IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2022.Doi Number

SkipGateNet: A Lightweight CNN-LSTM Hybrid Model with Learnable Skip Connections for Efficient Botnet Attack Detection in IoT

Mohammed S. Alshehri[1], Jawad Ahmad[2, *], Sultan Almakdi[1], Mimonah Al Qathrady[3], Yazeed Yasin Ghadi[4], and William J. Buchanan [2]

[1] Department of Computer Science, College of Computer Science and Information Systems, Najran University, Najran 61441 Saudi Arabia
[2] School of Computing, Engineering and the Built Environment, Edinburgh Napier University, Edinburgh EH10 5DT, UK
[3] Department of Information Systems, College of Computer Science and Information Systems, Najran University, Najran 61441 Saudi Arabia
[4] Department of Computer Science, Al Ain University, Abu Dhabi 112612, United Arab Emirates

Corresponding author: Jawad Ahmad (e-mail: j.ahmad@napier.ac.uk).

“This work was supported by the Deanship of Scientific Research at Najran University under the General Research Funding Program Grant NU/DRP/SERC/12/46.”

ABSTRACT The rise of Internet of Things (IoT) has led to increased security risks, particularly from botnet attacks that exploit IoT device vulnerabilities. This situation necessitates effective Intrusion Detection Systems (IDS), that are accurate, lightweight, and fast (having less inference time), designed particularly to detect botnet attacks in resource constrained IoT devices. This paper proposes SkipGateNet, a novel deep learning model designed for detecting Mirai and Bashlite botnet attacks in resource constrained IoT and fog computing environments. SkipGateNet is a lightweight, fast model combining 1D-Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) layers. The novelty of this model lies in the integration of ‘Learnable Skip Connections’. These connections feature gating mechanisms that enhance detection by focusing on relevant features and ignoring irrelevant ones. They add adaptability to the architecture, performing feature selection and propagating only essential features to deeper layers. Tested on the N-BaIoT dataset, SkipGateNet efficiently detects ten types of botnet attacks, with a remarkable test accuracy of 99.91%. It is also compact (2596.87 KB) and demonstrates a quick inference time of 8.0 milliseconds, suitable for real-time implementation in resource-limited settings. While evaluating its performance, parameters like precision, recall, accuracy, and F1 score were considered, along with statistical reliability measures like Cohen’s Kappa Coefficient and Matthews Correlation Coefficient. These highlight its reliability and effectiveness in IoT security challenges. The paper also compares SkipGateNet to existing models and four other deep learning architectures, including two sequential CNN architectures, a simple CNN+LSTM architecture, and a CNN+LSTM with standard skip connections. SkipGateNet surpasses all in accuracy and inference time, demonstrating its superiority in addressing IoT security issues.

INDEX TERMS Botnets, Botnet attacks, Bashlite, Intrusion Detection, Mirai.

I. INTRODUCTION

The Internet of Things (IoT) is an emerging technology that allows automated data sensing, collection, and transmission. It uses interconnected devices ranging from computers, sensors, vehicles, phones, and home appliances and supports various applications, such as intelligent transportation, smart grids, smart homes, smart cities, and smart agriculture [1, 2]. This widespread adoption of IoT devices has increased susceptibility to various security threats, particularly botnet attacks that exploit IoT device vulnerabilities [3]. Recent reports have revealed that 41% of attacks exploit IoT device vulnerabilities due to 98% of the IoT device traffic being unencrypted [4]. The botnets, such as BASHLITE and Mirai, pose significant threats to IoT networks due to their capacity to compromise many devices and the variety of attacks they employ [5]. These security issues become even more critical, particularly in the context of fog computing – a decentralized processing and storage paradigm that enables data handling closer to the network edge. Fog computing's unique constraints and operational context demand specialized intrusion detection solutions. These solutions need to address the limited computational resources, low-latency requirements, and the dynamic nature of fog-based IoT networks, thus contributing to the body of knowledge in this area [6, 7]. The Mirai botnet attack emerged in 2016 and compromised various vulnerable IoT devices, including cameras and routers, to conduct large-scale distributed denial of service (DDoS) attacks such as Ack flooding, Syn flooding, UDP flooding, UDP plain flooding, etc. [8]. Similarly, the Bashlite botnet (also known by other names such as Gafgyt, Q-Bot, Torlus, Lizard-Stresser, and Lizkebab) targets IoT devices and has been responsible for launching DDoS attacks, spreading malware, and exploiting device vulnerabilities through certain types of attacks including Scan, Junk, UDP, TCP, and COMBO [9]. Therefore, efficient and effective intrusion detection systems (IDS) are required to counteract the threat of botnet attacks, especially for edge devices and fog computing environments.

IDS plays a crucial role in detecting and mitigating cyber threats. The anomaly-based IDS, in particular, are designed to identify unusual patterns in network traffic, which may indicate the presence of an attack. As the use of IoT has been increasing recently, the need for effective anomaly-based IDS has become indispensable [10]. While the traditional machine learning models have widely been used for intrusion detection in IoT networks [11], they face certain challenges, e.g., limited scalability, inadequate performance in dealing with complex and evolving attack patterns, and difficulty in handling high-dimensional data, [12], etc. Therefore, there is a pressing need to develop new and efficient deep learning models that can be used in IDS, particularly designed for detecting botnet attacks in IoT devices. In recent years, deep learning models have emerged as a promising alternative, demonstrating superior performance in handling large-scale, high-dimensional data and capturing complex patterns (features) in the data [13]. Most of the existing deep learning-based solutions for intrusion detection are not lightweight and pose latency issues, making them unsuitable for implementation in edge devices in IoT networks or fog computing. However, deep learning models, if designed specifically for the type of attacks or keeping in view the challenges in the IoT networks, can perform efficiently and adequately well.

To address the aforementioned challenges, this paper presents a lightweight and efficient deep-learning model tailored specifically to detect the Mirai and Bashlite botnet attacks. The proposed model is based on a combination and tailored arrangement of 1D-Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) layers. The novelty of the proposed model lies in using 'Learnable Skip Connections'. Traditional skip connections allow information to bypass one or more layers, and flow directly from one part of the network to another. Unlike standard skip connections, which pass the information without any modulation, learnable skip connections incorporate gating mechanisms to control the flow of information dynamically. Essentially, these mechanisms learn to regulate what information is useful to propagate forward and what can be omitted. The learnable skip connections add a level of adaptability to our proposed architecture. These connections perform a kind of feature selection, determining which features are important enough to be directly propagated to deeper layers. As a result, the network becomes better at focusing on the most relevant patterns in the data, leading to improved model performance. Moreover, the learnable skip connections contribute to the overall compactness of the SkipGateNet, maintaining the model's lightweight characteristics. The gating mechanisms, despite their adaptive capabilities, don't introduce an extensive number of parameters into the network, keeping the computational costs manageable. This is particularly advantageous for IoT settings, where computational resources are often limited, such as real-time servers and processors that work as edge devices or Fog Nodes.

The main contributions of this paper are:

1) This paper introduces a novel convolutional and recurrent neural network architecture, SkipGateNet, designed specifically for IoT botnet attack detection. A key aspect of this architecture is the use of 'Learnable Skip Connections'. These connections are capable of dynamically controlling the flow of information across the network, enabling the model to focus on salient features and ignore irrelevant ones, thus enhancing its detection capabilities.
2) Gating mechanisms have been integrated into the learnable skip connection blocks. Each learnable skip connection employs a 1D convolution layer followed by a sigmoid activation function to create a gate. Integration of gating mechanisms enables adaptive feature selection in the proposed model. This process allows the model to pay attention to more informative features, thereby mitigating the impact of noise or irrelevant features that are prevalent in IoT data streams.
3) SkipGateNet is compact, light and highly efficient having a size of only 2596.87 KBs, a total of only 683,083 parameters and a fast inference time of only 8 milliseconds. This makes SkipGateNet an efficient solution for botnet attack detection in resource-constrained IoT networks.
4) Four deep learning architectures have been implemented, trained and tested on the same dataset to compare their accuracies and inference times with the proposed SkipGateNet model. Experimented models include two sequential CNN architectures with dense

layers, a simple CNN+LSTM architecture with dense layers, and a CNN+LSTM architecture with standard skip connections. The proposed SkipGateNet outperformed all these architectures exhibiting an accuracy of 99.91% and 8.0 milliseconds inference time.

The rest of the paper is organized as follows. Section 1 introduces the problem domain and the need for the development of new and improved deep learning models for botnet detection. Section 2 discusses the related work and the existing solutions available for botnet detection. In Section 3, the approach for detecting botnet attacks is outlined. Section 4 is devoted to the dataset used for the study, with a detailed explanation of the data pre-processing and splitting techniques. The main contribution of this paper is given in Section 5, where the proposed model with learnable skip connections—SkipGateNet is presented in detail. Section 6 is dedicated to the performance evaluation and results. In Section 7, we present a comparative analysis of our proposed model against deep learning models in the literature and self-implemented architectures, as well as against traditional machine learning models. Lastly, Section 8 concludes the paper by summarizing the key findings and contributions of the paper.

II. RELATED WORK

Recently, various machine learning (ML) and deep learning (DL) algorithms have been utilized for intrusion detection applications. However, most of them did not focus on the inference time and size of the utilized models. Speaking of employing machine learning techniques in particular, a handful of IDS can be found in which ML techniques have been employed; for instance, the authors in [14] use ML classification to secure IoT devices against attacks like DoS. The authors utilize three datasets, i.e., UNSW-NB15, NSL-KDD, and CIDDS-001, to benchmark the proposed classifiers. Similarly, the authors in [15] have proposed semi-distributed and distributed methods to address the limitations of centralized IDS for resource-constrained devices, achieving comparable detection accuracy to superior centralized IDS with inherent trade-offs between accuracy and building time performance. Although these methods seem suitable for resource-constrained devices, such as those used in fog computing, their inference time has not been discussed. Furthermore, a cross-layer-based IDS has been proposed for detecting malicious activities in mobile ad-hoc networks (MANETs) and other IoT networks in [16]. The authors claimed 98% and 90 % detection rates for high and low power velocity scenarios, respectively. Moreover, the study in [17] discusses shallow and deep machine learning-based IDS in IoT environments. It evaluated their performance using five benchmark datasets (NSL-KDD, IoTDevNet, DS2OS, IoTID20, and IoT Botnet dataset). The authors claim that deep ML IDS works better than shallow ML IDS, especially in the case of IoT attack detection. Besides, in [18], six ML models are utilized to compare and evaluate the performance of three different Feature Extractors (FE). The evaluation has been carried out on three benchmark datasets (UNSW-NB15, ToN-IoT, and CSE-CIC-IDS2018). The authors concluded that the choice of datasets significantly alters the performance of the applied techniques, highlighting the need for a universal benchmark feature set. Although the paper analyzed the performance of the feature extractors, it does not mention their inference time, model size, or suitability for resource constrained devices. In addition, an IDS that integrates the MapReduce framework with machine learning (ML) techniques is presented in [19]. Utilizing a dataset with multiple network attacks, the model exhibited a detection accuracy of 95.7% validation accuracy implying that combining MapReduce and ML is beneficial in intrusion detection. Regarding machine learning-based IDS for Fog computing, the authors in [20] propose a novel distributed IDS using fog computing to detect DDoS attacks in blockchain-enabled IoT networks. The model trains Random Forest (RF) and an optimized gradient tree boosting system (XGBoost) on distributed fog nodes, with RF outperforming XGBoost in certain scenarios. While some proposed techniques are designed for IoT networks and resource-constrained devices, most do not explicitly discuss their inference time, model size, and suitability in intrusion detection systems intended for fog nodes or edge devices.

In addition to traditional machine learning techniques, various deep learning techniques have also been proposed for intrusion detection systems. For example, in [21], the authors introduce a DL artificial neural network (ANN) model for detecting botnet attacks. The model is trained and evaluated on the CTU-13 dataset and can efficiently identify botnets, achieving 99.6% accuracy. Similarly, the authors in [22] presented a hybrid IDS for the Internet of Medical Things (IoMT). This system combines CNN and LSTM networks and exhibits an average accuracy of 97.63%. Some hybrid deep learning models have also been presented, such as the authors in [23] an IDS utilizing a hybrid approach of machine learning (ML) and deep learning (DL) techniques. The model uses SMOTE for data balancing and XGBoost for feature selection, aiming to handle large and imbalanced datasets efficiently. The authors tested the model on two datasets: KDDCUP'99 and CIC-MalMem-2022, achieving exceptional accuracy of 99.99% and 100% respectively, with no overfitting issues.

Regarding intrusion detection systems designed for botnets, the authors in [24] leveraged latent representations of network traffic features from CNNs to detect and classify botnet attacks. Moreover, the work in [25] presented an ML algorithm utilizing explainable AI. It used the IRA-CIC-DoHBrw-2020 dataset. The authors claimed a high precision and F1 score of 99.91% and a recall of 99.92%. In addition, the authors in [26] studied smart home security attack properties and suggested effective intrusion prevention mechanisms using various ML models and feature sets. Besides, the authors in [27] have developed a malware detection system, the FedMalDE, as named by the authors. Their framework is based on federated learning and knowledge transfer techniques. They employed a subgraph aggregated capsule network (SACN) for capturing malicious behaviors. While these papers showcase various deep learning techniques for intrusion detection systems, the majority do not explicitly discuss the inference time and model size.

Moreover, researchers have also utilized various deep learning techniques for the classification of various attacks, e.g., in [28], the authors employed a CNN-LSTM algorithm on the N-BaIoT dataset, achieving an F1-score, precision and recall of 0.88, 93.04% and 91.91%, respectively, with an overall accuracy of 90.88%. Similarly, in [29], a deep belief network (DBN) algorithm was applied to the N-BaIoT dataset, yielding a higher F1-score of 0.92. The precision and recall were reported as 98.27% and 92.82%, respectively, resulting in an accuracy of 95.60%. Another study [30] utilized the CNN-LSTM algorithm on the N-BaIoT dataset, achieving an F1-score of 0.93. The precision and recall were reported as 93.48% and 93.675%, respectively, resulting in an overall accuracy of 94.30%. Moreover, the authors in [31] explored the use of autoencoders on the N-BaIoT dataset, but specific performance metrics such as F1-score, precision, and recall were not provided. However, the accuracy was reported as 90.2%. Lastly, in reference [32], both autoencoders and DNN algorithms were utilized on the N-BaIoT dataset. The F1-score achieved was 0.80, with a precision of 99% and a recall of 66%. The overall accuracy was reported as 97.21%. These results demonstrate the performance of different algorithms on the N-BaIoT dataset, highlighting their effectiveness in detecting and classifying IoT network traffic. However, it is important to note that the design and arrangement of deep learning layers, dataset preprocessing techniques, and other factors like hyperparameters can influence the results obtained in each study. Therefore, further investigation and comparative analysis are required to determine the most suitable algorithm for the N-BaIoT dataset for botnet detection.

The recent literature review reveals that while various deep learning-based IDS have been proposed for detecting botnet attacks in IoT networks, they present limitations in size, inference time, and suitability for deployment in resource-constrained devices, such as edge IoT devices that work as fog nodes. This research gap highlights the need for lightweight and efficient deep learning-based intrusion detection systems, which can easily be deployed in real-time scenarios, especially for IoT devices in fog computing environments.

III. BOTNET ATTACK DETECTION APPROACH

The proposed SkipGateNet model is intended to be deployed in anomaly-based IDS for resource constrained devices in IOT and fog computing. A general overview of such type of IDS is depicted in Fig. 1. Such an IDS comprises a series of components, including a fog node, traffic capture, data filtering, feature selection using a deep learning model, a warning logger, and alert notification. A Fog Node is a decentralized computing infrastructure that extends cloud computing capabilities closer to the edge of the network. In the IDS framework given in Fig. 1, the Fog Node acts as the primary point for capturing, filtering, and analyzing network traffic, thereby improving response time and reducing the load on the central server. The IDS uses network sensors or agents to collect and store the raw data packets, which are then forwarded to the Data Filtering component. This process helps gather the necessary information to detect malicious activities and identify potential security threats. By capturing, filtering, and analyzing network traffic in real time, the system can identify and respond to potential threats before they cause significant damage. Such intrusion detection systems leverage tailored deep learning models and preferably a distributed computing approach to provide a robust and efficient solution for detecting network intrusions. In addition to processing data efficiently, this approach provides real-time analysis while reducing latency and bandwidth consumption.

Detecting botnet attacks is a critical challenge in maintaining the security and integrity of modern IoT networks.

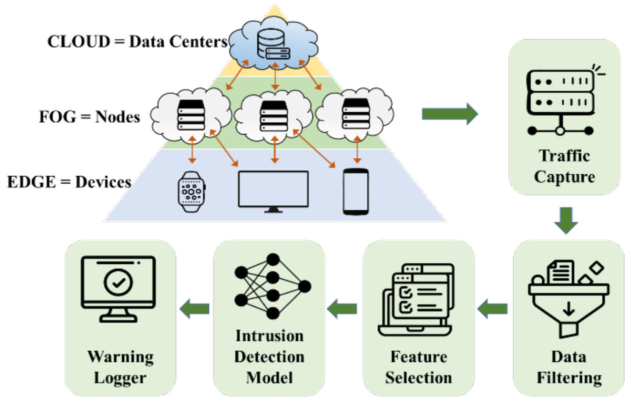


FIGURE 1. A general depiction of an IDS in a Fog Computing Network.

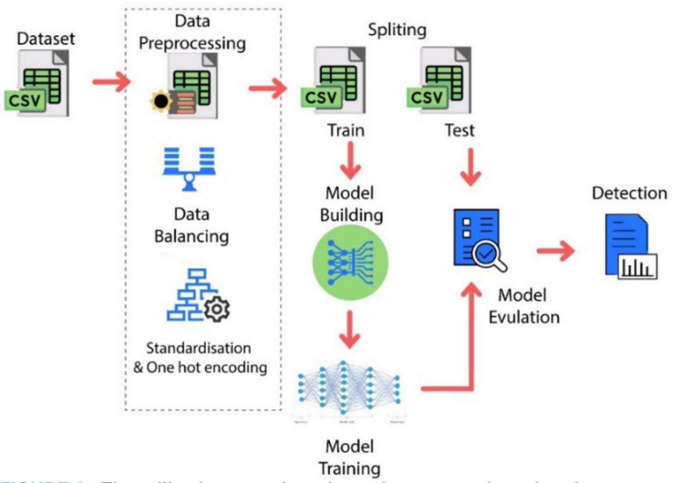


FIGURE 2. The utilized approach to detect botnet attacks using deep learning.

The deep learning approach utilized in this paper to identify botnet activities is given in Fig. 2. The process involves key steps, such as dataset formulation, data pre-processing, data splitting, building a tailored deep learning model, model training, evaluation, and detection. The first step involves collecting a comprehensive dataset containing normal and botnet traffic data. N-BaIoT dataset has been utilized for this purpose. The next step is data pre-processing. It includes data balancing to ensure equal representation of normal and botnet traffic, standardization to scale the input features to a similar range, and one hot encoding to convert categorical features into a binary format. These techniques help improve the model's learning capability and reduce the risk of overfitting. After pre-processing, the dataset is split into training and testing sets. In this paper, an 80-20 split is used, where the larger portion is reserved for training and validation of the model, and the smaller portion is used to evaluate the model's performance on unseen data.

The deep learning model is then trained on the prepared training dataset. Keras Tuner has been utilized in this paper for the optimization of hyper-parameters. During this phase, the model learns to identify patterns and features that distinguish botnet traffic from normal traffic. Once the training is complete, the model is evaluated on the test dataset using various performance metrics, such as accuracy, precision, recall, and F1-score. In addition, the reliability parameters, i.e., Cohen's Kappa coefficient and Mathew's Correlation Coefficient, have also been calculated to confirm the reliability of the proposed model.

IV. DATASET

In this paper, the N-BaIoT [33] dataset has been utilized. The N-BaIoT dataset is a comprehensive collection of network traffic data specifically designed and collected for detecting botnet attacks targeting IoT devices [5]. The dataset consists of benign and malicious traffic data captured from various types of IoT devices, including cameras, routers, and smart home appliances. It comprises a total of 7,062,606 instances. Each instance represents a network traffic snapshot, captured, and processed to facilitate the identification of both benign and malicious activities in IoT networks. The dataset includes 115 distinct features extracted from network traffic data. These features are derived from several temporal windows, capturing various aspects of the traffic, such as originating IP, source MAC and IP address, communication channels, and TCP/UDP sockets. The features are calculated over five-time windows (100ms, 500ms, 1.5sec, 10sec, and 1min), and they are designed to be computed quickly and incrementally, supporting real-time anomaly detection. The attributes extracted from the packet stream cover statistical measures like weight, mean, standard deviation, radius, magnitude, covariance, and Pearson correlation coefficient, among others. These attributes are grouped under different headers like stream aggregation (H, HH, HpHp, HH_jit) and timeframe (with varying decay factors such as L5, L3, L1, etc.).

The N-BaIoT dataset features authentic traffic data from nine commercial IoT devices infected with Mirai and BASHLITE malware, incorporating ten different IoT attacks (five types of attacks from each botnet). The Mirai attacks involved automatic network scanning for vulnerable devices (Scan), Ack flooding (Ack), Syn flooding (Syn), UDP flooding (UDP), and a limited option UDP flooding optimized for higher packets per second (UDPplain). On the other hand, the BASHLITE attacks include network scanning for vulnerable devices (Scan), the transmission of spam data (Junk), UDP flooding (UDP), TCP flooding (TCP), and a combination of spam data transmission and establishing a connection to a specified IP address and port (COMBO). The class-wise detail of the complete dataset is given in Table I.

A. DATASET PREPARATION AND PREPROCESSING

To effectively manage the extensive size of the dataset and to utilize the complete dataset in both the training and testing phases of the proposed model, the dataset is divided into ten equal-sized subsets. There is no overlapping in these subsets, i.e., no samples from a subset are repeated in any other subset. Details of the data subsets for all classes are given in Table II. An Incremental learning strategy has been employed for the subset-wise training of the model. This approach allows the model to sequentially learn from the subsets of data, integrating new information while retaining previously acquired knowledge. One of the major challenges in sequential incremental learning is catastrophic forgetting. Catastrophic forgetting occurs in neural networks when they learn new tasks sequentially; the training on the new data can lead to the loss of previously learned information. To resolve this, elastic weight consolidation (EWC) technique has been utilized. EWC selectively slows down the learning on certain weights based on how important they are to previously learned data. This technique helped the proposed model to be trained on each subset one after the other, while minimizing the forgetting of what it learned from the previous subsets.

TABLE I
DETAILS OF THE DATASET

Sr.	Classes	No. of Samples
1	mirai_udp	1,229,999
2	gafgyt_udp	946,366
3	gafgyt_tcp	859,850
4	mirai_syn	733,299
5	mirai_ack	643,821
6	benign	555,932
7	mirai_scan	537,979
8	mirai_udpplain	523,304
9	gafgyt_combo	515,156
10	gafgyt_junk	261,789
11	gafgyt_scan	255,111
	Total Samples	**7,062,606**

TABLE II
SUBSETS OF CLASSES FOR THE TRAINING PURPOSES

Sr.	Classes	Subset 1	Subset 2	Subset 3	Subset 4	Subset 5	Subset 6	Subset 7	Subset 8	Subset 9	Subset 10	Total
1	mirai_udp	123000	123000	123000	123000	123000	123000	123000	123000	123000	122999	1,229,999
2	gafgyt_udp	94637	94637	94637	94637	94637	94637	94636	94636	94636	94636	946,366
3	gafgyt_tcp	85985	85985	85985	85985	85985	85985	85985	85985	85985	85985	859,850
4	mirai_syn	73330	73330	73330	73330	73330	73330	73330	73330	73330	73329	733,299
5	mirai_ack	64383	64382	64382	64382	64382	64382	64382	64382	64382	64382	643,821
6	benign	55594	55594	55593	55593	55593	55593	55593	55593	55593	55593	555,932
7	mirai_scan	53798	53798	53798	53798	53798	53798	53798	53798	53798	53797	537,979
8	mirai_udpplain	52331	52331	52331	52331	52330	52330	52330	52330	52330	52330	523,304
9	gafgyt_combo	51516	51516	51516	51516	51516	51516	51515	51515	51515	51515	515,156
10	gafgyt_junk	26179	26179	26179	26179	26179	26179	26179	26179	26179	26178	261,789
11	gafgyt_scan	25512	25511	25511	25511	25511	25511	25511	25511	25511	25511	255,111
	Total Samples											**7,062,606**

In the context of the N-BaIoT dataset utilized in this paper, EWC is implemented by first training the model on the initial subset and calculating a loss function that represents the model's performance on this subset. Following this, for each subsequent subset, a new loss function is computed, reflecting the model's performance on the new data. The crucial aspect of EWC is in its penalty term, which is added to the loss function. This term identifies crucial parameters (weights) in the neural network that are significant for the performance on the previous subset. By adding a penalty for significant changes to these weights, EWC effectively retains the model's performance on earlier subsets while allowing it to learn from new data.

B. DATA SPLITTING

To test the performance of the proposed model, each subset was divided into three sets: training, validation, and testing. An 80-20 split ratio was utilized for the training-testing set, i.e., allocating 80% of the data for training and 20% for testing. The training set was further divided using an 80-20 split ratio, with 80% of the data dedicated to training and 20% to validation. This second split is beneficial for evaluating the performance of the deep learning model during training by measuring its accuracy on the validation set. This data splitting ensured that the deep learning model was trained on a distinct set of data and tested on a non-overlapping dataset, i.e., this testing data was not included in the training and validation set.

V. THE PROPOSED MODEL WITH LEARNABLE SKIP CONNECTIONS

This paper presents SkipGateNet, a deep learning model based on the combination and tailored arrangement of 1D-CNN and LSTM layers having 'Learnable Skip Connections'. The novelty of the proposed model lies in using learnable skip connections having gating mechanisms to control the flow of information dynamically. The architecture of the proposed model is given in Fig. 3. Before digging into the proposed model's architectural details, and it is essential to first describe the details of the utilized layers, i.e., 1D-CNN, LSTM, and the Learnable Skip Connections. The following subsections explain the utilization of layers in the proposed model.

A. 1D-CNN

A 1D CNN is a convolutional neural network that handles one-dimensional input data. It alternates between convolution layers and pooling layers to extract features. These layers are explained as follows.

Convolutional layers

In a 1D CNN layer, as depicted in Fig. 4, each convolutional feature $X_n (n = 1,2,3,..,N)$ is linked with multiple input features through a local weight matrix W_n having dimensions $P \times Q$. Here, P refers to the number of filters, Q represents the length of the convolutional kernel (or filter). Each filter (of length Q) convolves across the input data to produce a feature map, and there are P number of such feature maps due to P filters. A single unit of a convolutional feature is mathematically expressed as follows [34]:

$$x_{n,k} = \alpha \left(\sum_{p=1}^{P} \sum_{q=1}^{Q} i_{p,q+k-1} w_{p,n,q} + w_{0,n} \right) \tag{2}$$

Where

$x_{n,k}$ represents the kth unit of the feature X_n.

α represents the activation function.

$i_{p,q}$ represents the kth unit of the input feature I_p.

$w_{p,n,q}$ represents the unit q of the weight matrix $W_{p,n}$.

Similarly, the convolution operation or linking of the convolutional feature to the input features via the weight matrix can be expressed mathematically as (3).

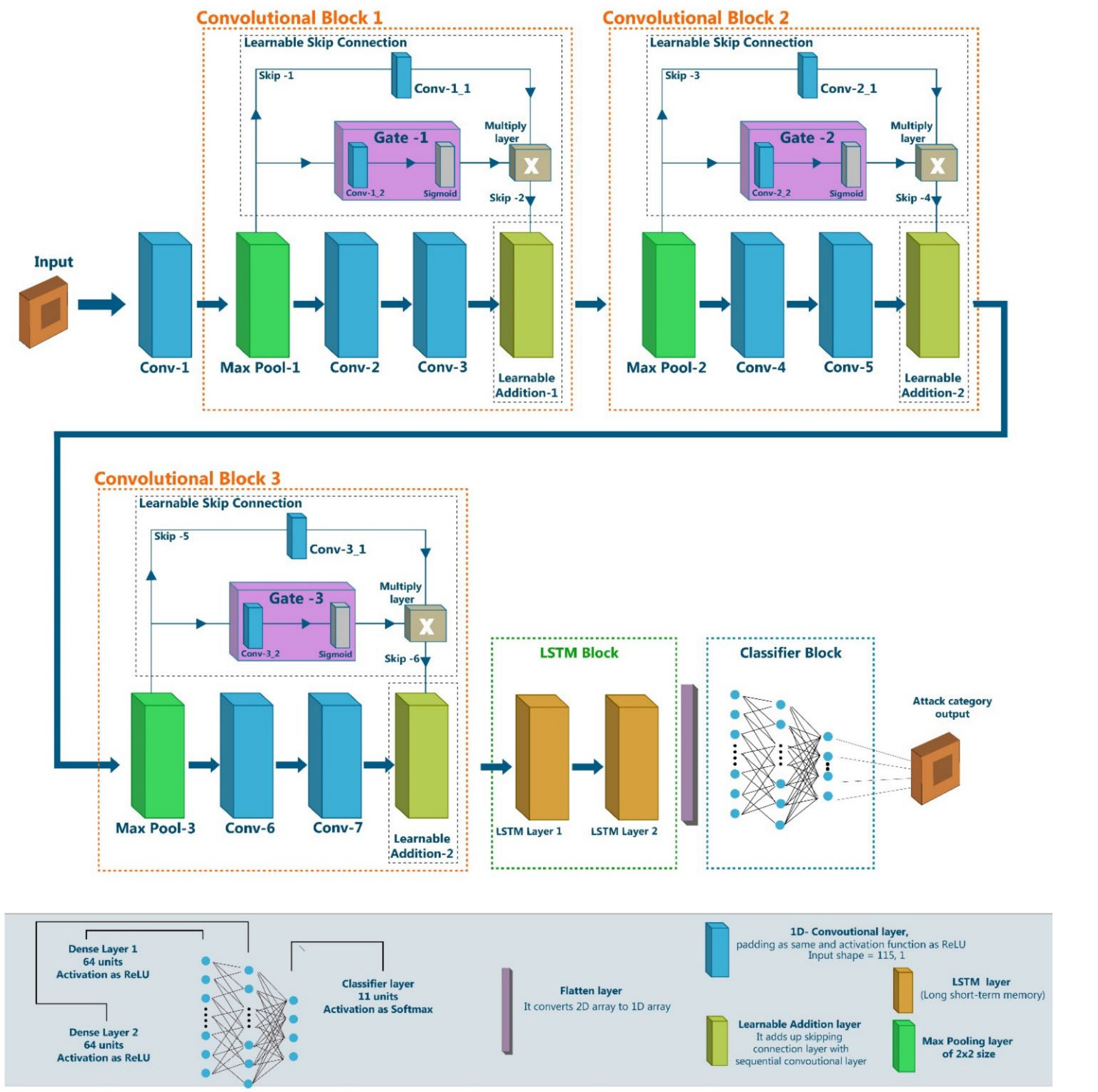


FIGURE 3. Architecture of the proposed model.

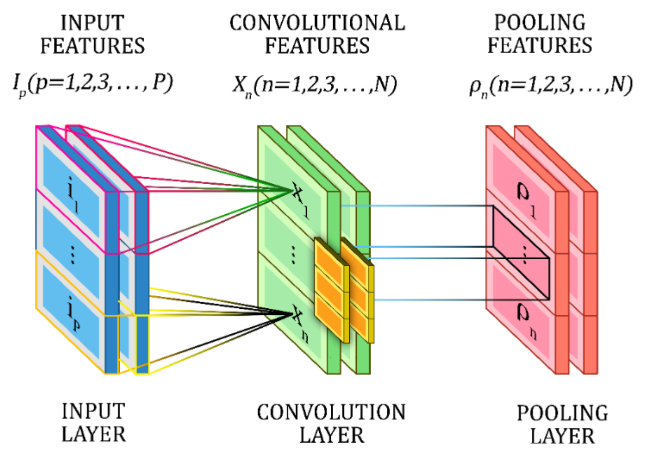


FIGURE 4. **1D CNN depicting an input layer, a convolution layer, and a pooling layer**

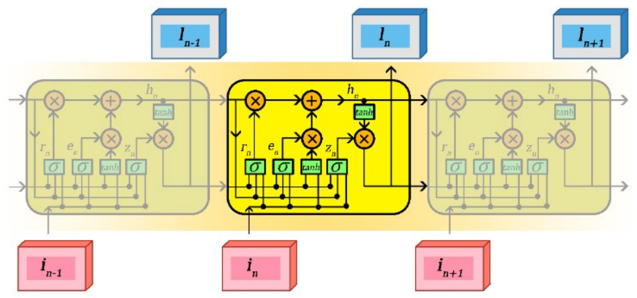


FIGURE 5. **Structure of an LSTM cell.**

$$X_n = \alpha\left(\sum_{p=1}^{P} I_p * W_{pn}\right) \; (n = 1,2,..,N) \tag{3}$$

Where
I_p represents the pth input feature.
$*$ represents the convolution operator.

1) POOLING LAYERS

The function of a pooling layer in a 1D CNN is to reduce the dimensionality of the input features while preserving the most important information. The pooling operation helps capture the essential patterns in the data, which aids in identifying potential intrusions. Also, it reduces computational complexity, making the model more efficient and less prone to overfitting.

Pooling functions normally include an average function and a maximum function. For the maximum pooling function, the pooling layer is defined as (4) [34].

$$\rho_{n,k} = \max_{m=1}^{M} \left(x_{n,(k-1)\times s+m}\right) \tag{4}$$

Where
M represents the pooling size.
s represents the stride size.

And for the average pooling function, the pooling layer output is defined as (5) [34].

$$\rho_{n,k} = \beta \sum_{m=1}^{M} \left(x_{n,(k-1)\times s+m}\right) \tag{5}$$

Where
β represents the scale factor.
s represents the stride size.

It is believed that maximum pooling performance is better than average pooling [35]. In this paper, the maximum pooling (MaxPooling 1D) has been employed.

B. LSTM

For a standard RNN, if the input sequence $i = (i_1, i_2, \dots, i_N)$ is known, (6) and (7) can be used to find the hidden layer sequence $l = (l_1, l_2, \dots, l_N)$ and the output $j = (j_1, j_2, \dots, j_N)$, respectively, by using an iterative method from $n = 1$ to N [35].

$$l_n = \Delta(W_{il} i_n + W_{ll} l_{n-1} + u_l) \tag{6}$$

$$j_n = W_{lj} l_n + u_j \tag{7}$$

Where
$i_n (n = 1,2, \dots, N)$ is a P dimensional vector.
$l_n (n = 1,2, \dots, N)$ is a Q dimensional vector.
$j_n (n = 1,2, \dots, N)$ is an R dimensional vector.
W_{il} represents the input-hidden layer weight matrix.
u represents the bias vector.
$\Delta(\,)$ represents the activation function.

LSTMs are believed to perform better than simple RNNs. To better grasp (8) to (12), a simple LSTM cell is depicted in Fig. 5. The activation function $\Delta(\,)$ is calculated as follows [36]:

$$e_n = \sigma\,(W_{ie} i_n + W_{le} l_{n-1} + W_{he} h_{n-1} + u_e) \tag{8}$$

$$r_n = \sigma\,(W_{ir} i_n + W_{lr} l_{n-1} + W_{hr} h_{n-1} + u_r \tag{9}$$

$$h_n = r_n h_{n-1} + e_n \tanh(W_{ih} i_n + W_{lh} l_{n-1} + u_h) \tag{10}$$

$$z_t = \sigma\,(W_{iz} i_n + W_{lz} l_{n-1} + W_{hz} h_{n-1} + u_z \tag{11}$$

$$v_t = z_t \tanh(h_t) \tag{12}$$

Where
σ represents the sigmoid function.
e, r, h, z represent the input gate, forgetting gate, output gate, and the cell activation vector, respectively.

C. LEARNABLE SKIP CONNECTIONS

Traditional skip or residual connections were introduced as a solution to the vanishing gradient problem in deep networks. These connections allow information to bypass one or more layers, and flow directly from one part of the network to

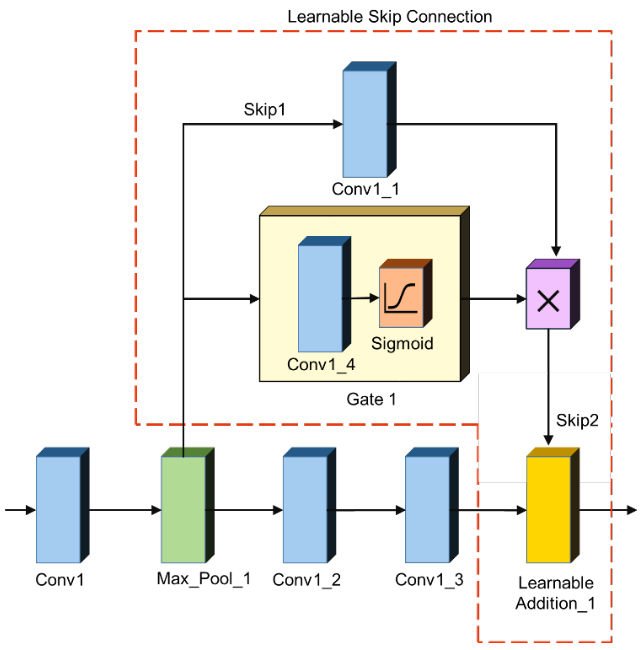


FIGURE 6. A learnable skip connection with gating mechanism.

another. This strategy aids in backpropagation by creating an unobstructed path for gradients to flow, enabling successful training of deeper networks.

This research presents a novel method of enhancing these skip connections with learnability. Unlike standard skip connections, learnable skip connections incorporate gating mechanisms to control the flow of information dynamically. Essentially, these mechanisms learn to regulate what information is useful to propagate forward and what can be omitted. A learnable skip connection with gating mechanism is depicted in Fig. 6.

In the proposed SkipGateNet model, each learnable skip connection employs a 1D convolution layer followed by a sigmoid activation function to create a gate. The 1D convolution layer acts as a learnable filter, learning the importance of each feature in the data. This output is then passed through a sigmoid function, which scales the values between 0 and 1, effectively determining the proportion of information that should be forwarded through the skip connection. This dynamic information flow is given in Fig. 7 and the gating mechanism is mathematically expressed as;

$$gate = sigmoid(Conv1D(pool)) \tag{13}$$

$$skip\_con = Multiply()\ ([skip\_con, gate]) \tag{14}$$

Where 'Conv1D (pool)' denotes the 1D convolution operation on the pooling layer output, 'sigmoid' is the sigmoid activation function, and 'Multiply()' represents the element-wise multiplication operation between the original skip connection and the gate output.

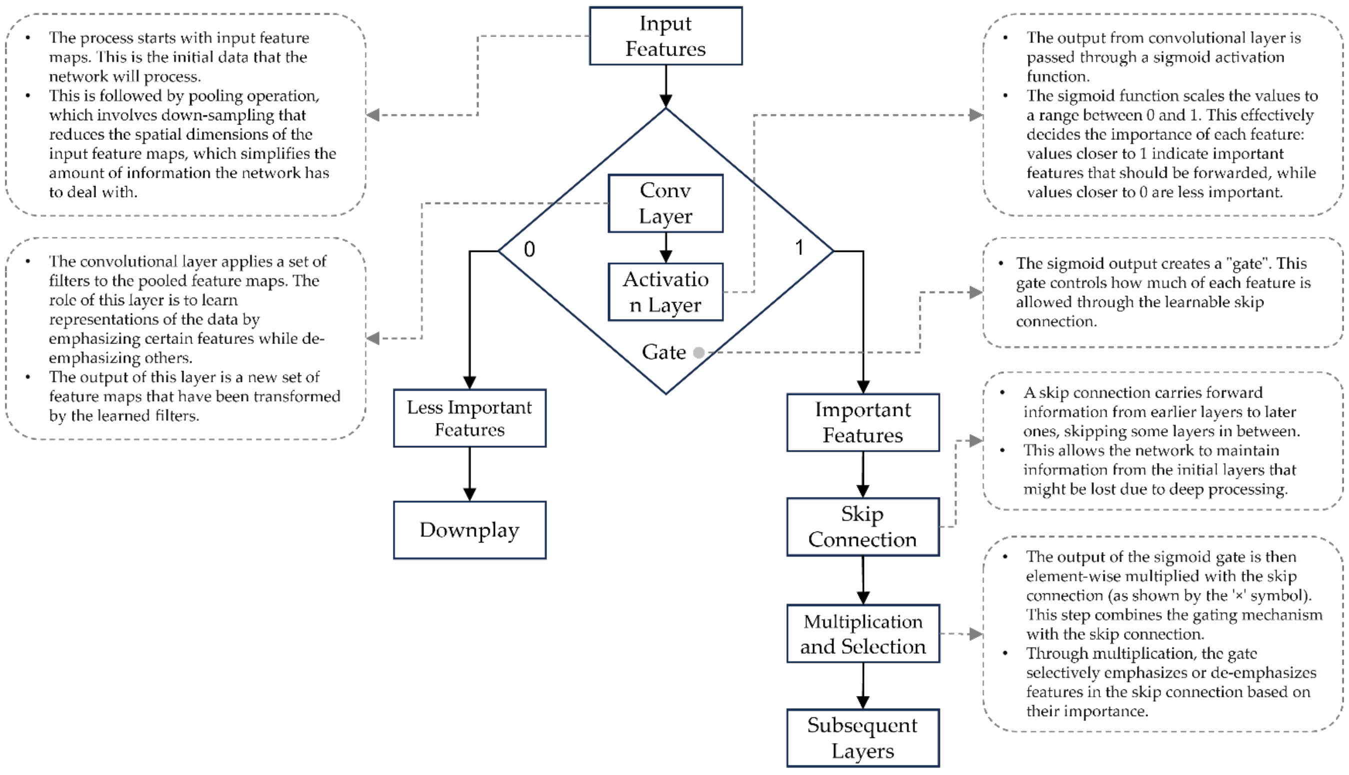

• The process starts with input feature maps. This is the initial data that the network will process.
• This is followed by pooling operation, which involves down-sampling that reduces the spatial dimensions of the input feature maps, which simplifies the amount of information the network has to deal with.
• The convolutional layer applies a set of filters to the pooled feature maps. The role of this layer is to learn representations of the data by emphasizing certain features while de-emphasizing others.
• The output of this layer is a new set of feature maps that have been transformed by the learned filters.
Input Features
Conv Layer
Activation Layer
Gate
0
1
Less Important Features
Downplay
Important Features
Skip Connection
Multiplication and Selection
Subsequent Layers
• The output from convolutional layer is passed through a sigmoid activation function.
• The sigmoid function scales the values to a range between 0 and 1. This effectively decides the importance of each feature: values closer to 1 indicate important features that should be forwarded, while values closer to 0 are less important.
• The sigmoid output creates a "gate". This gate controls how much of each feature is allowed through the learnable skip connection.
• A skip connection carries forward information from earlier layers to later ones, skipping some layers in between.
• This allows the network to maintain information from the initial layers that might be lost due to deep processing.
• The output of the sigmoid gate is then element-wise multiplied with the skip connection (as shown by the '×' symbol). This step combines the gating mechanism with the skip connection.
• Through multiplication, the gate selectively emphasizes or de-emphasizes features in the skip connection based on their importance.

FIGURE 7. Information flow in Learnable Skip Connection

The learnable skip connections with gating mechanisms add a level of adaptability to our architecture. These connections perform a kind of feature selection, determining which features are important enough to be directly propagated to deeper layers. As a result, the network becomes better at focusing on the most relevant patterns in the data, leading to improved model performance. In the context of the proposed model SkipGateNet, the integration of learnable skip connections provides a significant advantage in dealing with botnet attack detection in IoT. IoT data streams are often noisy and have many redundant features. By using learnable skip connections, our model can learn to focus on the most pertinent information while ignoring irrelevant data, significantly improving detection accuracy.

It is pertinent to mention here that the learnable skip connections were originally introduced in [37]. The authors used a Select, Attend, and Transfer (SAT) gate architecture. The SAT Gate employs a sparsity-constrained selection mechanism for channel selection, followed by an attention mechanism for spatial focus. However, the learnable skip connections used in this paper use a convolution-based gating mechanism with sigmoid functions that dynamically adjusts the flow of information based on the learned importance of features.

D. ARCHITECTURAL DETAILS OF THE PROPOSED MODEL

As illustrated in Fig. 3, the proposed SkipGateNet model comprises three convolutional blocks, each containing multiple convolutional layers enhanced by learnable skip connections with gating mechanisms. These skip connections facilitate the seamless flow of information throughout the network. After the third convolutional block, the output is processed through two Long Short-Term Memory (LSTM) layers. These LSTM layers are designed to address the vanishing gradient problem associated with traditional RNNs, enabling the model to learn long-term dependencies effectively. Subsequently, the output of the LSTM layers is directed through a series of dense layers to generate the final predictions for each input sample. The inclusion of learnable skip connections with gating mechanisms allows the model to dynamically adapt the information flow, improving the model's performance on complex tasks.

The complete overview of the model architecture is given in Table III. The deep learning model architecture begins with an input tensor of dimensions 115 x 1. The first layer is a 1D convolutional layer with 64 filters, a kernel size of 3, and 'same' padding, which generates a 115 x 64 output tensor with 256 parameters. Next, a max-pooling layer with a pool size of 2 reduces the output shape to 57 x 64 without adding any parameters. Subsequently, two 1D convolutional layers with 128 filters, a kernel size of 3, and 'same' padding are added.

TABLE III
THE ARCHITECTURE OF THE PROPOSED MODEL

Layers	Output Shape	Parameters	Connected to Layer
input_3 (InputLayer)	[(None, 115, 1)]	0	[]
conv1d_65 (Conv1D)	(None, 115, 64)	256	['input_3[0][0]']
max_pooling1d_15 (MaxPooling1D)	(None, 57, 64)	0	['conv1d_65[0][0]']
conv1d_66 (Conv1D)	(None, 57, 128)	8320	['max_pooling1d_15[0][0]']
conv1d_69 (Conv1D)	(None, 57, 128)	8320	['max_pooling1d_15[0][0]']
conv1d_67 (Conv1D)	(None, 57, 128)	24704	['max_pooling1d_15[0][0]']
multiply_15 (Multiply)	(None, 57, 128)	0	['conv1d_66[0][0]',
conv1d_68 (Conv1D)	(None, 57, 128)	49280	['conv1d_67[0][0]']
add_15 (Add)	(None, 57, 128)	0	['multiply_15[0][0]',
max_pooling1d_16 (MaxPooling1D	(None, 28, 128)	0	['add_15[0][0]']
conv1d_70 (Conv1D)	(None, 28, 128)	16512	['max_pooling1d_16[0][0]']
conv1d_73 (Conv1D)	(None, 28, 128)	16512	['max_pooling1d_16[0][0]']
conv1d_71 (Conv1D)	(None, 28, 128)	49280	['max_pooling1d_16[0][0]']
multiply_16 (Multiply)	(None, 28, 128)	0	['conv1d_70[0][0]',
conv1d_72 (Conv1D)	(None, 28, 128)	49280	['conv1d_71[0][0]']
add_16 (Add)	(None, 28, 128)	0	['multiply_16[0][0]',
max_pooling1d_17 (MaxPooling1D	(None, 14, 128)	0	['add_16[0][0]']
conv1d_74 (Conv1D)	(None, 14, 128)	33024	['max_pooling1d_17[0][0]']
conv1d_77 (Conv1D)	(None, 14, 128)	33024	['max_pooling1d_17[0][0]']
conv1d_75 (Conv1D)	(None, 14, 128)	98560	['max_pooling1d_17[0][0]']
multiply_17 (Multiply)	(None, 14, 128)	0	['conv1d_74[0][0]',
conv1d_76 (Conv1D)	(None, 14, 128)	196864	['conv1d_75[0][0]']
add_17 (Add)	(None, 14, 128)	0	['multiply_17[0][0]',
lstm_10 (LSTM)	(None, 14, 128)	82176	['add_17[0][0]']
lstm_11 (LSTM)	(None, 32)	12416	['lstm_10[0][0]']
flatten_5 (Flatten)	(None, 32)	0	['lstm_11[0][0]']
dense_10 (Dense)	(None, 64)	2112	['flatten_5[0][0]']
dense_11 (Dense)	(None, 32)	2080	['dense_10[0][0]']
dense_12 (Dense)	(None, 11)	363	['dense_11[0][0]']

TABLE IV
SUMMARY OF THE PROPOSED MODEL PRESENTING ITS SIZE AND TOTAL PARAMETERS

Total Parameters	Trainable parameters	Non-Trainable parameters	Model Size in KB
683,083	683,083	0	2596.87

These convolutional layers, along with a Multiply layer representing the gating mechanism, are connected to the previous layers. This combination results in a 57 x 128 output tensor. An Add layer then merges the outputs of the gated convolutional layers. The architecture repeats this pattern for the next convolutional block. The second block starts with a max-pooling layer, followed by two convolutional layers and a gating mechanism, resulting in another 28 x 128 output tensor. The third convolutional block follows a similar pattern, with two convolutional layers and a gating mechanism generating a 14 x 128 output tensor.

Subsequently, two LSTM layers with 64 units each are added. The first LSTM layer, connected to the previous Add layer, has 82,176 parameters and outputs a 14 x 64 tensor. The second LSTM layer, connected to the first LSTM layer, has 12,416 parameters and outputs a 32-dimensional vector.

The architecture concludes with three dense layers. The first dense layer with 64 units connects to a Flatten layer, which reshapes the input tensor into a 1D vector, and has 2,112 parameters. The second dense layer connects to the first dense layer, has 32 units, and contributes 2,080 parameters. The final dense layer comprises 11 units, corresponding to the 11 classes, and has 363 parameters.

This model architecture is well-suited for classifying multiple classes and particularly excels in managing complex data due to the added learnable skip connections with gating mechanisms. The blend of 1D convolutional layers, max-pooling layers, multiply layers for gating, LSTM layers, and dense layers results in a powerful deep learning model capable of producing accurate classifications.

E. SIZE AND PARAMETERS OF THE PROPOSED MODEL

A summary of the proposed deep learning model presenting the total parameters and its size is given in Table IV. The model consists of a total of 683,083 parameters, all of which are trainable. The table displays the model size as 2596.87 KB, which represents the memory required to store the model and its associated parameters.

A smaller model size is generally preferred, as it enables easier deployment on devices with limited memory capacity, such as edge devices and fog nodes. This compactness allows the model to be more efficient, making it suitable for the real-time classification of IoT attacks across various IoT platforms.

VI. PERFORMANCE EVALUATION AND RESULTS

A. PERFORMANCE PARAMETERS

To assess the performance of each model, the key performance parameters, such as precision, recall, accuracy, and F1-score, have been computed. These parameters were calculated using (15) to (18). The quantities involved in the calculation of the aforementioned performance parameters, specifically True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN), are obtained from confusion matrices. Precision, recall, and F1 score serve as metrics for evaluating the performance of a classification model, while accuracy is employed to gauge the overall correctness of predictions made by a model.

$$Precision = \frac{nTP}{nTP + nFP} \tag{15}$$

$$Recall = \frac{nTP}{nFN + nTP} \tag{16}$$

$$Accuracy = \frac{nTP + nTN}{nFP + nTP + nTN + nFN} \tag{17}$$

$$F1 - score = 2\,\text{x}\,\frac{Precision\ x\ Recall}{Precision + Recall} \tag{18}$$

B. RELIABILITY PARAMETERS

Cohen's Kappa coefficient and Matthews Correlation Coefficient are two metrics frequently used to evaluate the performance of a classification model. This paper calculates both to gauge the reliability of the proposed model.

1) COHEN'S KAPPA COEFFICIENT

The Cohen's Kappa coefficient quantifies the agreement between predicted and actual classifications while accounting for the possibility of chance agreement [38]. Ranging from -1 to 1. A score of 1 indicates perfect agreement, 0 signifies chance agreement, and -1 demonstrates perfect disagreement.

The Kappa coefficient or the Kappa statistic is a measure of two accuracies, i.e., the observed accuracy and the expected accuracy, which depend on the obtained confusion matrices [38, 39]. The observed accuracy is defined as the ratio of actual predicted labels from all the labels in a confusion matrix and can be calculated by using (17). Whereas the expected accuracy, which is dependent on the predicated and actual labels, can be calculated using (18). After obtaining the observed and expected accuracies, the kappa coefficient can easily be calculated using (19).

$$Obs\ Acc = \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i} \tag{17}$$

$$Exp\ Acc = \frac{(\sum_i (Pred\_labels \times\ Actual\_labels))}{TP_i +\ TN_i + FP_i + FN_i} \tag{18}$$

$$Kappa = \frac{Obs\ Acc -\ Exp\ Acc}{1 - Exp\ Acc} \tag{19}$$

Where $i\ \in 0,1,2,3, \dots ,9,10$, and it represents the 11 classes used in this study.

2) MATTHEWS CORRELATION COEFFICIENT

The Matthews correlation coefficient (MCC) is a metric measuring the quality of classification when the data is imbalanced [40]. Considering true and false positives as well as true and false negatives, the MCC ranges from -1 to 1. A score of 1 indicates perfect prediction, 0 signifies random prediction, and -1 demonstrates completely wrong prediction. MCC is calculated using (20).

$$MCC = \frac{TP_i \times TN_i -\ FP_i \times FN_{i_i}}{\sqrt{(TP_i + FP_i)(TP_i + FN_i)(TN_i + FP_i)(TN_i + FN_i)}} \tag{20}$$

C. RESULTS AND DISCUSSION

1) TRAINING AND VALIDATION RESULTS

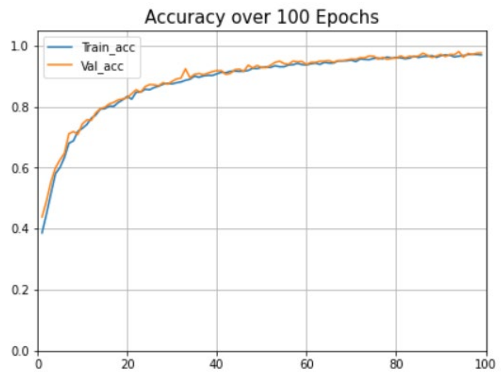
The performance of the proposed model during training was assessed using key metrics, including training accuracy, validation accuracy, training loss, and validation loss at various epochs. These metrics help in evaluating the overfitting and underfitting of the trained models. Fig. 8 (a) and Fig. 8 (b) display the training loss versus validation loss and the training accuracy versus validation accuracy graphs of the proposed model, respectively. The proposed model demonstrates a low training loss 0.096% and an excellent training accuracy of 99.93%. The training performance of the proposed model is also given in Table V.

2) CLASS-WISE CLASSIFICATION REPORT

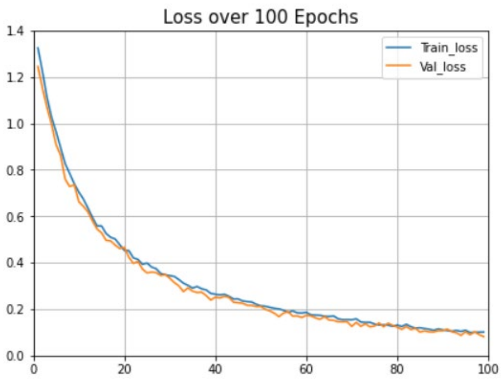
The performance of the proposed model on the test dataset has also been measured using the key performance parameters, i.e., precision, recall, F1-score, and accuracy. A class-wise classification report that presents the proposed model's class-wise performance is given in Table VI. The confusion matrices for the testing and validation dataset have also been generated and are shown in Fig. 9 and Fig. 11. Fig.9 shows the confusion matrices of the model's performance on each subset of the dataset, whereas Fig. 11 shows an aggregate confusion matrix of the complete test dataset. The confusion matrices help in the calculation of the aforementioned performance parameters. The proposed model's exceptional performance is evident from the results given in Table VI. In addition to the performance of each class, the macro-averages of precision, recall, and F1-score stand at 99.00, signifying overall strong performance. The total accuracy of the model is 99.91%, depicting that the model has correctly classified 99.91% of instances in the dataset. Moreover, the class-wise comparison of the classification is also given in Fig. 10.

3) RELIABILITY PARAMETERS' RESULTS

As described earlier, Cohen's Kappa coefficient and Matthews Correlation Coefficient have been calculated to check the proposed models' reliability. Table VII presents each class's Kappa coefficient and Matthews Correlation Coefficient (MCC) values. For the Kappa coefficient, a score



(a)



(b)

FIGURE 8. Training Curves on complete dataset (a) Training loss vs. validation loss, and (b) Training accuracy vs. validation accuracy

TABLE V
AGGREGATE TRAINING PERFORMANCE OF THE PROPOSED MODEL FOR ALL SUBSETS

Model	Epochs	Training Loss	Validation Loss	Training Accuracy	Validation
The Proposed Model	1	1.336	1.381	39.32%	38.05%
	.	.	.	.	.
	.	.	.	.	.
	99	0.031	0.1101	99.38%	97.01%
	100	0.096	0.0915	99.93%	99.95%

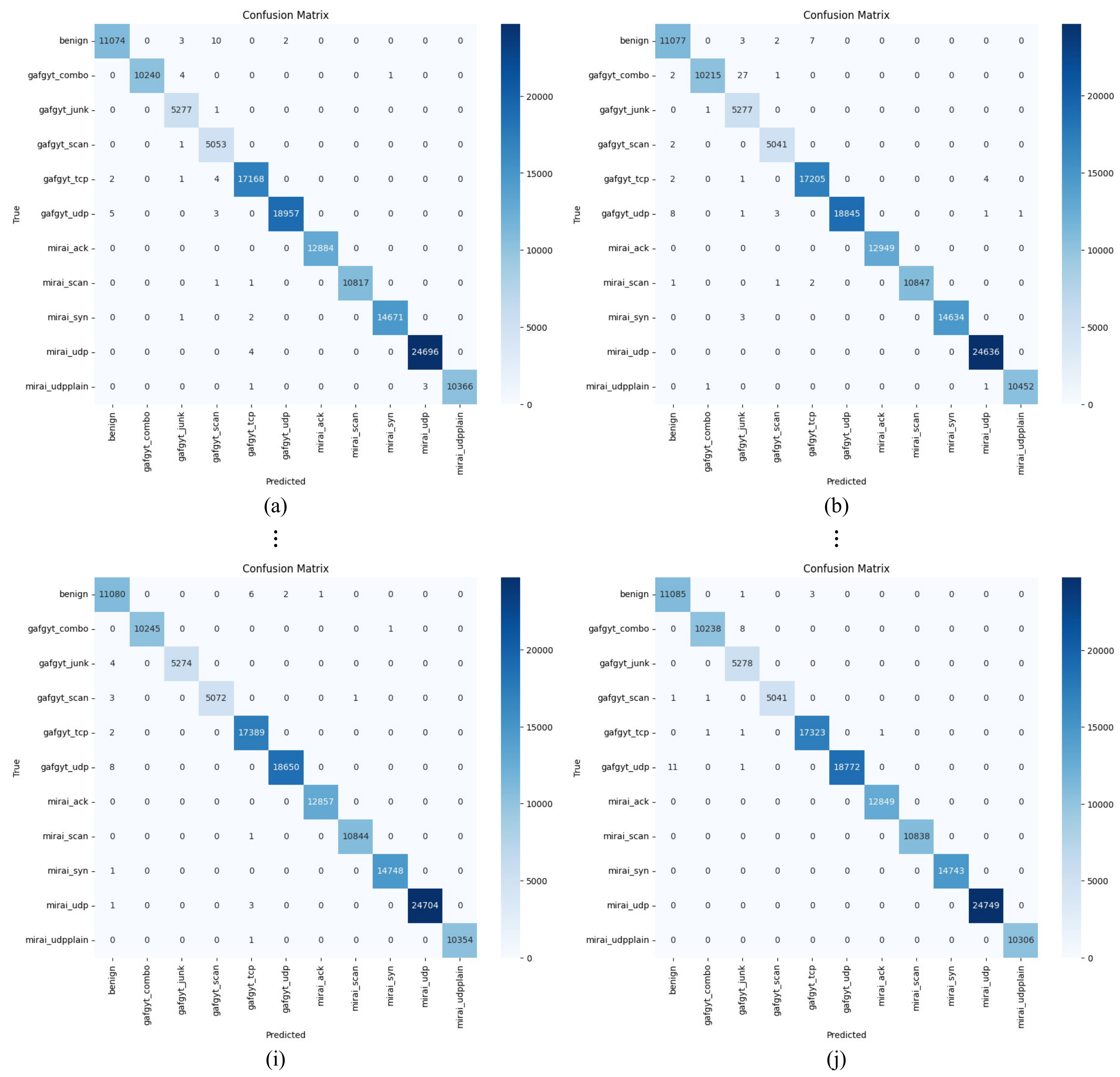


FIGURE 9. Confusion matrices of Testing dataset for all 10 subsets: (a) Subset 1, (b) Subset 2, ⋯, (j) Subset 9, and (j) Subset 10.

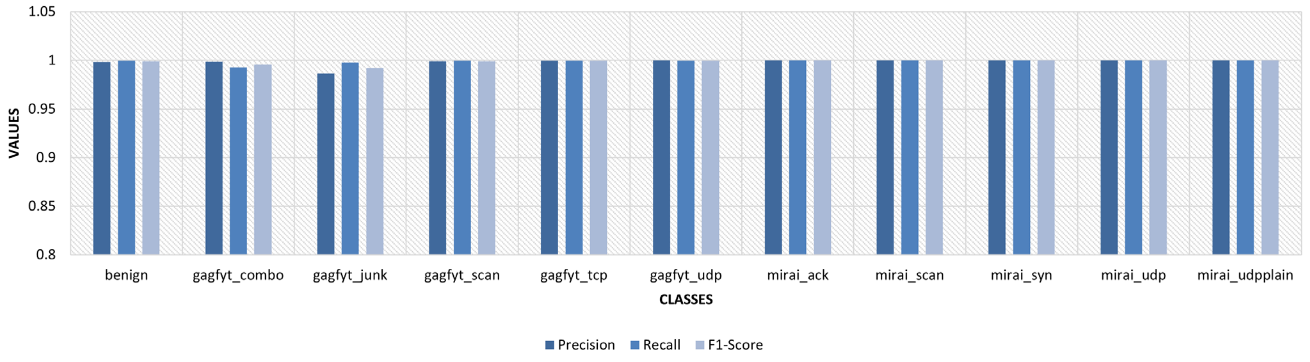


FIGURE 10. Class-wise comparison of the aggregate classification parameters.

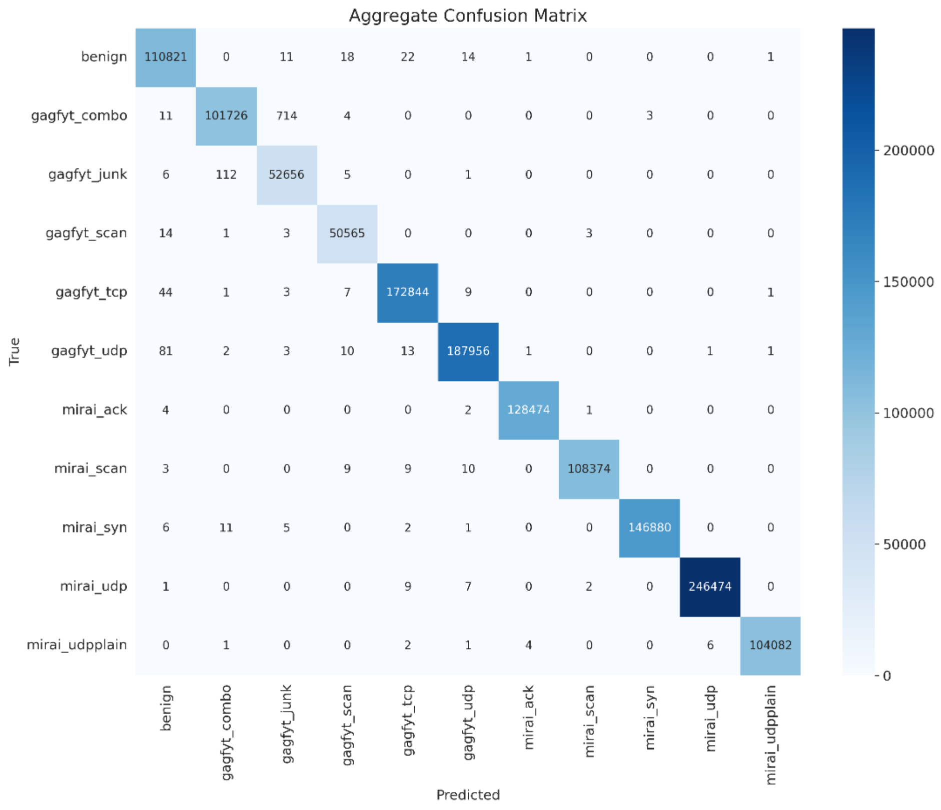


FIGURE 11. Aggregate confusion matrix for the complete dataset.

TABLE VI
CLASSIFICATION REPORT OF EACH CLASS ON THE TEST DATASET

Class	Aggregate Precision	Aggregate Recall	Aggregate F1-score	Test Samples
benign	0.998469	0.999396	0.998932	110888
gafgyt_combo	0.998754	0.992856	0.995766	102458
gafgyt_junk	0.986533	0.997651	0.991957	52780
gafgyt_scan	0.998955	0.999585	0.99927	50586
gafgyt_tcp	0.99967	0.999624	0.999647	172909
gafgyt_udp	0.99976	0.999405	0.999582	188068
mirai_ack	0.999953	0.999945	0.999949	128481
mirai_scan	0.999945	0.999714	0.999829	108405
mirai_syn	0.99998	0.999829	0.999905	146905
mirai_udp	0.999972	0.999923	0.999947	246493
mirai_udpplain	0.999971	0.999865	0.999918	104096
Macro Average	**0.99**	**0.99**	**0.99**	**1412069**
Accuracy		**99.91%**		

TABLE VII
RESULTS OF RELIABILITY PARAMETERS

Class	Kappa	MCC
benign	0.999831	0.99035
gafgyt_combo	0.999388	0.99002
gafgyt_junk	0.999388	0.98004
gafgyt_scan	0.999948	0.99501
gafgyt_tcp	0.999912	0.99505
gafgyt_udp	0.999887	0.99005
mirai_ack	0.999991	0.99506
mirai_scan	0.999974	0.99026
mirai_syn	0.99998	0.98965
mirai_udp	0.999981	0.99368
mirai_udpplain	0.999988	0.99007

of 1 indicates perfect agreement, 0 signifies chance agreement, and -1 demonstrates perfect disagreement. It can be seen from Table VII that all values are quite close to 1 or 0.99, which more precisely confirms the performance reliability of the proposed model. Similarly, the MCC values for all classes are also 0.99, further validating the proposed model's performance.

4) HYPERPARAMETERS TUNING RESULTS.

Hyper-parameters are parameters that are not learned by the machine learning algorithm during training but are set before training commences. These parameters govern the algorithm's behavior and can significantly influence the model's performance. In deep learning, these parameters include the learning rate, batch size, number of epochs, optimizer, activation functions, and number of layers, among

TABLE VIII
DETAILS OF HYPERPARAMETERS

Parameter	Value	Test Accuracy
Learning Rate	1e-2	99.41%
	1e-3	99.95%
	1e-4	90.75%
	1e-5	96.36%
	1e-6	88.41%
	1e-7	34.08%
Dense Layers Units	32	98.35%
	64	99.95%
	128	98.75%
LSTM layers units	32	99.75%
	64	99.95%
	128	98.75%

others. Table VIII provides the hyper-parameters tuned in this study, which have been optimized using the Keras Tuner. These parameters include the learning rate, the number of units in the dense layers, and the number of units in the LSTM layers. Various values for each hyper-parameter were tested, and the corresponding test accuracy is reported in the table. The proposed model's overall test accuracy of 99.91% shows the fine-tuning of these parameters.

5) INFERENCE TIME

Inference time refers to the duration required for a model to process the input data and generate a prediction or classification regarding the presence of malicious activity or an intrusion attempt. Lower inference times are generally preferred, especially for edge devices and fog computing environments, as they enable faster decision-making and more effective intrusion prevention. Table IX shows the inference time of all layers of the proposed model, along with the average inference time of the model. The overall average inference time of the proposed model is only 8 ms, which makes it suitable for intrusion detection systems intended for edge devices and fog computing environments.

TABLE IX
INFERENCE TIME OF ALL LAYERS OF THE PROPOSED MODEL

Layers	Inference time in ms
input_3 (InputLayer)	17.657
conv1d_65 (Conv1D)	1.321
max_pooling1d_15 (MaxPooling1D)	15.258
conv1d_66 (Conv1D)	16.954
conv1d_69 (Conv1D)	13.857
conv1d_67 (Conv1D)	11.258
multiply_15 (Multiply)	1.112
conv1d_68 (Conv1D)	14.256
add_15 (Add)	12.632
max_pooling1d_16 (MaxPooling1D	14.256
conv1d_70 (Conv1D)	1.005
conv1d_73 (Conv1D)	18.528
conv1d_71 (Conv1D)	10.756
multiply_16 (Multiply)	9.865
conv1d_72 (Conv1D)	19.256
add_16 (Add)	2.056
max_pooling1d_17 (MaxPooling1D	2.025
conv1d_74 (Conv1D)	2.349
conv1d_77 (Conv1D)	5.256
conv1d_75 (Conv1D)	5.658
multiply_17 (Multiply)	4.256
conv1d_76 (Conv1D)	2.589
add_17 (Add)	2.056
lstm_10 (LSTM)	2.654
lstm_11 (LSTM)	2.799
flatten_5 (Flatten)	5.665
dense_10 (Dense)	5.657
dense_11 (Dense)	6.656
dense_12 (Dense)	4.346
Average inference time	7.999 ≈ 8.0

VII. COMPARITIVE ANALYSIS OF THE PROPOSED MODEL

A. COMPARISON WITH DL MODELS IN LITERATURE AND SELF-IMPLEMENTED ARCHITECTURES

In addition to being lightweight, compact in size and fast, the proposed SkipGateNet model was intended to detect botnet attacks effectively. The performance of the SkipGateNet has been compared with the five recent state-of-the-art works that used deep learning (including CNN+LSTM) on the same dataset (N-BaIoT). It is evident from the results shown in Table X that the proposed SkipGateNet outperformed all models in terms of F1-score, precision, recall and accuracy, validating the efficient detection of botnet attacks.

In addition, four different types of deep learning architectures also have been implemented, trained and tested on the same dataset for comparison purposes. Experimented models include:

- **Model A:** a sequential CNN having 5 convolution layers and 3 dense layers.
- **Model B:** a sequential CNN having 7 convolution layers and 3 dense layers.
- **Model C:** a CNN+LSTM having 7 convolution layers, 2 LSTM layers, and a dense layer.
- **Model D:** a CNN+LSTM with simple skip connections, 7 convolution layers, 2 LSTM layers, and a dense layer.

The architectural details of the model are given in Fig. 12 and the comparison of these models with the proposed SkipGateNet model in terms of test accuracy, inference time, total parameters, and size in KBs is given in Table XI. It can be seen that the proposed SkipGateNet has highest accuracy of 99.91% in the fastest inference time of 8 milliseconds. The simple sequential CNNs have less inference time but they exhibit low accuracies and fail to extract features efficiently.

B. COMPARISON WITH MACHINE LEARNING MODELS

The proposed deep learning model has been intended to detect botnet attacks in edge devices and fog computing environments and is presented to be more accurate and faster than the existing traditional machine learning techniques. Therefore, several traditional machine learning models have

TABLE X
COMPARISON OF THE PROPOSED SKIPGATENET WITH OTHER DL MODELS WITH SAME DATASET

Paper	Dataset	Algorithm	F1-score	Precision	Recall	Accuracy
[28]	N-BaIot	CNN-LSTM	0.88	93.04%	91.91%	90.88%
[29]	N-BaIoT	DBN	0.92	98.27%	92.82%	95.60%
[30]	N-BaIoT	CNN-LSTM	0.93	93.48%	93.675	94.30%
[31]	N-BaIoT	Auto encoders	-	-	-	90.2%
[32]	N-BaIoT	Auto encoders, DNN	0.80	99%	66%	97.21%
[41]	N-BaIoT	Auto encoders	0.99	99%	99%	-
The Proposed SkipGateNet	**N-BaIoT**	**CNN-LSTM with learnable skip connections**	**0.99**	**99%**	**99%**	**99.91%**

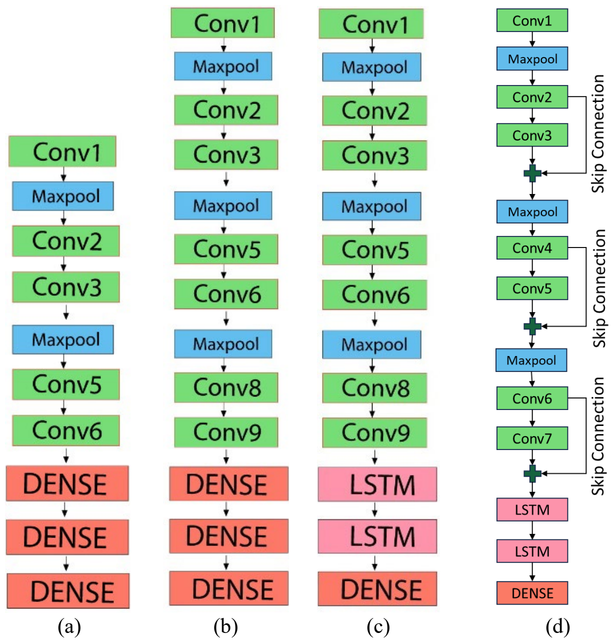


FIGURE 12. Architectures of implemented models on same dataset for comparison purposes; (a) Model A: a sequential CNN with 5 convolution layers, (b) Model B: a sequential CNN with 7 convolution layers, (c) Model C: a CNN+LSTM architecture, and (d) Model D: a CNN+LSTM with simple skipping connections.

been implemented on the same dataset to compare its performance and inference time. Five traditional machine learning models have been implemented for the comparison with the proposed model, i.e., Logistic Regression, Random Forest, SVM, Naïve Bayes, and K Neighbors Classifier. Models have been compared on two metrics, test accuracy and inference time. It can be seen from Table XII that the proposed deep learning model has the highest accuracy of 99.91% and has a minimum inference time of just 8.0 ms.

TABLE XI
COMPARISON OF THE PROPOSED MODEL WITH SIMILAR ARCHITECTURES

Models	Test accuracy	Inference time (ms)	Size in KB	Total Parameters
Model A	92.19%	8.1	2645.65	1,437,515
Model B	95.34%	9.9	2684.97	1,733,195
Model C	96.53%	10.4	2697.12	1,661,803
Model D	99.15%	8.3	2618.07	739,915
The Proposed SkipGateNet	**99.91%**	**8.0**	**2596.87**	**683,083**

TABLE XII
COMPARISON OF THE PROPOSED MODEL WITH ML MODELS

Models	Test Accuracy	Inference Time in ms
Logistic Regression	82.56%	11.8
Random forest	99.05%	10.7
SVM	82.45%	9.6
Naïve Bayes	60. 48%	10.3
K Neighbors Classifier	98.98%	9.3
The Proposed SkipGateNet	**99.91%**	**8.0**

VIII. ABLATION STUDY

An ablation study involves systematically removing or altering certain components of the model to understand the impact of each component on the model's performance. We used the same dataset and experimental setup as for the baseline or proposed. Different components of the model were removed and then added step-by-step or removed to compare their performance. It can be seen in Table XIII that simple sequential convolutional components exhibited low accuracies with large model sizes. Adding skip connections helped in reducing the model size and improving the accuracy. While adding the learnable skip connections, significantly improved the accuracy of the model with the smallest size and fast inference time. The ablation study validated the efficacy of the proposed architecture to detect botnet attacks.

TABLE XIII
RESULTS OF THE ABLATION STUDY

Exp. No.	Components	Accuracy	Inference Time (ms)	Model Size (KB)
1	1 Convolutional Block	95.38%	5.12	10,976.19
2	2 Convolutional Blocks	96.66%	6.89	5,365.12
3	3 Convolutional Block	98.21%	9.20	7,912.00
4	1 Convolutional Block + Simple Skip Connections	98.25%	6.49	10,282.61
5	2 Convolutional Blocks + Simple Skip Connections	98.69%	9.11	6,134.25
6	3 Convolutional Blocks + Simple Skip Connections	99.72%	11.36	6,951.73
7	3 Convolutional Blocks + Learnable Skip Connections	99.70%	7.78	5,560.08
8	**3 Convolutional Blocks + Learnable Skip Connections + LSTM (The Proposed Model)**	**99.91%**	**7.99 ≈ 8.0**	**2,596.87**

IX. CONCLUSION

This paper addressed the critical need for effective and efficient Intrusion Detection Systems (IDS) to detect botnet attacks, especially in IoT and Fog computing environments. Such resource-constrained environments need small sizes and fast yet powerful decision-making models to detect malicious intrusions in the network. For the challenges mentioned above, a 1D-CNN and LSTM-based deep neural network with learnable skip connections was proposed and presented in this paper. This combination of convolutional and LSTM layers enables the model to learn both temporal and spatial features in the data, while the learnable skip connections are capable of dynamically controlling the flow of information across the network, enabling the model to focus on salient features and ignore irrelevant ones, thus enhancing its detection capabilities. The proposed model was trained and tested on actual IoT network traffic data (the N-BaIoT dataset). This dataset features authentic traffic data from nine commercial IoT devices, including cameras, routers, and smart home appliances infected with the Mirai and BASHLITE malware, incorporating a total of 10 different IoT attacks. With a compact size of 2596.87 KB, an inference time of 8.0 milliseconds, and a test accuracy of 99.91%, the proposed model proved to be well-suited to be deployed in resource-constrained environments. The proposed SkipGateNet model outperformed all models in comparison in terms of accuracy and inference time. Furthermore, the future research could explore the integration of SkipGateNet with federated learning for distributed IoT environments, and the application of transfer learning to enhance its adaptability to different IoT domains and attack types.

ACKNOWLEDGMENT

The authors are thankful to the Deanship of Scientific Research at Najran University for funding this work under the General Research Funding program grant code (NU/DRP/SERC/12/46).

REFERENCES

[1] N. Mishra and S. Pandya, “Internet of Things Applications, Security Challenges, Attacks, Intrusion Detection, and Future Visions: A Systematic Review,” IEEE Access, vol. 9, pp. 59353–59377, 2021, doi: https://doi.org/10.1109/access.2021.3073408.

[2] Y. Li, Y. Zuo, H. Song, and Z. Lv, “Deep Learning in Security of Internet of Things,” IEEE Internet of Things Journal, vol. 9, no. 22, pp. 22133–22146, 2021, doi: https://doi.org/10.1109/jiot.2021.3106898.

[3] D. Ucci, L. Aniello, and R. Baldoni, “Survey of machine learning techniques for malware analysis,” Computers & Security, vol. 81, pp. 123–147, Mar. 2019, doi: https://doi.org/10.1016/j.cose.2018.11.001.

[4] M. K. Kagita, N. Thilakarathne, T. R. Gadekallu, P. K. Reddy, and S. Singh , “A Review on Cyber Crimes on the Internet of Things,” in Deep Learning for Security and Privacy Preservation in IoT, N. Kumar, Ed., Springer, 2022, pp. 83–98.

[5] Y. Meidan et al., “N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders,” IEEE Pervasive Computing, vol. 17, no. 3, pp. 12–22, Jul. 2018, doi: https://doi.org/10.1109/mprv.2018.03367731.

[6] H. Atlam, R. Walters, and G. Wills, “Fog Computing and the Internet of Things: A Review,” Big Data and Cognitive Computing, vol. 2, no. 2, p. 10, Apr. 2018, doi: https://doi.org/10.3390/bdcc2020010.

[7] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, “A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures,” IEEE Access, vol. 7, pp. 82721–82743, 2019, doi: https://doi.org/10.1109/access.2019.2924045.

[8] A. Sharifi and S. Goli-Bidgoli, “IFogLearn++: A new platform for fog layer’s IoT attack detection in critical infrastructure using machine learning and big data processing,” Computers and Electrical Engineering, vol. 103, p. 108374, Oct. 2022, doi: https://doi.org/10.1016/j.compeleceng.2022.108374.

[9] M. M. Salim, S. Rathore, and J. H. Park, “Distributed denial of service attacks and its defenses in IoT: a survey,” The Journal of Supercomputing, vol. 76, no. 7, Jul. 2019, doi: https://doi.org/10.1007/s11227-019-02945-z.

[10] M. Antonakakis et al., “Understanding the Mirai Botnet,” in 26th USENIX security symposium , 2017, pp. 1093–1110.

[11] H. Liu and B. Lang, “Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey,” Applied Sciences, vol. 9, no. 20, p. 4396, Oct. 2019, doi: https://doi.org/10.3390/app9204396.

[12] S. Omar, A. Ngadi, and H. H. Jebur, “Machine Learning Techniques for Anomaly Detection: An Overview,” International Journal of Computer Applications, vol. 79, no. 2, pp. 33–41, Oct. 2013, doi: https://doi.org/10.5120/13715-1478.

[13] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, “Deep Learning for IoT Big Data and Streaming Analytics: A Survey,” IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 2923–2960, 2018, doi: https://doi.org/10.1109/comst.2018.2844341.

[14] Verma and V. Ranga, “Machine Learning Based Intrusion Detection Systems for IoT Applications,” Wireless Personal Communications, vol. 111, no. 4, Nov. 2019, doi: https://doi.org/10.1007/s11277-019-06986-8.

[15] M. A. Rahman, A. T. Asyhari, L. S. Leong, G. B. Satrya, M. Hai Tao, and M. F. Zolkipli, “Scalable machine learning-based intrusion detection system for IoT-enabled smart cities,” Sustainable Cities and Society, vol. 61, p. 102324, Oct. 2020, doi: https://doi.org/10.1016/j.scs.2020.102324.

[16] Amouri, V. T. Alaparthy, and S. D. Morgera, “A Machine Learning Based Intrusion Detection System for Mobile Internet of Things,” Sensors, vol. 20, no. 2, p. 461, Jan. 2020, doi: https://doi.org/10.3390/s20020461.

[17] N. Islam et al., “Towards Machine Learning Based Intrusion Detection in IoT Networks,” Computers, Materials & Continua, vol. 69, no. 2, pp. 1801–1821, 2021, doi: https://doi.org/10.32604/cmc.2021.018466.

[18] M. Sarhan, S. Layeghy, N. Moustafa, M. Gallagher, and M. Portmann, “Feature extraction for machine learning-based intrusion detection in IoT networks,” Digital Communications and Networks, Sep. 2022, doi: https://doi.org/10.1016/j.dcan.2022.08.012.

[19] M. Asif, S. Abbas, M. A. Khan, A. Fatima, M. A. Khan, and S.-W. Lee, “MapReduce based intelligent model for intrusion detection using machine learning technique,” Journal of King Saud University - Computer and Information Sciences, vol. 34, no. 10, Dec. 2021, doi: https://doi.org/10.1016/j.jksuci.2021.12.008..

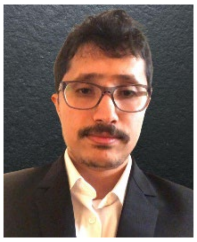
[20] R. Kumar, P. Kumar, R. Tripathi, G. P. Gupta, S. Garg, and M. M. Hassan, “A distributed intrusion detection system to detect DDoS attacks in blockchain-enabled IoT network,” Journal of Parallel and Distributed Computing, vol. 164, pp. 55–68, Jun. 2022, doi: https://doi.org/10.1016/j.jpdc.2022.01.030.

[21] A. A. Ahmed, W. A. Jabbar, A. S. Sadiq, and H. Patel, “Deep learning-based classification model for botnet attack detection,” Journal of Ambient Intelligence and Humanized Computing, vol. 13, no. 7, Mar. 2020, doi: https://doi.org/10.1007/s12652-020-01848-9.

[22] N. Faruqui et al., “SafetyMed: A Novel IoMT Intrusion Detection System Using CNN-LSTM Hybridization,” Electronics, vol. 12, no. 17, pp. 3541–3541, Aug. 2023, doi: https://doi.org/10.3390/electronics12173541.

[23] Md. A. Talukder et al., “A dependable hybrid machine learning model for network intrusion detection,” Journal of Information

Security and Applications, vol. 72, no. 10, p. 103405, Feb. 2023, doi: https://doi.org/10.1016/j.jisa.2022.103405.
[24] T. Hasan et al., "Securing Industrial Internet of Things Against Botnet Attacks Using Hybrid Deep Learning Approach," IEEE Transactions on Network Science and Engineering, pp. 1–1, 2022, doi: https://doi.org/10.1109/TNSE.2022.3168533.
[25] T. Zebin, S. Rezvy, and Y. Luo, "An Explainable AI-Based Intrusion Detection System for DNS Over HTTPS (DoH) Attacks," IEEE Transactions on Information Forensics and Security, vol. 17, pp. 2339–2349, 2022, doi: https://doi.org/10.1109/tifs.2022.3183390.
[26] P. Illy, G. Kaddoum, K. Kaur, and S. Garg, "ML-Based IDPS Enhancement With Complementary Features for Home IoT Networks," IEEE Transactions on Network and Service Management, vol. 19, no. 2, pp. 772–783, Jun. 2022, doi: https://doi.org/10.1109/tnsm.2022.3141942.
[27] Pei, X. Deng, S. Tian, L. Zhang, and K. Xue, "A Knowledge Transfer-based Semi-Supervised Federated Learning for IoT Malware Detection," IEEE Transactions on Dependable and Secure Computing, vol. 20, no. 3, pp. 1–1, 2022, doi: https://doi.org/10.1109/tdsc.2022.3173664.
[28] H. Alkahtani and T. H. H. Aldhyani, "Botnet Attack Detection by Using CNN-LSTM Model for Internet of Things Applications," Security and Communication Networks, vol. 2021, pp. 1–23, Sep. 2021, doi: https://doi.org/10.1155/2021/3806459.
[29] Tran Viet Khoa et al., "Collaborative Learning Model for Cyberattack Detection Systems in IoT Industry 4.0," in 2020 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2020.
[30] G. De La Torre Parra, P. Rad, K.-K. R. Choo, and N. Beebe, "Detecting Internet of Things attacks using distributed deep learning," Journal of Network and Computer Applications, vol. 163, p. 102662, Aug. 2020, doi: https://doi.org/10.1016/j.jnca.2020.102662.
[31] N. Sakthipriya, V. Govindasamy, and V. Akila, "A Comparative Analysis of various Dimensionality Reduction Techniques on N-BaIoT Dataset for IoT Botnet Detection," in 2023 2nd International Conference on Paradigm Shifts in Communications Embedded Systems, Machine Learning and Signal Processing (PCEMS), 2023.
[32] M. A. Pynadath, K. J. Pavithra, and S. Elton Lobo, "Anomaly Detection and Multi-Output Classification of IoT Attacks," in 2023 International Conference on Inventive Computation Technologies (ICICT), IEEE, 2023.
[33] Kaggle, "N-BaIoT Dataset to Detect IoT Botnet Attacks," www.kaggle.com, 2020. https://www.kaggle.com/datasets/mkashifn/nbaiot-dataset
[34] K. Yang, Z. Huang, X. Wang, and X. Li, "A Blind Spectrum Sensing Method Based on Deep Learning," Sensors, vol. 19, no. 10, p. 2270, May 2019, doi: https://doi.org/10.3390/s19102270.
[35] V. Suárez-Paniagua and I. Segura-Bedmar, "Evaluation of pooling operations in convolutional architectures for drug-drug interaction extraction," BMC Bioinformatics, vol. 19, no. S8, Jun. 2018, doi: https://doi.org/10.1186/s12859-018-2195-1.
[36] F. Gers, N. Schraudolph, and J. Schmidhuber, "Learning Precise Timing with LSTM Recurrent Networks," Journal of Machine Learning Research, vol. 3, pp. 115–143, 2002.
[37] Taghanaki, Saeid Asgari et al., "Select, Attend, and Transfer: Light, Learnable Skip Connections," in Machine Learning in Medical Imaging, H. Suk, M. Liu, P. Yan, and C. Lian, Eds., Cham: Springer International Publishing, 2019, pp. 417–425.
[38] E. Yilmaz and H. Demirhan, "Weighted kappa measures for ordinal multi-class classification performance," Applied Soft Computing, vol. 134, p. 110020, Jan. 2023, doi: https://doi.org/10.1016/j.asoc.2023.110020.
[39] G. M. Foody, "Explaining the unsuitability of the kappa coefficient in the assessment and comparison of the accuracy of thematic maps obtained by image classification," Remote Sensing of Environment, vol. 239, p. 111630, Mar. 2020, doi: https://doi.org/10.1016/j.rse.2019.111630.
[40] Chicco and G. Jurman, "The Matthews correlation coefficient (MCC) should replace the ROC AUC as the standard metric for assessing binary classification," BioData Mining, vol. 16, no. 1, Feb. 2023, doi: https://doi.org/10.1186/s13040-023-00322-4.
[41] M. Catillo, A. Pecchia, and U. Villano, "A Deep Learning Method for Lightweight and Cross-Device IoT Botnet Detection," Applied Sciences, vol. 13, no. 2, p. 837, Jan. 2023, doi: https://doi.org/10.3390/app13020837.



Mohammed S. Alshehri received the B.S. degree in Computer Science from the King Khalid University, Abha, KSA, in 2010, and received M.S degree in Computer Science from the University of Colorado, Denver, USA, in 2014. Mohammed received the Ph.D. degree in Computer Science from the University of Arkansas, Fayetteville, USA, in 2021. Mohammed also received a graduate certificate in Cybersecurity from the University of Arkansas, Fayetteville, USA, in 2021. Mohammed joined, moderator volunteer, IEEE CS DVP-SYP Virtual Conference in 2021. Mohammed has gained multiple professional certificates during his graduate life, such as: Security+, Network+, and CISM. Mohammed's areas of interest contains Cyber Security, Computer Networks, Cloud-Fog-Edge Computing, IoT, Blockchain, Machine Learning, and Deep Learning. Mohammed is currently joining Najran University, Najran, KSA as assistant professor in the department of Computer Science.

Jawad Ahmad is an experienced researcher with more than ten years of cutting-edge research and teaching experience in prestigious institutes, including Edinburgh Napier University, U.K., Glasgow Caledonian University, U.K., Hongik University, South Korea, and HITEC University, Taxila, Pakistan. He has co-authored more than 150 research articles, in international journals and peer-reviewed international conference proceedings. He has taught various courses both at Undergraduate (UG) and Postgraduate (PG) levels during his career. He regularly organizes timely special sessions and workshops for several flagship IEEE conferences. His research interests include cybersecurity, machine learning and deep learning, chaos theory and multimedia encryption.

Mimonah Al Qathrady. Mimonah Al Qathrady received her Ph.D. in Computer Engineering (CE) in 2020 from the University of Florida (UF), Gainesville, USA. She obtained her MSc in CE in 2013 from UF, and her Bachelor of Information Systems from King Khalid University, Abha, Saudi Arabia, 2007. Since 2020, she has been working as an assistant professor in Najran University. Her research interests include IoT data analysis and modeling, and applications in mobility modeling, encounter and infection tracing, and security fields; as well as build- ing data-driven systems incorporating machine and deep learning. She worked on several projects, such as MobiBench, as well as being the lead investigator in a multitude of projects, such as i-Hospital and AI in IoT. She received granter group fellowship award. She was the coordinator of NOMADS lab at UF, and she is a member of ACM and IEEE.

Sultan Almakdi received the B.S. degree in computer science from King Khalid University, Abha, Saudi Arabia, in 2010, the M.S. degree in computer science from the University of Colorado Denver, Denver, USA, in 2014, and the Ph.D. degree in computer science from the University of Arkansas, Fayettiville, USA, in 2020. He is currently working as an Assistant Professor with the Department of Computer Science and Infor- mation Systems, Najran University, Saudi Arabia. His research interests include cloud security, fog security, edge computing security, the IoT security, and computer security. He received a Graduate Certificate in cybersecurity from the University of Arkansas, in 2020.

William J. Buchanan is currently a Professor with the School of Computing, Edinburgh Napier University. He was awarded an OBE in the Queen’s Birthday awards, in June 2017. He also leads the Centre for Distributed Computing, Networks, and Security and The Cyber Academy, and works in the areas of security, cloud security, Web-based infrastructures, e-crime, cryptography, triage, intrusion detection systems, digital forensics, mobile computing, agent-based systems, and security risk. He has one of the most extensive academic sites in the World and is involved in many areas of novel research and teaching in computing. He has published more than 27 academic books, and more than 250 academic research articles, along with several awards for excellence in knowledge transfer, and for teaching. He was named as one of the Top 100 people for Technology in Scotland from 2012 to 2017. Recently, he was included in the FutureScot Top 50 Scottish Tech People Who Are Changing The World. He is a Fellow of the BCS and the IET.