Investigate the Use of Deep Learning in IoT Attack Detection

Mohamed Saddek Ghozlane, Adlen Kerboua, Smaine Mazouzi, and Lakhdar Laimeche

Original scientific article

***Abstract*—The Internet of Things (IoT) has provided many benefits to society and introduced new security challenges. Attackers can target IoT devices to steal sensitive information or launch large-scale attacks. In this field, deep learning algorithms have provided encouraging results in the discovery and classification of intrusions in IoT devices. This study investigates the implementation and performance of four deep learning models: One-Dimensional Convolutional Neural Network (1DCNN), Long Short-Term Memory (LSTM), a hybrid 1DCNN-LSTM, and Two-Dimensional Convolutional Neural Network (2DCNN) for detecting and classifying IoT device attacks. Using the BoTNeTIoT-L01-v2 dataset, which includes normal and attack traffic provided by various IoT devices, we preprocess the data, extract features, and train the models, including weighted versions to optimize feature importance. Our findings highlight that the 2DCNN and hybrid 1DCNN-LSTM models shows superior performance, achieving high classification accuracy. This study contributes a comprehensive comparative analysis of deep learning models for IoT security, focusing on the effectiveness of weighted features in improving detection accuracy. The results provide valuable information for the advancement of real-time IoT attack detection systems.**

***Index Terms*—Deep learning, IoT, Attack, BotNet.**

I. Introduction

THE Internet of Things has revolutionized communication and connectivity, particularly in industrial applications [1], enabling smart spaces, intelligent transportation, and seamless device interaction. However, the exponential growth of IoT devices has also expanded the attack surface, allowing malicious actors to exploit vulnerabilities to gain unauthorized access, disrupt critical systems, and cause significant financial and operational damage. These challenges require robust security measures capable of detecting and mitigating attacks in real time.

Deep learning techniques have emerged as a powerful tool for IoT security, using the ability of the neural network to identify intricate patterns and anomalies in large real-time data streams. These methods surpass traditional rule-based systems, offering adaptability to evolving attack patterns and reducing false positives, thereby minimizing manual intervention and operational costs. However, the performance of deep learning models is dependent on effective feature selection and model architecture.

Although deep learning has shown significant potential in IoT security, previous studies have focused primarily on individual model architecture without a comprehensive comparative analysis of different deep learning approaches. This study looks at four deep learning models—LSTM, 1DCNN, hybrid 1DCNN-LSTM, and 2DCNN—for detecting IoT attacks, showing what each model does well and where it falls short. In addition, we introduce a feature weighting mechanism within these models that optimizes the importance of input features during training. Unlike prior research, our study not only compares these models but also examines how the importance of features affects detection performance. This provides valuable insights into model selection and optimization for real-time IoT security applications:

- Investigating the performance of four deep learning models (LSTM, 1DCNN, hybrid 1DCNN-LSTM, and 2DCNN) in detecting IoT device attacks.
- Proposing weighted versions of these models to enhance feature selection during the training process.
- Conduct a detailed comparative analysis of the models' performance, providing insight into their strengths and limitations.

The remainder of this paper is organized as follows. Section II reviews related work, highlighting gaps in IoT attack detection research. Section III provides an overview of the proposed method, while Section IV details the experimental setup and results. Section V discusses the findings, and Section VI concludes with future research directions.

II. Related Work

The Internet of Things (IoT) has rapidly evolved, offering numerous applications in industrial automation, healthcare, and smart cities. However, this expansion has also introduced significant security vulnerabilities, making IoT devices a prime target for various cyberattacks. Deep learning-based approaches have emerged as powerful solutions for detecting and mitigating these attacks due to their ability to identify complex patterns and anomalies in large datasets.

In most of the related literature, the classification of deep learning-based IoT security techniques is shown in Fig.1.

Manuscript received November 6, 2024; revised December 3, 2024. Date of publication May 23, 2025. Date of current version May 23, 2025. The associate editor prof. Teodoro Montanaro has been coordinating the review of this manuscript and approved it for publication

M. S. Ghozlane is with the Department of Petrochemical, University of Skikda and with Department Computer Science, University of Tebessa, Algeria.

A. Kerboua is with the Department of Petrochemical and LGMM Laboratory, University of Skikda, Algeria (corresponding author: ad.kerboua@univ-skikda.dz).

S. Mazouzi is with the Department of Computer Science and LICUS Laboratory, University of Skikda, Algeria.

L. Laimeche is with the Department Computer Science, University of Tebessa, Algeria

Digital Object Identifier (DOI): 10.24138/jcomss-2024-0101

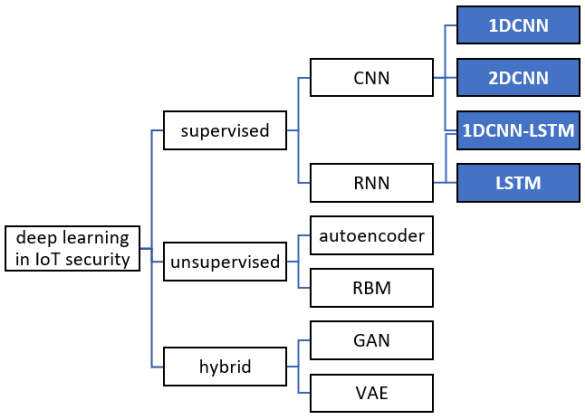


Fig. 1. Taxonomy of the deep learning-based IoT attack detection.

- Supervised: Labeled datasets are used to train deep learning models to classify and detect security threats to the IoT. Examples include CNNs, LSTMs, and Support Vector Machines (SVMs).
- Unsupervised: using unlabeled data to train deep learning models to identify anomalies in the data that may indicate security problems. Examples that use autoencoders, Generative Adversarial Networks (GANs), and Self-Organizing Maps (SOMs),
- Hybrid: methods combining both supervised and unsupervised learning techniques to increase the precision and effectiveness of identifying anomalies and intrusions in IoT security. Examples include Deep Belief Networks (DBNs), stacked autoencoders, and hybrid Convolutional and Recurrent Neural Networks (CRNNs).

Before diving into the heart of the matter, we review some recent related works that use deep learning to detect IoT attacks, along with the methods and datasets used, the advantages, and the limitations.

Several studies have explored the use of deep learning techniques for IoT security. Ullah and Mahmoud [2] proposed a stacked autoencoder model to detect anomalies in IoT networks, achieving high precision in public datasets. Similarly, Shone et al. [3] implemented a deep learning framework combining autoencoders and SoftMax classifiers for intrusion detection, demonstrating superior performance compared to traditional machine learning models.

In [4] and [5], they propose a distributed attack detection scheme integrating big data analytics using deep learning for IoT networks, showcasing significant improvements in detection accuracy and scalability for various types of attacks. The approach is computationally intensive and may not be practical for real-time detection in resource-constrained IoT environments.

In [6], Roopak et al. introduced a hybrid model that integrates convolutional neural networks (CNNs) with long- and short-term memory (LSTM) networks to analyze IoT traffic data. Their approach effectively captured spatial and temporal dependencies in the data, achieving promising results for real-time attack detection.

The authors of [7] talk about how deep learning can be used to improve cybersecurity for IoT networks. For intrusion detection, they make use of an RNN and a CNN. The performance of these models is thoroughly analyzed by the authors, who also draw comparisons between them and more conventional machine learning models.

A comprehensive overview of deep learning models for IoT object security is given in the work [8]. It considers authentication, intrusion detection, privacy preservation, and malware detection. The article also highlights the challenges and limitations of these techniques and provides insight into future research directions.

The work [9] discusses the security issues caused by deep learning in the IoT. It particularly highlights the overloads encountered in securing connected objects due to the heterogeneity of IoT objects and communication protocols, as well as the limited computing and storage resources available on these machines.

A concept for utilizing big data and deep learning to protect IoT devices is presented in the study [10]. The essay outlines the inadequacies of conventional security techniques as well as the security difficulties facing the Internet of Things. The authors propose the use of these technologies to address security challenges in the Internet of Things.

An assembly study utilizing deep learning and machine learning-based intrusion detection systems (IDSs) for the Internet of Things networks is provided in [11]. The authors discuss deep machine learning techniques used for IDSs, particularly deep neural networks.

A thorough overview of deep learning-based security behavior analysis in IoT networks is given by Yue et al. [12]. The article focuses on using this type of learning to detect anomalous behavior and improve IoT security.

The authors of [13] suggest using federated deep learning (FDL) to create a global deep learning model that will guarantee the security of connected IoT items without requiring the transfer of sensitive data to a centralized server.

In [14], the authors offer a thorough analysis of deep learning techniques for Internet of Things security. The authors reviewed 33 research articles and categorized them according to the type of security threats addressed, the datasets, and the deep learning models that were employed.

A general analysis of deep learning approaches used for the cybersecurity of connected objects can be found in the paper by Ahmed and Askar [15]. The study also covers numerous deep learning methods that have been used to identify different kinds of cyberattacks in IoT networks. Additionally, the article has a more in-depth discussion on the main limitations and drawbacks of these deep learning techniques.

In [16], the authors propose a deep learning algorithm based on image recognition in IoT applications. The algorithm uses CNNs. The authors apply this algorithm by conducting experiments on a dataset of various objects and achieve an accuracy of 97.5%.

Sarker et al. [17] give a thorough summary of the difficulties with IoT security. They discuss the limitations.

of traditional security mechanisms that are not effective in the dynamic and heterogeneous IoT environment. The study addresses several machine learning methods and their possible

uses in IoT security, including supervised, unsupervised, and reinforcement learning.

An overview of supervised deep learning techniques for protecting connected things (IoT) is provided by Abdel-Basset et al. [18]. The authors discuss the possibilities and difficulties of using these techniques for IoT security. In addition, the paper offers information and draws attention to the shortcomings of the deep learning models used today for IoT security.

Finally, the article [19] proposes an IoT security solution using deep learning mechanisms. The authors propose a framework consisting of three main components: data acquisition, data pre-processing, and deep learning-based security mechanisms. The proposed framework is evaluated with a dataset of different IoT gadgets, and the findings indicate that the suggested deep learning-based mechanism can accurately detect attacks on IoT networks.

Despite these advancements, certain limitations persist. Many existing studies rely heavily on simulated datasets, limiting their practical applicability in real-world IoT environments. Additionally, while hybrid models have shown improved accuracy, there is a lack of comprehensive comparative analyses to determine the most effective architectures for specific attack types. Furthermore, the role of feature weighting in improving model performance remains underexplored.

A. Research Gap and Problem Statement

Although deep learning has proven to be a powerful tool for IoT security, significant gaps remain in the current body of research. First, most existing studies focus on a single type of attack or dataset, overlooking the diverse nature of IoT traffic and threats. Second, the effectiveness of feature-weighted models has not been thoroughly investigated, particularly for detecting complex attack patterns. Third, there is a lack of clarity on which deep learning architectures are best suited for real-time IoT attack detection under varying conditions.

To address these gaps, this study provides a comprehensive comparative analysis of four deep learning models: LSTM, 1DCNN, hybrid CNN-LSTM, and 2DCNN. Additionally, it introduces weighted versions of these models to evaluate the impact of feature weighting on classification performance. We ensure practical relevance by using the BoTNeTIoT-L01-v2 dataset, which covers various types of attacks and IoT traffic. This research aims to bridge the gap by identifying robust models and methods for real-time IoT attack detection and proposing strategies to improve their effectiveness.

III. Overview of the Proposed Method

The main purpose of this study is to evaluate the role that deep learning models play in the classification of IoT security data. Specifically, the investigation focuses on comparing the performance of four different models: LSTM, 1DCNN, hybrid 1DCNN-LSTM, and 2DCNN.

Recurrent neural networks (RNNs), which are useful for processing sequential data such as time series data, include the LSTM network.

The 1DCNN model, on the other hand, uses convolutional layers to extract features from a one-dimensional signal.

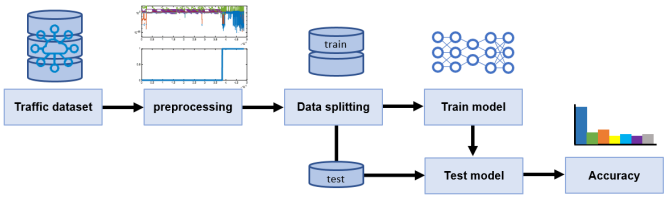


Fig. 2. Overview of the proposed framework.

The combination of convolutional and recurrent layers in the hybrid CNN-LSTM model enables it to efficiently capture temporal and spatial dependencies in the data.

Finally, the 2DCNN model processes raster data, such as 2D spectrogram images, created by converting time series signals into visual representations.

To evaluate the performances of these models, this study uses the BoTNeTIoT-L01-v2 dataset, a public dataset that includes network traffic captured from various IoT devices. The dataset encompasses a range of attacks, such as SQL injection, DDoS, brute-force attacks, and botnet attacks. Each model is trained and evaluated using this dataset, with preprocessing steps including normalization, feature extraction, and data splitting into training and testing sets.

In addition to evaluating these models, the study explores weighted versions of each model, where a learnable weight is assigned to each feature during training. The weights are optimized within the training loop to enhance the model's ability to focus on features most relevant to IoT attack detection.

The main steps of the proposed method (Fig.2) are summarized below:

1) *Dataset Preprocessing:* Preprocess the BoTNeTIoT-L01-v2 dataset to make sure the input data is of high quality, which involves normalizing the data, extracting important features, and splitting it into training and testing sets.
2) *Model Design:* Implement four deep learning models (LSTM, 1DCNN, CNN-LSTM, and 2DCNN) with fully connected layers, dropout layers, and a SoftMax classification layer.
3) *Weighted Models:* Develop weighted versions of all models by incorporating optimized learnable feature weights during training to improve classification accuracy.
4) *Model Training:* Train each model on the processed dataset using hyperparameters tuned for optimal performance, as detailed in Section IV.
5) *Performance Evaluation:* Evaluate model performance using accuracy. Compare the results to identify the most effective model for IoT attack detection.

An important aspect of this work is the inclusion of spectrogram representations for the 2DCNN model. By converting raw time series data into spectrograms, we allow the 2DCNN model to leverage spatial features, enhancing its ability to detect complex attack patterns.

The steps outlined above provide a structured approach to comparing and improving the performance of deep learning

models for IoT attack detection, offering valuable insights into their strengths and limitations.

IV. Details of the Proposed Approach

The proposed study plans to compare how well four deep learning models, called LSTM, 1DCNN, 1DCNN-LSTM, and 2DCNN, perform in classifying IoT security tasks. In addition, a weighted version of these models is also proposed to weight the features used in training.

The LSTM model represents a family of recurrent neural networks that are well suited to serial data processing. It has been used effectively for numerous time series analyses and natural language processing tasks, and it is capable of learning long-term patterns.

The 1DCNN model also represents a family of convolutional neural network models used to process time series data. This is particularly useful for detecting local patterns in the data.

The combination of the 1DCNN-LSTM model can identify short-term and long-term patterns in sequential data because it integrates the best features of the CNN and LSTM models.

The 2DCNN model is commonly used for processing two-dimensional data, such as images. It has been adapted for use in IoT security classification tasks by treating sensor data as an image.

In addition to those models, we designed a weighted version of each model, assigning a weight to each feature based on its importance in the classification task. The weights are then used to adjust the contribution of each feature to the overall classification score.

The general goal of the proposed study is to assess how well these four deep learning models perform on IoT security classification tasks and whether employing a weighted feature selection strategy may increase classification accuracy.

A. Proposed Deep Models

In this subsection, we outline the four suggested structures of deep learning models. which are the LSTM, 1DCNN, CNN-LSTM, and 2DCNN models. Every model stacks a fully connected layer at its top. A loss layer, which is a mathematical function that determines the error rate between the actual and the ground-truth data, is positioned at the top of each model. The optimization technique then computes the category cross-entropy and iterates to minimize this loss. The SoftMax layer must come before the classification layer in most classification jobs. The classification layer uses the cross-entropy function to assign each input to one of the K mutually exclusive classes during the training phase after taking the values from the SoftMax activation function [20]:

$$\text{loss} = \sum_{i=1}^{N} \sum_{j=1}^{K} t_{\text{ij}} \log\left(y_{\text{ij}}\right) \tag{1}$$

This formula is as follows: N is the number of observations, K is the number of classes, t_{ij} denotes that the i^{th} sample is a member of the j^{th} class, and y_{ij} is the output, in this case the value obtained from the SoftMax function, for sample i from class j. Stated differently, it represents the likelihood that the network links the i^{th} input to the class j.

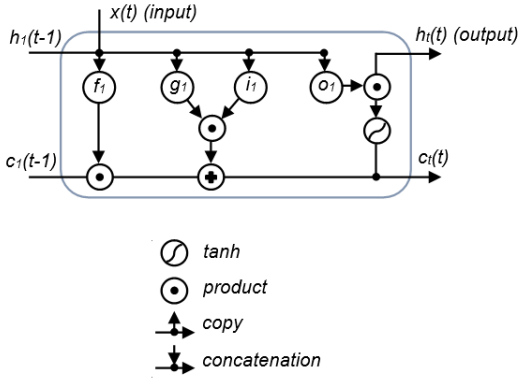


Fig. 3. Internal structure of an LSTM cell.

To improve feature selection during model training, we implemented a feature weighting mechanism in which each feature is assigned a learnable weight. These weights are initialized randomly and optimized through backpropagation using the Adam optimizer. This ensures that more informative features contribute significantly to model predictions.

B. LSTM Model

LSTM is an RNN-family model formalized to capture long-term dependencies in serial data, such as time series or natural language data. In 1997, Hochreiter and Schmidhuber developed it [21]. LSTM models leverage memory cells to store data for an extended period and use gates to regulate the flow of information inside and outside the cell. The three types of gates in an LSTM are the following:

1) Forget gate: controlling which data are deleted from the cell,
2) Front gate: controlling what new information is added to the cell,
3) Exit gate: controlling what information will leave the cell.

Gates regulate data flow by producing values between 0 and 1 using sigmoid activation functions. The memory cell modifies its state through the use of a tanh activation function.

The structure of an LSTM model (Fig.3) allows it to save or forget through data selection, making it well suited to activities that require long-term memory, such as time series prediction, language translation, and speech recognition. For time series analysis and prediction tasks, LSTM has been widely employed. Time series data analysis using LSTM is applicable to IoT attack detection, collected from various sensors and devices in the network to detect anomalous behavior.

To use LSTM for IoT attack detection, the dataset must be preprocessed to represent it in a suitable format. For example, the $BoTNeTIoT - L01 - v2$ dataset can be represented as a time series of sensor readings over a certain period. The LSTM model can then be trained on these time series data to learn the normal patterns of behavior in the network. During testing, the model can detect any deviations from the learned patterns and flag them as potential attacks.

Since LSTM can handle time series data with long-term dependencies, it is a useful tool to detect IoT attacks. This means that the model can capture patterns that occur over a longer period, which may be indicative of more sophisticated attacks. However, one limitation of using LSTM is that it may require a large flow of training data to learn the normal behavior architectures in the network. Additionally, the model may have difficulty detecting new or previously unseen attack patterns that were not present in the training data.

Methods like anomaly detection and transfer learning can be used to improve the model's performance in order to solve this problem. The model is made up of a fully connected layer, a SoftMax layer, and two LSTM layers with dropout. This model is designed for sequence classification tasks with a single input sequence and multiple output classes. The model can identify long-term dependencies in the input data series using LSTM layers, and the overfitting can be minimized using dropout layers with a value of 0.2. The fully connected layer provides additional learning capacity, and the SoftMax layer ensures that the output probabilities sum up to 1.

C. 1DCNN Model

The 1DCNN is a deep learning model used for sequence-based data analysis. It has been applied to IoT attack detection to identify patterns in time series data generated by IoT devices. In 1DCNN-based IoT attack detection, in general, input data are shown as a time series or a one-dimensional array. A convolutional layer that applies filters to the input data is combined with a pooling layer to reduce the size of the feature maps to form the 1DCNN architecture.

The final output is then created by combining the output of the pooling layer with one or more fully linked layers. The 1DCNN is a mathematical calculation that uses two functions, called the input signal and kernel, to generate a third function, which is the output signal. For an input signal x of length N and a kernel h of length K, the output signal y can be calculated as follows:

$$x_i = \sum_{k=1}^{k} h_k * x_{n-k} \tag{2}$$

Here, n is the output signal index and k is the convolutional kernel index. This equation represents the computation of a single output value $y[n]$ using a sliding window of length K over the input signal x, where the kernel h is multiplied element-wise with the corresponding elements of the input signal x in the window, and the output value $y[n]$ is calculated by adding the resultant products.

The convolution operation can be seen to filter or transform the input signal using the filter. Depending on the choice of this kernel, different types of filters or transformations can be applied to the input signal, such as smoothing, edge detection, feature extraction, etc.

Signal processing, audio and picture processing, natural language processing, and many more domains make extensive use of 1D convolution. The 1DCNN-based IoT attack detection approach has several advantages, including the ability to manage large data streams, flexibility in the representation of inputs, and good performance in identifying temporal patterns in the data. However, it may suffer from overfitting when the dataset is small, and it may not perform well on non-temporal data.

To improve the performance of 1DCNN-based IoT attack detection, techniques such as data augmentation and regularization can be applied. Additionally, weighted versions of the 1DCNN model (W1D CNN) can be used to focus the model on the most informative features of the data, which can lead to improved accuracy due to the variation of the features used. We can even enhance the detection robustness of our model by using a weighted feature fusion method; the weighted feature vector is computed as follows:

$$x_{\mathrm{wi}} = x_i * w_i \tag{3}$$

Where x_i is the vector of features i, and x_{wi} is the corresponding learnable weight, which is initialized randomly and then tuned during the training process using backpropagation optimization.

D. 1DCNN-LSTM Model

1DCNN-LSTM is a model that integrates one-dimensional convolution and LSTM layers. This model is useful for capturing both the temporal and spatial fields of a time series of data. In the context of IoT attack detection, the 1DCNN-LSTM architecture can be used to capture occurrences of relevant input data and classify whether an attack is present or not. The input data first pass through the 1DCNN layers, which apply a set of convolutional filters to extract spatial features from time series data. Next, the 1DCNN layer's output is combined with the LSTM layer's task of identifying temporal connections in the data. The LSTM layer has cells and memory gates for saving and forgetting data from the previous time steps in a selective manner, which allows it to formalize long-term data dependencies.

The 1DCNN-LSTM model is trained using a process like that of the LSTM and 1DCNN models. It is fed by labeled training data, and stochastic gradient descent is one optimization approach that is used to update the model weights. The weighted 1DCNN-LSTM model can also be used to weight the features used to train the model (W1D-CNN-LSTM), giving more importance to the relevant features that can improve the model's accuracy. Overall, the 1DCNN-LSTM model can provide an effective solution for the detection of IoT attacks, especially when the attack patterns have temporal and spatial dependencies.

E. 2D-CNN

A 2D-CNN can also be used for IoT attack detection, especially when it comes to image-based features such as spectrograms extracted from time series signals. This image can reflect the signal's frequency content over time; this can generate pertinent data that can be utilized to identify irregularities in the traffic on IoT networks.

To extract characteristics from the spectrogram, the input traffic data is preprocessed using signal processing techniques like the Short-Time Fourier Transform (STFT).

V. Results and Discussion

This section presents the evaluation results of the proposed deep learning models (LSTM, 1DCNN, CNN-LSTM, and 2DCNN) for IoT attack detection. The analysis includes their performance on detecting specific attack types and comparisons between weighted and non-weighted models.

These results demonstrate the potential of deep learning in IoT security and underscore the importance of feature weighting in improving model performance.

A. Dataset Description

The BoTNeTIoT-L01-v2 dataset is a publicly available dataset that consolidates IoT network traffic data, including both normal and attack samples. It is widely used to evaluate intrusion detection systems and includes various IoT devices and attack scenarios. The dataset captures network traffic under various conditions, with labels that indicate whether the traffic is normal (label 0) or anomalous (label 1).

The dataset includes a range of attack types, such as Mirai and Gafgyt botnet attacks, SQL injection, DDoS, brute-force, and scanning attacks. These attacks target vulnerabilities in IoT devices like smart cameras, routers, and home automation systems. Table I provides the distribution of attack types and normal traffic in the dataset.

Upon further examination, inaccuracies in the dataset labels were identified. Specifically, some samples labeled "1" (anomalous) were found to represent normal traffic, and vice versa. These discrepancies were corrected during preprocessing to ensure data integrity. Additionally, the dataset exhibits a significant class imbalance, with normal traffic samples vastly outnumbering attack samples. For example, approximately 92% of the samples represent normal traffic, while only 8% are labeled attacks. This imbalance can introduce bias during model training, leading to overfitting toward the majority class.

The dataset was preprocessed to include features such as packet size, protocol type, flow duration, and time-based statistics. These features are critical in distinguishing between normal and malicious traffic.

To address the identified biases, several preprocessing techniques were applied:

- *Data Resampling:* Oversampling of attack samples and undersampling of normal samples were performed to balance the dataset distribution.
- *Feature Normalization:* Continuous features such as packet size, flow duration, and inter-packet arrival times were normalized to ensure uniform scaling across all data points.
- *Augmentation:* Synthetic data generation techniques were used to create additional attack samples, simulating underrepresented attack types like SQL injection.
- *Class Weighting:* Weighted loss functions were utilized during model training to penalize misclassification of minority-class samples more heavily, ensuring balanced learning.

These measures helped mitigate bias and improve the generalizability of the deep learning models when tested on imbalanced datasets. Moreover, the dataset preprocessing steps ensured accurate representation of the IoT network traffic, enabling robust evaluations of the proposed models.

TABLE I
Dataset statistics.

State	Designation	Count	Percent
Label	0	650668	92.1285
	1	55593	7.8715
Attack	Normal	55 593	7.8715
	Gafgyt attack	283827	40.1873
	Mirai	366841	51.9413
$Attack_subType$	Normal	55 593	7.8715
	Ack	64383	9.1160
	Combo	51516	7.2942
	Junk	26179	3.7067
	Scan	79309	11.2294
	Syn	73331	10.3830
	Tcp	85983	12.1744
	Udp	217637	30.8154
	Udpplain	52330	7.4094

In Table I, we can see that the dataset includes a total of 10 attack scenarios and normal traffic data. The attacks include various types of botnet attacks, such as Mirai, Ack, and Combo attacks. The dataset also includes data from different IoT objects, such as cameras, routers, and smart home devices.

B. BoTNeTIoT-L01-V2 Features

The BoTNeTIoT-L01-v2 dataset contains a total of 115 features, which are a combination of network traffic and system-level features. According to [22], the following groups can be used to broadly classify the features:

1) Fundamental features: These consist of elements such as the protocol type, packet size, port numbers, source and destination IP addresses, and number of packets,
2) Statistical features: These include various statistical measures computed over the network traffic, such as the mean, standard deviation, variance, skewness, kurtosis, minimum, maximum, and entropy,
3) Time-based features: These features capture the temporal characteristics of the network traffic and include features such as the inter-arrival time between packets, duration of flows, and time-based statistics,
4) Content-based features: These features capture the payload of the network traffic and include features such as the size, entropy, and payload-specific statistics,
5) Control flow-based features: These features capture the control flow of the network traffic and include features such as the number of SYN, ACK, and FIN packets,
6) IoT-specific features: These features capture the unique characteristics of oT traffic and include details like the quantity of distinctive IoT devices, distinct IoT protocols, and quantity of IoT-specific packets.

The features in the dataset are designed to capture a wide range of network and system-level characteristics that are relevant for IoT security and attack detection. In order to select the most relevant features in a dataset based on their relative importance, we use a learnable weight vector inside a deep learning model, which involves incorporating the feature weighting mechanism as part of the model learning process, and the weight vector is learned through the model training

process and optimized to minimize the loss function of the model.

The learnable weight vector technique provides flexibility in the feature-weighting mechanism, as it allows the model to learn the optimal weights for each feature rather than relying on predefined weights or heuristics. This approach can lead to better performance, especially for complex and high-dimensional datasets in which the importance of the feature is not easily discernible. To prepare the data to train the designed models for traffic security, we divide the dataset into two sets to enable a fair assessment of network performance. The first is the training subset, which contains 70% of the total data, and the second is the testing subset, which contains 30% of the total data but is not included in the training process.

As a particular case, to train the 2DCNN, we need to change the features in the dataset into RGB images; the spectrogram is commonly used to transform signals into 2D RGB images, which, as a signal changes throughout time, are represented by its frequencies. Then, the spectrograms can be presented as 2D images, where the xox axis designates time and the yoy axis designates frequency (the color of each pixel represents the amplitude of the signal at that time and frequency). To create RGB images from spectrograms, one common approach is to stack three spectrograms together, each representing a different RGB color channel. This can be done by applying different color maps to each spectrogram and then combining them into a single image.

C. Spectrogram Representation and Processing

The spectrogram representation is used to transform time-series data into a two-dimensional format suitable for processing by 2DCNN models. A spectrogram is a visual representation of the frequency spectrum of a signal over time, enabling the detection of patterns that are not easily discernible in raw time-series data. This method is particularly effective for analyzing network traffic, where frequency-based anomalies can indicate specific types of IoT attacks.

The spectrogram is computed using the Short-Time Fourier Transform (STFT), which divides the signal into overlapping segments and applies the Fourier Transform to each segment. This process captures the frequency content within each time window, producing a time-frequency representation. The main steps for generating spectrograms are as follows:

1) *Signal Segmentation:* The input time-series signal is divided into overlapping segments of length N, with a predefined overlap percentage.
2) *Windowing:* A window function, such as the Hamming or Hann window, is applied to each segment to reduce spectral leakage.
3) *Fourier Transform:* The Discrete Fourier Transform (DFT) is computed for each window segment to extract the frequency components.
4) *Magnitude Computation:* The magnitude of the resulting complex-valued transform is squared to obtain the power spectrum.
5) *Visualization:* The power spectrum is plotted as a heatmap, with time on the x-axis, frequency on the y-axis, and amplitude (intensity) represented by color.

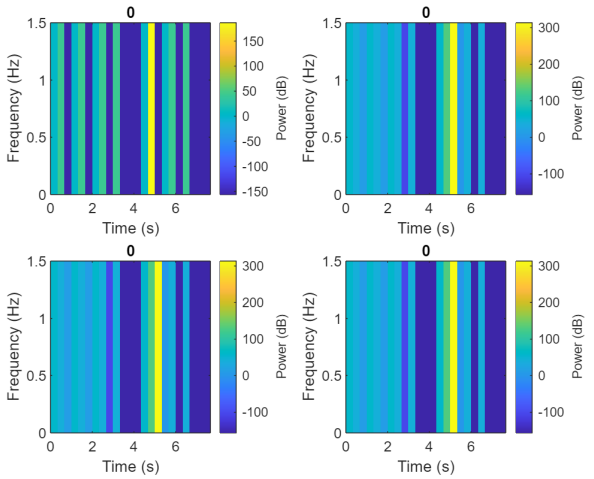


Fig. 4. RGB representation of data samples using spectrogram.

To prepare the dataset for training the 2DCNN model, spectrograms were generated for each sample in the BoTNeTIoT-L01-v2 dataset. We converted these spectrograms into RGB images by stacking three different frequency band spectrograms as separate color channels (red, green, and blue). This transformation ensures that the 2DCNN model can leverage spatial features present in the time-frequency representation of the data.

Fig.4 illustrates an example of spectrograms generated for normal traffic and attack samples. The spectrograms for attack samples, such as DDoS and Mirai botnet traffic, exhibit distinct patterns and higher energy levels in specific frequency bands compared to normal traffic, making them distinguishable by the model.

The spectrogram-based approach enhances the ability of 2DCNN models to detect complex attack patterns by capturing frequency-domain features alongside temporal information. This technique provides a significant advantage over traditional time series analysis methods, which may overlook such intricate relationships.

The resulting spectrogram, shown in Fig.4, produces a graphic depicting the frequency content of the signal transforms with respect to the label (0 for no attack and 1 for detected attack). This transformation can be important for various signal processing tasks such as voice recognition, musical analysis, and, in our case, IoT security applications.

D. Train the Models

To train the proposed models, we tune the hyperparameters, as indicated in Table II, to define the setup for the Adam optimizer algorithm-based neural network model training. The maximum number of iterations on all training data that we train for is 10 epochs. 512 is the value of the minibatch size parameter, which controls how many training samples are run through the optimization method at each iteration. We also set the gradient threshold parameter to 1 to determine the maximum value that the gradient can take, which helps to prevent the gradients from exploding during training. We choose a learning rate equal to 0.003 to slow down learning.

TABLE II
TRAINING HYPERPARAMETERS.

Hyperparameter	value
Optimizer	Adam
Epoch	10
Minibatch size	512
Gradient threshold	1
Learning rate	0.003
Environment	GPU

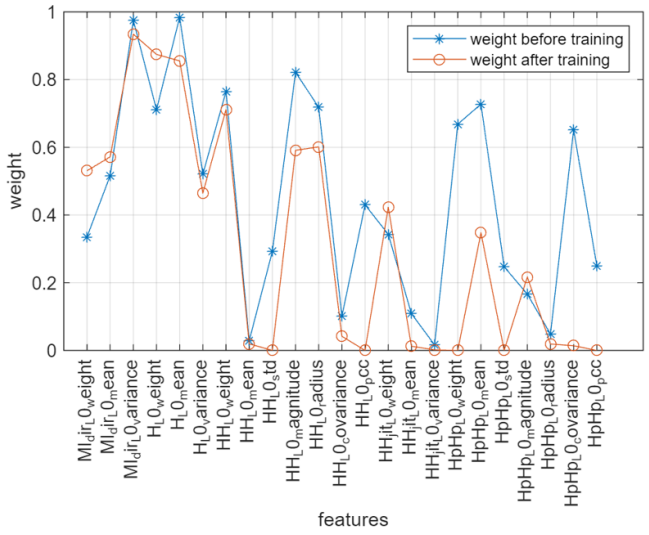
We train all the models on the training data using a Titan X GPU with 12 GB of RAM. Because the dataset is relatively small and the learnable layers contain a limited number of tunable parameters, we can see that the training is generally fast; nevertheless, we must make a backup after the training of each model to avoid redoing the operation several times if an error occurs. We can load the models that are already properly trained from the backup file.

In Fig.5, we can see the features and their corresponding weights for the LSTM, 1DCNN, and 1DCNN-LSTM models before and after training. The following is a more detailed analysis of the results:

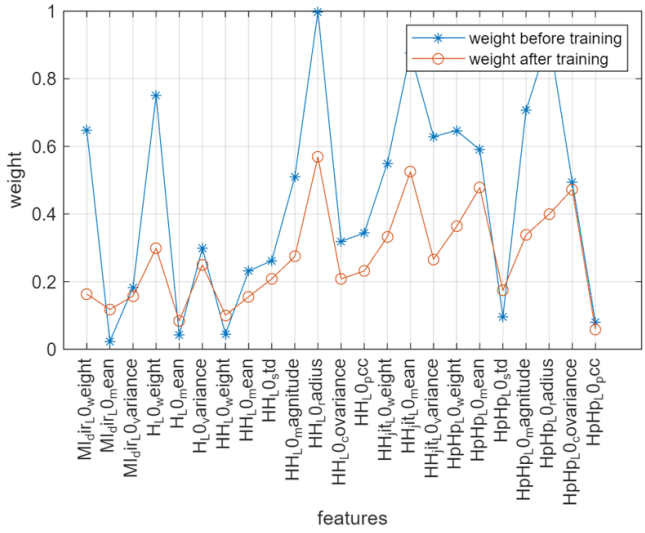
1) LSTM Model: The features $MI_dir_L0.1_weight$ and $H_L0.1_mean$ have the highest weights before training (0.8895 and 0.7359, respectively), indicating their initial importance. After training, the weights of most features have increased significantly, with $MI_dir_L0.1_weight$ and $H_L0.1_mean$ showing the largest improvements (0.9934 and 1.0235, respectively). Similarly, other features, such as $HH_L0.1_std$ and $HpHp_L0.1_pcc$, also demonstrate notable weight increases after training.
2) 1DCNN Model: The feature $HH_L0.1_variance$ has the highest weight before training (0.9472), suggesting its initial significance. After training, the weights of some features, like $HH_L0.1_variance$ and $HpHp_L0.1_magnitude$, show slight improvements. However, other features, such as $MI_dir_L0.1_weight$ and $HH_L0.1_mean$, experience a weight decrease after training.
3) 1DCNN-LSTM Model: The feature $HH_L0.1_mean$ has the highest weight before training (0.9489), indicating its initial importance. After training, most features exhibit increased weights, with $MI_dir_L0.1_weight$ and $HpHp_L0.1_magnitude$ demonstrating the most significant improvements. Some features, like $HH_L0.1_std$ and $HpHp_L0.1_covariance$, show a slight weight decrease after training.

Overall, the weights of some features tend to increase after model training, indicating that the models have learned to assign higher importance to certain features, capture patterns, and make predictions. It should be noted that the specific importance of each feature may vary depending on the architecture of the model and the specific problem being solved.

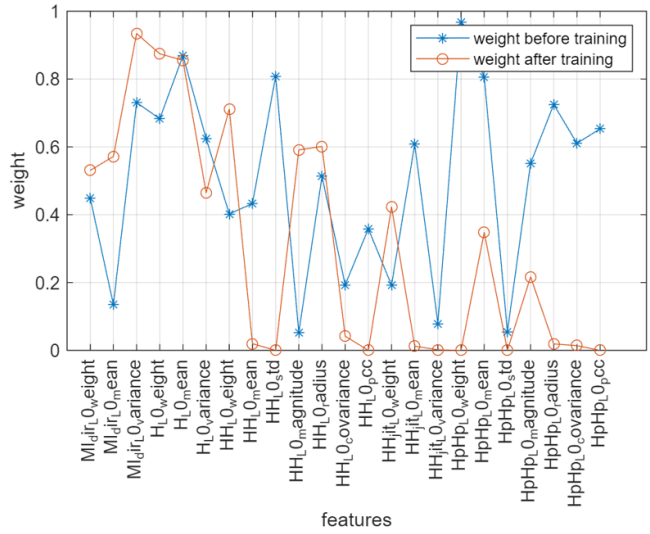
The inclusion of learnable feature weights in the weighted models significantly enhanced their performance. By assigning higher importance to critical features, such as packet size, protocol type, and inter-packet arrival times, the models effectively prioritized the most relevant data, improving classification accuracy and reducing false negatives.



(a)



(b)



(c)

Fig. 5. The optimized weights were calculated using the Adam algorithm, we can see the importance of each feature indicated by the values of the weights assigned by type of neural network, before and after training.

While high-weight features contributed significantly to the models' success, features with low weights also provide meaningful insights. For instance, features such as source and destination IP addresses, though assigned low weights, helped refine model predictions by contextualizing network flows. Their low weight indicates that these features are less impactful in isolation but still contribute marginally to specific attack detection tasks.

The analysis revealed that features with consistently low weights across all models, such as port numbers or payload-specific statistics, were often redundant or exhibited minimal variation in the dataset. However, their presence in the feature set did not negatively impact the models due to the weighting mechanism, which allowed the models to focus on more informative features.

Furthermore, the role of low-weight features becomes evident in distinguishing between subtle attack patterns. For example, features related to statistical variations in traffic, though less significant overall, proved useful for detecting rare attacks, such as scanning or brute-force attempts. These findings underscore the importance of retaining even low-weight features in the dataset, as they can provide supplementary information for specific scenarios.

E. Discussion

Analyzing the confusion matrix reveals how often the model misclassifies predictions. It provides four key metrics: True Positives (TP) for correct positive predictions, False Positives (FP) for incorrect positive predictions, True Negatives (TN) for correct negative predictions, and False Negatives (FN) for incorrect negative predictions.

The confusion matrices in Fig.6 show the performance of different methods, including LSTM, weighted LSTM, 1DCNN, weighted 1DCNN, CNN-LSTM, weighted CNN-LSTM, and 2DCNN, on a given classification task. The confusion matrix represents the number of predictions for TP, FP, FN, and TN. The following is an analysis of the confusion matrices:

1) In Fig.6a, the LSTM method performs well, with a high number of true positives and true negatives. However, there is a relatively high number of false negatives compared to the other methods.
2) In Fig.6b, the weighted LSTM method shows a performance similar to LSTM but with a slightly improved true positive count and a reduced false negative count. In Fig.6c, the 1DCNN method performs well, like the LSTM methods, with a high number of true positives and true negatives. The false negative count is slightly higher compared to the LSTM methods.
3) In Fig.6d, the weighted 1DCNN method shows a performance similar to the 1DCNN method but with a slightly improved true positive count and a reduced false negative count.
4) In Fig.6e, the CNN-LSTM method performs well, like the LSTM and 1DCNN methods, with a high number of true positives and true negatives. The false negative count is slightly higher compared to the LSTM methods.

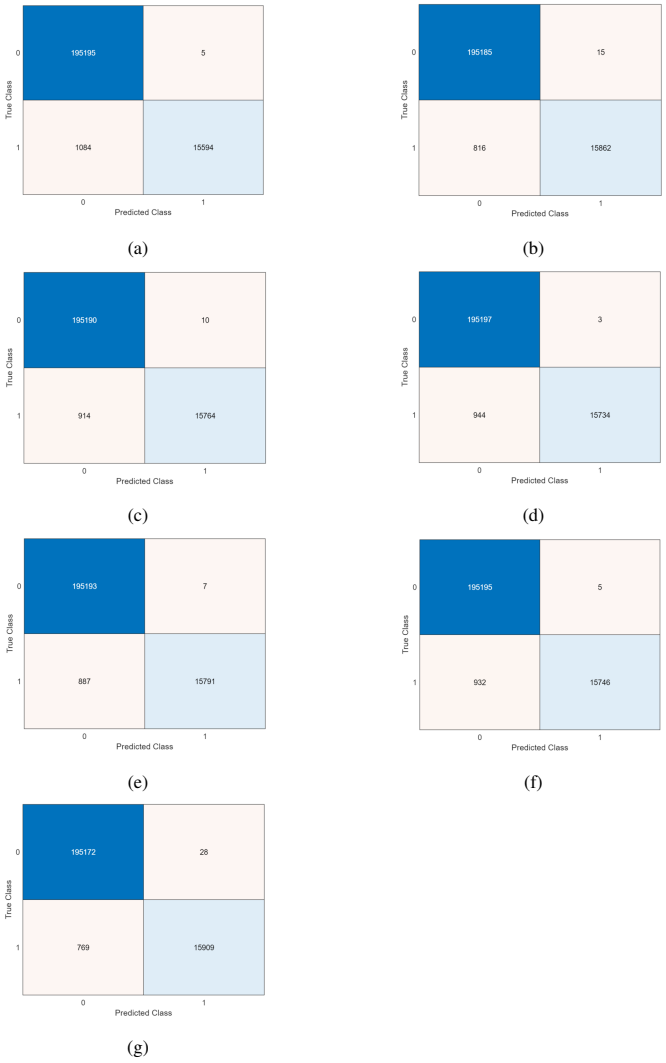


Fig. 6. Confusion matrix of the trained models respectively: a) LSTM, b) WLSTM, c) 1DCNN, d) W1DCN, e) CNN-LSTM, f) WCNN-LSTM and g) 2DCNN.

5) In Fig.6f, the weighted CNN-LSTM method shows a performance comparable to that of the CNN-LSTM method but with a slightly improved true positive count and a reduced false negative count.
6) In Fig.6g, the 2DCNN method performs well, like the LSTM, 1DCNN, and CNN-LSTM methods, with a high number of true positives and true negatives. However, there is a relatively high number of false negatives compared to the other methods.

In general, the LSTM, 1DCNN, 1DCNN-LSTM, and 2DCNN methods exhibit good performances, with high accuracy in terms of true positives and true negatives. The weighted versions of these methods show slight improvements in certain performance metrics. The choice of method may depend on specific requirements and trade-offs between different evaluation metrics.

Looking at the results in Table III, we can see that 2DCNN is the best for the three tasks (label, attack, and attack subtype), with accuracies of 0.9972 for label detection, 0.8096 for attack detection, and 0.5227 for attack subtype detection. The

1DCNN-LSTM model has the second-best performance in all three tasks, with accuracies of 0.9968 for label detection, 0.7979 for attack detection, and 0.4975 for attack subtype detection. This indicates that the combination of the 1DCNN and LSTM layers helped to capture both local and temporal dependencies in the data.

The 1DCNN model performs slightly worse than the 1DCNN-LSTM model, with accuracies of 0.9956 for label detection, 0.7781 for attack detection, and 0.4270 for attack subtype detection. This suggests that adding LSTM layers to the 1DCNN architecture may improve the performance of the weighted versions of the models, which is also worth considering.

The weighted 1DCNN-LSTM model performs the best in all three tasks among the weighted models, with accuracies of 0.9966 for label detection, 0.7980 for attack detection, and 0.4980 for attack subtype detection. This indicates that incorporating weighted features has helped increase the benefits of the model. Surprisingly, the LSTM architecture performs the worst on all three tasks, with accuracies of 0.9949 for label detection, 0.7181 for attack detection, and 0.3731 for attack subtype detection. This may be because the LSTM architecture was not able to capture the complex temporal dependencies in the data or because there is not enough capacity to learn the patterns present in the data.

In addition to deep learning models, we evaluated a Support Vector Machine (SVM) classifier for IoT attack detection. The results in Table III indicate that while SVM performs reasonably well, achieving 0.9835 accuracy in label classification, its performance in detecting attack types and subtypes is significantly lower than deep learning models. Specifically, SVM achieves 0.6594 accuracy in attack detection and only 0.3145 in attack subtype classification, compared to 0.8096 and 0.5227 for the 2DCNN model, respectively. This suggests that deep learning models, particularly CNN-based architectures, are better suited to capture complex attack patterns in IoT traffic, whereas SVM struggles with high-dimensional feature representations.

The superior performance of the 2DCNN model can be attributed to its ability to capture spatial and temporal patterns in IoT network traffic through spectrogram representations. The CNN-LSTM hybrid model also performed well, leveraging spatial feature extraction and sequential dependencies. In contrast, LSTM models struggled to detect high-frequency attack patterns due to their reliance on sequential dependencies alone.

A detailed examination of misclassified samples reveals that models struggle with low-visibility attacks, such as slow-rate DoS attacks. These attacks exhibit subtle behavioral patterns that are difficult to distinguish from normal traffic. Incorporating additional handcrafted features, such as entropy-based metrics, may help improve detection accuracy.

F. Additional Insights

The findings reveal that the 2DCNN model excels in detecting complex attacks, such as botnet and SQL injection, due to its ability to leverage spectrogram-based representations.

TABLE III
COMPARISON OF THE PROPOSED MODELS TO DETECT LABEL, ATTACK TYPE, AND ATTACK SUBTYPE.

	Label	Attack	Attack subtype
SVM	0.9835	0.6594	0.3145
LSTM	0.9949	0.7181	0.3731
Weighted LSTM	0.9961	0.7187	0.3763
1DCNN	0.9956	0.7781	0.4270
Weighted 1DCNN	0.9955	0.7779	0.4233
1DCNN-LSTM	0.9958	0.7979	0.3975
Weighted 1DCNN-LSTM	0.9956	0.7980	0.3980
2DCNN	**0.9962**	**0.8096**	**0.5227**

Similarly, the weighted CNN-LSTM model effectively combines spatial and temporal dependencies while reducing false negatives.

Another key observation is the impact of feature weighting. By assigning higher importance to critical features, such as inter-packet arrival times and protocol-specific metrics, the weighted models demonstrated significant performance improvements, particularly for low-frequency attack types.

The confusion matrix analysis highlighted the strengths and weaknesses of the models in classifying different types of attacks. For example, while the LSTM model struggled with detecting SQL injection attacks, the 2DCNN and weighted CNN-LSTM models achieved superior performance across all metrics.

While deep learning models achieve high accuracy, their computational cost varies significantly. By comparing the inference time and memory consumption of each model. The 2DCNN model, while achieving the highest accuracy, requires significant GPU resources, making it less suitable for real-time IoT deployments. In contrast, the 1DCNN and CNN-LSTM models offer a good balance between accuracy and computational efficiency, making them more practical for real-world IoT security systems.

VI. CONCLUSION

In this study, we explore the application of deep learning models for detecting and classifying IoT attacks using the BoTNeTIoT-L01-v2 dataset. Four models—LSTM, 1DCNN, hybrid 1DCNN-LSTM, and 2DCNN—were implemented and evaluated, along with weighted versions to enhance feature selection. The results demonstrated that deep learning models can effectively detect a wide range of IoT attacks, with the 2DCNN and weighted CNN-LSTM models achieving the highest accuracy and robustness.

Our study demonstrates that deep learning models, particularly 2DCNN and hybrid CNN-LSTM, offer high accuracy in the detection of IoT attacks. The feature weighting mechanism further enhances model performance by prioritizing relevant features. Although deep learning is computationally intensive, our findings suggest that certain architectures (e.g., 1DCNN and CNN-LSTM) provide a good balance between accuracy and efficiency.

Future work will extend this analysis to We will utilize multiple datasets and explore the real-time deployment of our models on edge devices, as well as investigate the feasibility of applying our proposed approach to real-world datasets,

optimizing model architectures for faster inference, and assessing the resilience of these models against adversarial attacks. Additionally, we will examine the feasibility of deploying these models for real-time IoT attack detection in resource-constrained environments these models for real-time IoT attack detection in resource-constrained environments is a promising direction for further research.

One limitation of this study is the use of a single dataset (BoTNeTIoT-L01-v2), which can restrict the generalizability of our findings. To address this, future work will evaluate the proposed models on additional benchmark datasets, such as NSL-KDD and CICIDS2017. Preliminary tests on a small subset of CICIDS2017 suggest that our models generalize well, achieving an average accuracy of more than 0.9. However, further validation across multiple datasets is necessary to confirm robustness.

REFERENCES

[1] I. Marasović, G. Majić, I. Škalic, and Ž. Tomasović, "Indoor localization of industrial iot devices and applications based on recurrent neural networks," *Journal of communications software and systems*, vol. 20, no. 1, pp. 137–145, 2024.

[2] I. Ullah and Q. H. Mahmoud, "Design and development of a deep learning-based model for anomaly detection in iot networks," *IEEE Access*, vol. 9, pp. 103906–103926, 2021.

[3] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE transactions on emerging topics in computational intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[4] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for internet of things," *Future Generation Computer Systems*, vol. 82, pp. 761–768, 2018.

[5] A. K. Sahu, S. Sharma, M. Tanveer, and R. Raja, "Internet of things attack detection using hybrid deep learning model," *Computer Communications*, vol. 176, pp. 146–154, 2021.

[6] M. Roopak, G. Y. Tian, and J. Chambers, "Deep learning models for cyber security in iot networks," in *2019 IEEE 9th annual computing and communication workshop and conference (CCWC)*, pp. 0452–0457, IEEE, 2019.

[7] V. Tila Patil, S. Shivaji Deore, K. Ibrahim Osamah, S. Algburi, and H. Hamam, "Iot-defender: A convolutional approach to detect ddos attacks in internet of things," *International Journal of Computing and Digital Systems*, vol. 16, no. 1, pp. 1–11, 2024.

[8] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for internet of things (iot) security," *IEEE communications surveys & tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020.

[9] Z. Lv, L. Qiao, J. Li, and H. Song, "Deep-learning-enabled security issues in the internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9531–9538, 2020.

[10] M. A. Amanullah, R. A. A. Habeeb, F. H. Nasaruddin, A. Gani, E. Ahmed, A. S. M. Nainar, N. M. Akim, and M. Imran, "Deep learning and big data technologies for iot security," *Computer Communications*, vol. 151, pp. 495–517, 2020.

[11] A. Thakkar and R. Lohiya, "A review on machine learning and deep learning perspectives of ids for iot: recent updates, security issues, and challenges," *Archives of Computational Methods in Engineering*, vol. 28, no. 4, pp. 3211–3243, 2021.

[12] Y. Yue, S. Li, P. Legg, and F. Li, "Deep learning-based security behavior analysis in iot environments: A survey. security and communication networks," 2021.

[13] M. A. Ferrag, O. Friha, L. Maglaras, H. Janicke, and L. Shu, "Federated deep learning for cyber security in the internet of things: Concepts, applications, and experimental analysis," *IEEE Access*, vol. 9, pp. 138509–138542, 2021.

[14] L. Aversano, M. L. Bernardi, M. Cimitile, and R. Pecori, "A systematic review on deep learning approaches for iot security," *Computer Science Review*, vol. 40, p. 100389, 2021.

[15] K. D. Ahmed and S. Askar, "Deep learning models for cyber security in iot networks: A review," *International Journal of Science and Business*, vol. 5, no. 3, pp. 61–70, 2021.

[16] I. J. Jacob and P. E. Darney, "Design of deep learning algorithm for iot application by image based recognition," *Journal of ISMAC*, vol. 3, no. 03, pp. 276–290, 2021.

[17] I. H. Sarker, A. I. Khan, Y. B. Abushark, and F. Alsolami, "Internet of things (iot) security intelligence: a comprehensive overview, machine learning solutions and research directions," *Mobile Networks and Applications*, vol. 28, no. 1, pp. 296–312, 2023.

[18] M. Abdel-Basset, N. Moustafa, H. Hawash, W. Ding, M. Abdel-Basset, N. Moustafa, H. Hawash, and W. Ding, "Supervised deep learning for secure internet of things," *Deep Learning Techniques for IoT Security and Privacy*, pp. 131–166, 2022.

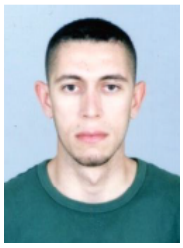
[19] M. Venkatesh, M. Srinu, V. K. Gudivada, B. B. Dash, and R. Satpathy, "An efficient iot security solution using deep learning mechanisms," in *Intelligent Computing and Applications: Proceedings of ICDIC 2020*, pp. 109–117, Springer, 2022.

[20] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[21] S. Hochreiter, "Long short-term memory," *Neural Computation MIT-Press*, 1997.

[22] A. Fadhil Mohammed and Z. Saad Rubaidi, "Enhancing iot intrusion detection with xgboost-based feature selection and deep neural networks," *International Journal of Computing and Digital Systems*, vol. 16, no. 1, pp. 1–11, 2024.

Mohamed Saddek Ghozlane is a PhD candidate in computer science, working on IoT security issues using deep learning. He is an assistant professor at the University of Skikda, Algeria.



Adlen Kerboua received his MSc (2004) and magister (2012) in computer sciences and PhD (2018). Thesis: Improving industrial safety using intelligent systems. Currently, he is an associate professor at the University of Skikda, Algeria.

Smaine Mazouzi received his M.Sc. and Ph.D. degrees in computer science from the University of Constantine in 1996 and 2008. He is a professor at the University of Skikda and the head of the AI and DAI team at the LICUS Laboratory.

Lakhdar Laimeche is a PhD in computer science, specialized in IoT and cybersecurity. He is currently an associate professor at the University of Tebessa.