 dataset [18,47]. However, these studies did not consider energy and CPU consumption. Table 4 presents a comparison in terms of the datasets used for evaluation, number of epochs, accuracy, false alarm rate, and architecture design. The proposed system outperforms previous studies [11,19] in terms of false alarm rate, achieving high accuracy rates on the NSL-KDD and KDD-99 datasets, surpassing most prior studies. While [47] obtained high accuracy rates in both binary and multi-class classifications, our work emphasizes minimizing latency and energy consumption while maintaining a reasonable level of accuracy. Even though [18] achieved slightly higher accuracy for binary classification within an edge-fog-cloud framework, it is noteworthy that our proposed work takes into account the false alarm rate (FAR), which is measured at 0.38 for binary classification. Our scope is to develop a centralized architecture that aims to be lightweight and energy-efficient, distinguishing our scheme from existing approaches that may not prioritize energy efficiency. Despite the goal of being energy-efficient, our scheme has not compromised on accuracy. While the computational process may require more time and energy compared to some previous studies, we have successfully achieved a balance between efficiency and accuracy. Our approach maintains a high level of accuracy while remaining lightweight and suitable for resource-constrained environments. Through our experiments and evaluation, we have achieved a significant reduction in false positives, indicating a more accurate identification of genuine security threats while minimizing unnecessary alerts.

Table 4: Comparison of the proposed system with papers in literature review

IDS	Dataset	Epochs	Accuracy	FAR	Architecture
[10]	CIDDS-001	10	99.92% multi-class	0.09	Centralized
[11]	NSL-KDD	50	96.5% multi-class	1.29	Centralized
[18]	CICIoT2023	10	99.36% binary-class	–	Distributed
[19]	NSL-KDD	100	96.07% binary-class	7.03	Distributed
[45]	TON/IIoT	300	98.02% multi-class	–	Distributed
[46]	NSL-KDD	100	98.91% multi-class	–	Distributed
[47]	CICIoT2023	–	99.99% binary-class 99.96% multi-class	100 records misclassified 41 records misclassified	–

(Continued)

Table 4 (continued)

IDS	Dataset	Epochs	Accuracy	FAR	Architecture
Proposed model	CICIoT2023	20	99.10% binary-class	0.38 binary-class	Centralized
	NSL-KDD		92.92% multi-class	0.10 multi-class	
			99.22% NSL-KDD	0.32 NSL-KDD	
	KDD-99		99.94% KDD-99	0.12 KDD-99	

Second, the comparison with previous studies that proposed lightweight IDSs [12,14–17] reveals various methodologies developed to create efficient systems for resource-constrained environments. The authors in Reference [17] focus on reducing the size of features and data points required for abnormality detection, thereby lowering computational complexity. They address system overhead in terms of CPU and memory usage, with their model utilizing a constant 3.4% CPU and a range of 1.5% to 2.9% memory overhead. Papers [15,16] proposed IDS nodes and simulated their performance using OMNET++. In terms of lightweight IDSs, the studies primarily focus on energy consumption and latency. The authors in [16] investigated the performance of Raspberry Pi and Orange Pi used in different layers of the fog computing (FC) system, recording parameters to simulate the system using OMNET++. Papers [12,14] employed methodologies similar to our study, testing model performance on the Raspberry Pi. These papers address energy consumption and CPU time usage concerning the number of nodes and grams. Table 5 compares our proposed model with previous studies in terms of energy consumption, detection latency, and accuracy. Our model outperforms [16] in accuracy for the NSL-KDD and KDD-99 datasets and achieves higher accuracy rates for binary classifications on the CICIoT2023 dataset. The energy consumption varies slightly depending on the classification task and dataset used. Although Reference [16] reported marginally lower detection latency, our proposed model's detection times remain reasonably low, ensuring timely identification of intrusions. The maximum latency is approximately 8 s for prediction, with energy consumption at 6 watts. In conclusion, the proposed CNN-LSTM model is accurate, energy-efficient, and lightweight.

Table 5: Comparison of proposed model with lightweight IDS

IDS	Algorithm	Dataset	Accuracy	Energy consumption	Latency of detection
[16]	ANN	KDD-99	98.8%	2.45 w	6.07 s
		NSL-KDD	96.7%		6.27 s

(Continued)

Table 5 (continued)

IDS	Algorithm	Dataset	Accuracy	Energy consumption	Latency of detection
Proposed model	CNN-LSTM	CICIoT2023	99.10% binary	6.04 w binary	6.76 s binary
		NSL-KDD	92.92% multi 99.22% NSL	6.12 w multi 6.02 w NSL	8.23 s multi 8.11 s NSL
		KDD-99	99.94% KDD-99	6.00 w KDD	7.52 s KDD

6 Conclusion and Future work

This work focused on developing a lightweight Intrusion Detection System (IDS) for IoT systems in fog computing environments. By leveraging hybrid machine learning models, specifically Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks, and utilizing the CICIoT2023 dataset, we aimed to create an energy-efficient, accurate IDS suitable for resource-constrained environments.

The proposed model was evaluated using multiple datasets, including NSL-KDD, KDD-99, and CICIoT2023, for both binary and multi-class classification tasks. The results demonstrated significant improvements in accuracy and latency with each iteration, highlighting the robustness and effectiveness of our approach. The KDD-99 dataset produced the highest accuracy results, while the CICIoT2023 dataset showed the lowest false alarm rates for multi-class classification.

The proposed model was implemented on a Raspberry Pi, simulating a fog node to test its practicality in resource-limited settings. The findings indicated that the model maintained high accuracy and low latency, even on such low-power devices, with the maximum power consumption recorded during the data preprocessing step for the multi-class CICIoT2023 dataset.

The limitation of this work is that the performance and energy efficiency of the developed light machine learning models are influenced by the specific hardware platform used. Additionally, the complexity and size of a dataset can significantly impact the time and computational resources needed for feature selection algorithms, making it challenging to balance faster execution with effective feature identification.

Future work could extend the evaluation to different fog computing scenarios, assessing scalability to handle a large number of IoT devices, and considering variations in network architectures, and traffic volumes. We recommend performing further experiments focusing on CPU and power consumption rates to refine the model's efficiency.

Acknowledgement: The authors would like to acknowledge the support of the Computer Engineering Department at King Fahd University of Petroleum and Mineral for this work.

Funding Statement: This work was supported by the interdisciplinary center of smart mobility and logistics at King Fahd University of Petroleum and Minerals (Grant number INML2400).

Author Contributions: Hawazen Alzahrani, conducted the primary research and experiments, developed the lightweight model, implemented it on energy-limited devices, performed extensive experiments to evaluate the model's performance, analyzed the data, and drafted the initial manuscript. Tarek Sheltami, provided guidance and supervision throughout the research work, assisted in conceptualizing the research problem and methodology, reviewed and edited the manuscript, offering critical insights and revisions. Abdulaziz Barnawi, provided feedback on the experimental design and analysis, and reviewed the manuscript, suggesting improvements to enhance its clarity and impact. Muhammad Imam, reviewed the manuscript, providing constructive feedback and suggestions for improvement. Ansar Yaser, contributed to interpreting results and their implications for practical applications. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are openly available in (CIC) at https://www.unb.ca/cic/datasets/iotdataset-2023.html (accessed on 20 May 2024).

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of Things security: A survey," *J. Netw. Comput. Appl.*, vol. 88, no. 4, pp. 10–28, 2017. doi: 10.1016/j.jnca.2017.04.002.

[2] H. F. Atlam, R. J. Walters, and G. B. Wills, "Fog computing and the internet of things: A review," *Big Data Cogn. Comput.*, vol. 2, no. 2, 2018, Art. no. 10.

[3] S. Hajiheidari, K. Wakil, M. Badri, and N. J. Navimipour, "Intrusion detection systems in the internet of things: A comprehensive investigation," *Comput. Netw.*, vol. 160, no. 5, pp. 165–191, 2019. doi: 10.1016/j.comnet.2019.05.014.

[4] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019. doi: 10.1186/s42400-019-0038-7.

[5] M. A. Lawal, R. A. Shaikh, and S. R. Hassan, "Security analysis of network anomalies mitigation schemes in IoT networks," *IEEE Access*, vol. 8, pp. 43355–43374, 2020. doi: 10.1109/ACCESS.2020.2976624.

[6] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis and X. Bellekens, "Machine learning based IoT intrusion detection system: An MQTT case study (MQTT-IoT-IDS2020 dataset)," in *Proc. Int. Netw. Conf.*, Honolulu, HI, USA, Springer International Publishing, Sep. 2020, pp. 73–84.

[7] Y. Chen, N. Ashizawa, C. K. Yeo, N. Yanai, and S. Yean, "Multi-scale self-organizing map assisted deep autoencoding Gaussian mixture model for unsupervised intrusion detection," *Knowl.-Based Syst.*, vol. 224, no. 1, 2021, Art. no. 107086. doi: 10.1016/j.knosys.2021.107086.

[8] J. Pacheco, V. H. Benitez, L. C. Felix-Herran, and P. Satam, "Artificial neural networks-based intrusion detection system for internet of things fog nodes," *IEEE Access*, vol. 8, pp. 73907–73918, 2020. doi: 10.1109/ACCESS.2020.2988055.

[9] S. U. Jan, S. Ahmed, V. Shakhov, and I. Koo, "Toward a lightweight intrusion detection system for the internet of things," *IEEE Access*, vol. 7, pp. 42450–42471, 2019. doi: 10.1109/ACCESS.2019.2907965.

[10] I. Ullah, B. Raza, S. Ali, I. A. Abbasi, S. Baseer and A. Irshad, "Software defined network enabled fog-to-things hybrid deep learning driven cyber threat detection system," *Secur. Commun. Netw.*, vol. 2021, pp. 1–15, 2021. doi: 10.1155/2021/6136670.

[11] K. Kalaivani and M. Chinnadurai, "A hybrid deep learning intrusion detection model for fog computing environment," *Intell. Autom. Soft Comput.*, vol. 30, no. 1, pp. 1–15, 2021. doi: 10.32604/iasc.2021.017515.

[12] B. S. Khater, A. W. A. Wahab, M. Y. I. Idris, M. A. Hussain, and A. A. Ibrahim, "A lightweight perceptron-based intrusion detection system for fog computing," *Appl. Sci.*, vol. 9, no. 1, 2019, Art. no. 178. doi: 10.3390/app9010178.

[13] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu and A. A. Ghorbani, "CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment," *Sensors*, vol. 23, no. 13, 2023, Art. no. 5941. doi: 10.3390/s23135941.

[14] B. S. Khater *et al.*, "Classifier performance evaluation for lightweight IDS using fog computing in IoT security," *Electronics*, vol. 10, no. 14, 2021, Art. no. 1633. doi: 10.3390/electronics10141633.

[15] F. Aliyu, T. Sheltami, A. Mahmoud, L. Al-Awami, and A. Yasar, "Detecting Man-in-the-Middle attack in fog computing for social media," *Comput. Mater. Continua.*, vol. 69, no. 1, pp. 1159–1181, 2021. doi: 10.32604/cmc.2021.016938.

[16] F. Aliyu, T. Sheltami, M. Deriche, and N. Nasser, "Human immune-based intrusion detection and prevention system for fog computing," *J. Netw. Syst. Manag.*, vol. 30, no. 1, pp. 1–27, 2022. doi: 10.1007/s10922-021-09616-6.

[17] S. Roy, J. Li, B. J. Choi, and Y. Bai, "A lightweight supervised intrusion detection mechanism for IoT networks," *Future Gener. Comput. Syst.*, vol. 127, pp. 276–285, 2022. doi: 10.1016/j.future.2021.09.027.

[18] A. A. Wardana, G. Kołaczek, and P. Sukarno, "Lightweight, trust-managing, and privacy-preserving collaborative intrusion detection for internet of things," *Appl. Sci.*, vol. 14, no. 10, 2024, Art. no. 4109. doi: 10.3390/app14104109.

[19] A. Samy, H. Yu, and H. Zhang, "Fog-based attack detection framework for internet of things using deep learning," *IEEE Access*, vol. 8, pp. 74571–74585, 2020. doi: 10.1109/ACCESS.2020.2988854.

[20] C. A. De Souza, C. B. Westphall, and R. B. Machado, "Two-step ensemble approach for intrusion detection and identification in IoT and fog computing environments," *Comput. Electr. Eng.*, vol. 98, no. 15, 2022, Art. no. 107694. doi: 10.1016/j.compeleceng.2022.107694.

[21] A. Rai, "Optimizing a new intrusion detection system using ensemble methods and deep neural network," in *2020 4th Int. Conf. Trends Electron. Inform. (ICOEI)*, Tirunelveli, India, IEEE, Jun. 2020, pp. 527–532.

[22] K. Sadaf and J. Sultana, "Intrusion detection based on autoencoder and isolation forest in fog computing," *IEEE Access*, vol. 8, pp. 167059–167068, 2020. doi: 10.1109/ACCESS.2020.3022855.

[23] A. A. Abdussami, "Incremental deep neural network intrusion detection in fog based IoT environment: An optimization assisted framework," *Indian J. Comput. Sci. Eng.*, vol. 12, no. 6, pp. 1847–1859, 2021. doi: 10.21817/indjcse/2021/v12i6/211206191.

[24] C. A. de Souza, C. B. Westphall, R. B. Machado, J. B. Mangueira Sobral, and G. D. Santos Vieira, "Hybrid approach to intrusion detection in fog-based IoT environments," *Comput. Netw.*, vol. 180, no. 7, 2020, Art. no. 107417. doi: 10.1016/j.comnet.2020.107417.

[25] P. Sun *et al.*, "DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system," *Secur. Commun. Netw.*, vol. 2020, pp. 1–11, 2020. doi: 10.1155/2020/8890306.

[26] D. Attique, H. Wang, and P. Wang, "Fog-assisted deep-learning-empowered intrusion detection system for RPL-based resource-constrained smart industries," *Sensors*, vol. 22, no. 23, 2022, Art. no. 9416. doi: 10.3390/s22239416.

[27] S. Tuli, N. Basumatary, and R. Buyya, "EdgeLens: Deep learning based object detection in integrated IoT, fog and cloud computing environments," in *2019 4th Int. Conf. Inf. Syst. Comput. Netw. (ISCON)*, Mathura, India, IEEE, Nov. 2019, pp. 496–502.

[28] Ü. Çavuşoğlu, D. Akgun, and S. Hizal, "A novel cyber security model using deep transfer learning," *Arab J. Sci. Eng.*, vol. 49, no. 3, pp. 3623–3632, 2023. doi: 10.21203/rs.3.rs-2431742/v1.

[29] Q. Chang, X. Ma, M. Chen, X. Gao, and M. Dehghani, "A deep learning based secured energy management framework within a smart island," *Sustain. Cities Soc.*, vol. 70, no. 10, 2021, Art. no. 102938. doi: 10.1016/j.scs.2021.102938.

[30] W. Jiang and S. Lv, "Hierarchical deployment of deep neural networks based on fog computing inferred acceleration model," *Clust. Comput.*, vol. 24, no. 4, pp. 2807–2817, 2021. doi: 10.1007/s10586-021-03298-0.

[31] S. P. K. Gudla, S. K. Bhoi, S. R. Nayak, and A. Verma, "DI-ADS: A deep intelligent distributed denial of service attack detection scheme for fog-based IoT applications," *Math. Probl. Eng.*, vol. 2022, no. 1, pp. 1–17, 2022. doi: 10.1155/2022/3747302.

[32] S. Singh, K. Kumari, S. Gupta, A. Dua, and N. Kumar, "Detecting different attack instances of DDoS vulnerabilities on edge network of fog computing using gaussian Naive bayesian classifier," in *2020 IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Dublin, Ireland, IEEE, Jun. 2020, pp. 1–6.

[33] M. Roopak, G. Y. Tian, and J. Chambers, "Deep learning models for cyber security in IoT networks," in *in 2019 IEEE 9th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Las Vegas, NV, USA, IEEE, Jan. 2019, pp. 452–457.

[34] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for internet of things," *Future Gener. Comput. Syst.*, vol. 82, no. 6, pp. 761–768, 2018. doi: 10.1016/j.future.2017.08.043.

[35] I. M. Selim and R. A. Sadek, "DAE-BILSTM: A fog-based intrusion detection model using deep learning for IoT," *J. Theor. Appl. Inf. Technol.*, vol. 101, no. 5, pp. 2027–2042, 2023.

[36] A. I. A. Alzahrani, A. Al-Rasheed, A. Ksibi, M. Ayadi, M. M. Asiri and M. Zakariah, "Anomaly detection in fog computing architectures using custom tab transformer for internet of things," *Electronics*, vol. 11, no. 23, 2022, Art. no. 4017. doi: 10.3390/electronics11234017.

[37] J. O. Onah, M. Abdullahi, I. H. Hassan, and A. Al-Ghusham, "Genetic algorithm based feature selection and Naïve Bayes for anomaly detection in fog computing environment," *Mach. Learn. Appl.*, vol. 6, 2021, Art. no. 100156.

[38] P. Kumar, R. Kumar, G. P. Gupta, and R. Tripathi, "A distributed framework for detecting DDoS attacks in smart contract-based Blockchain-IoT systems by leveraging fog computing," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 6, 2021, Art. no. e4112. doi: 10.1002/ett.4112.

[39] A. Almaiah and O. Almomani, "An investigation of digital forensics for shamoon attack behaviour in fog computing and threat intelligence for incident response," *J. Theor. Appl. Inf. Technol.*, vol. 15, p. 98, 2020.

[40] S. Xu, Y. Qian, and R. Q. Hu, "Data-driven network intelligence for anomaly detection," *IEEE Netw.*, vol. 33, no. 3, pp. 88–95, 2019. doi: 10.1109/MNET.2019.1800358.

[41] M. A. Rahman, A. T. Asyhari, L. S. Leong, G. B. Satrya, M. H. Tao and M. F. Zolkipli, "Scalable machine learning-based intrusion detection system for IoT-enabled smart cities," *Sustain. Cities Soc.*, vol. 61, no. 1, p. 102324, 2020. doi: 10.1016/j.scs.2020.102324.

[42] S. Tu *et al.*, "Security in fog computing: A novel technique to tackle an impersonation attack," *IEEE Access*, vol. 6, pp. 74993–75001, 2018. doi: 10.1109/ACCESS.2018.2884672.

[43] P. Illy, G. Kaddoum, C. Miranda Moreira, K. Kaur, and S. Garg, "Securing fog-to-things environment using intrusion detection system based on ensemble learning," in *2019 IEEE Wireless Commun. Netw. Conf. (WCNC)*, Marrakesh, Morocco, IEEE, Apr. 15–18, 2019, pp. 1–7.

[44] N. Moustafa, K. K. R. Choo, I. Radwan, and S. Camtepe, "Outlier Dirichlet mixture mechanism: Adversarial statistical learning for anomaly detection in the fog," *IEEE Trans. Inf. Forensics Secur*, vol. 14, no. 8, pp. 1975–1987, 2019. doi: 10.1109/TIFS.2018.2890808.

[45] T. Gazdar, "FDeep: A fog-based intrusion detection system for smart home using deep learning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 12, pp. 348–355, 2022. doi: 10.14569/issn.2156-5570.

[46] S. P. K. Gudla, S. K. Bhoi, S. R. Nayak, K. K. Singh, A. Verma and I. Izonin, "A deep intelligent attack detection framework for fog-based IoT systems," *Comput. Intell. Neurosci.*, vol. 2022, no. 1, 2022, Art. no. 6967938. doi: 10.1155/2022/6967938.

[47] S. Yaras and M. Dener, "IoT-based intrusion detection system using new hybrid deep learning algorithm," *Electronics*, vol. 13, no. 6, 2024, Art. no. 1053. doi: 10.3390/electronics13061053.

[48] A. R. Gad, M. Haggag, A. A. Nashat, and T. M. Barakat, "A distributed intrusion detection system using machine learning for IoT based on ToN-IoT dataset," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 6, pp. 548–563, 2022. doi: 10.14569/issn.2156-5570.

[49] Y. Labiod, A. Amara Korba, and N. Ghoualmi, "Fog computing-based intrusion detection architecture to protect IoT networks, Wireless Pers," *Wireless Pers. Commun.*, vol. 125, no. 1, pp. 231–259, 2022.

[50] P. F. de Araujo-Filho, G. Kaddoum, D. R. Campelo, A. G. Santos, D. Macêdo and C. Zanchettin, "Intrusion detection for cyber-physical systems using generative adversarial networks in fog environment," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6247–6256, 2020. doi: 10.1109/JIOT.2020.3024800.

[51] Y. Meng, S. Tu, J. Yu, and F. Huang, "Intelligent attack defense scheme based on DQL algorithm in mobile fog computing," *J. Vis. Commun. Image Represent.*, vol. 65, no. 4, 2019, Art. no. 102656. doi: 10.1016/j.jvcir.2019.102656.

[52] R. Priyadarshini, R. K. Barik, and H. Dubey, "Fog-SDN: A light mitigation scheme for DDoS attack in fog computing framework," *Int. J. Commun. Syst.*, vol. 33, no. 9, 2020, Art. no. e4389. doi: 10.1002/dac.4389.

[53] R. Priyadarshini and R. K. Barik, "A deep learning based intelligent framework to mitigate DDoS attack in fog environment," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 3, pp. 825–831, 2022. doi: 10.1016/j.jksuci.2019.04.010.