

# The incorporation of stacked long short-term memory into intrusion detection systems for botnet attack classification

Ahmad Heryanto<sup>1,2</sup>, Deris Stiawan<sup>2</sup>, Adi Hermansyah<sup>2</sup>, Rici Firnando<sup>2</sup>, Hanna Pertiwi<sup>2</sup>,  
Mohd Yazid Idris<sup>3</sup>, Rahmat Budiarto<sup>4</sup>

<sup>1</sup>Faculty of Engineering Universitas Sriwijaya, Palembang, Indonesia

<sup>2</sup>Faculty of Computer Science, Universitas Sriwijaya, Palembang, Indonesia

<sup>3</sup>Department of Computer Science, University of Technolog, Johor, Malaysia

<sup>4</sup>Faculty of Computer Science, Albaha University, Al Aqiq, Saudi Arabia

## Article Info

### Article history:

Received Jun 13, 2024

Revised Jul 10, 2024

Accepted Jul 12, 2024

### Keywords:

Botnet

Cyber attack

Long short-term memory

Principal component analysis

Synthetic minority

oversampling technique

## ABSTRACT

Botnets are a common cyber-attack method on the internet, causing infrastructure damage, data theft, and malware distribution. The continuous evolution and adaptation to enhanced defense tactics make botnets a strong and difficult threat to combat. In light of this, the study's main objective was to find out how well techniques like principal component analysis (PCA), synthetic minority oversampling technique (SMOTE), and long short-term memory (LSTM) can help find botnet attacks. PCA shows the ability to reduce the feature dimensions in network data, allowing for a more efficient and effective representation of the patterns contained. The SMOTE addresses class imbalances in the dataset, enhancing the model's ability to recognize suspicious activity. Furthermore, LSTM classifies sequential data, understanding complex network patterns and behaviors often used by botnets. The combination of these three methods provided a substantial improvement in detecting suspicious botnet activities. We also evaluated the effectiveness using performance metrics such as accuracy, precision, recall, and F1-score. The results showed an accuracy of 96.77%, precision of 88.95%, recall of 88.58%, and F1-score of 88.64%, indicating that the proposed model was reliable in detecting botnet traffic compared to other deep learning models. Furthermore, LSTM can classify sequential data, understanding complex network patterns and behaviors often used by botnets.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



## Corresponding Author:

Deris Stiawan

Faculty of Computer Science, Universitas Sriwijaya

St. Lintas timur km.32 Indralay Ogan Ilir, Palembang, Indonesia

Email: deris@unsri.ac.id

## 1. INTRODUCTION

A botnet is a group of computers hacked and controlled through the internet using malware [1]–[3] by hackers without the knowledge of their owners. Specifically, hackers use botnets for various malicious activities like distributed denial of service (DDoS) attacks, data theft, email infiltration, and malware distribution [4], [5]. It comprises three main components. The first is the botmaster, an individual or entity that controls, manages, and organizes the infected network's operations. The command and control (C&C) server is the second component, which functions as a central server to record or control the computers infected by malware. This server facilitates communication between the botmaster and the bots dispersed throughout the network. Finally, bots refer to computers or devices infected with malware that are capable of

executing instructions given by the C&C server. The botmaster, C&C server, and bots' network communication system allow them to control and direct the bots remotely [4], [5].

Machine learning methods such as naive Bayesian, Bayesian networks, and decision trees performed botnet attack detection in previous investigations [6], [7]. The results demonstrated the efficient implementation of supervised learning algorithms for attack classification. Nearest neighbors, naïve Bayesian, support vector machine (SVM), artificial neural networks (ANN), and Gaussian-based classifiers were used in other studies to look at network traffic related to host and flow features [8]–[11]. Meanwhile, some scientific studies have identified weaknesses in applying machine learning methods for detecting botnets, particularly in attack classification. The rapid evolution and variety of botnet attacks often complicate the selection of features used in model training, leading to the main weaknesses. The development of increasingly complex attacks leads to a diverse range of features that may indicate botnets, adding to the complexity of classification. To address these weaknesses, certain studies have proposed the application of deep learning methods such as long short-term memory (LSTM) for detecting cyber attacks [12]–[15].

LSTM is an ANN architecture capable of effectively understanding and modeling temporal patterns of sequential data [13], [16]. It excels at processing long, sequential data by efficiently retaining and managing both long-term and short-term information. These features make LSTM highly suitable for tackling botnet detection issues, particularly where network traffic data tends to be sequential and dynamic [17]. The system can learn suspicious behavior patterns from network traffic, including unusual communication between devices, abnormal activity on rarely used ports, and unnatural coordination. Additionally, it can address class imbalance issues where the number of positive samples (botnet attack) is usually smaller than the number of negative samples (benign activity). This fact shows that the use of LSTM for botnet detection is an effective step in building a security system that is adaptive and responsive to continuously evolving cyber attacks. By using artificial intelligence and sequential pattern modeling, LSTM enhances the detection and response capabilities against botnet attacks efficiently and effectively.

By integrating dimension reduction techniques like principal component analysis (PCA) and class imbalance handling, such as the synthetic minority oversampling technique (SMOTE), we can optimize the use of LSTM in botnet attack detection [18]. Several studies have shown that LSTM has some problems, such as the chance of overfitting data with many features and issues with class imbalance [19]–[21]. Therefore, integrating PCA can help reduce the dimensions of complex features and mitigate the risk of overfitting. The use of SMOTE is also capable of improving the balance between classes in the dataset, allowing LSTM to learn better patterns related to botnet attacks [18], [22]–[25]. We anticipate that this combination will enable LSTM to generate more informative feature representations, surmount botnet attack detection challenges, and boost the accuracy and reliability of the deployed detection system. Thus, this paper proposes the stacked LSTM incorporation into intrusion detection systems (IDS) to address the accuracy issue in detecting botnet malware.

## 2. RELATED STUDIES

Researchers have conducted several studies on the use of machine and deep learning to detect botnet attacks and enhance network security. These efforts involve the development of various methods to create models that can detect attacks with optimal accuracy and speed. Table 1 summarizes some important works on the existing detection and classification methods.

Table 1. Related studies

Author	Method	Dataset	Result (%)
Jagadeesan and Amutha [26]	ESVNN	CTU-13	86.84
Bijalwan <i>et al.</i> [27]	Ensemble	ISCX	93.37.
Hoang and Nguyen [28]	Naïve Bayes	Domain name and T1 training set	Naïve Bayes = 86.50
	C4.5		C4.5 = 90.10
	KNN		KNN = 90.30
	RF		RF = 90.80
Kudugunta and Ferrara [29]	LSTM	Cresci	96.33
Saurabh <i>et al.</i> [30]	Stacked LSTM	UNSW NB15 and BoTIoT datasets	99.99

Jagadeesan and Amutha [26] introduced a new detection model known as the enhanced support vector neural network (ESVNN). To improve classification accuracy, we enhance this model by selecting appropriate features from the dataset's data traffic flow. Observing constant response packets allows us to identify features such as bot response packet ratio, initial packet length, ratio, and small packets. We then use these features as inputs for the proposed ESVNN classifier or prediction model, and apply the artificial flora (AF) algorithm to improve the performance of support vector neural network (SVNN). The results show that

this botnet detection model achieves better accuracy and F-measure, showing a precision of 0.8709, recall of 0.8636, accuracy of 0.8684, and F-score of 0.8669.

Bijalwan *et al.* [27] used the ISCX dataset to train and test ensemble classifier algorithms to identify bots. The results showed that using the voting method of the ensemble classifier increased accuracy to 96.41% from 93.37%. We also conducted investigations to test multiple machine learning algorithms on a domain dataset. The results showed accuracy for various models, where naïve Bayes achieved 86.50%, C4.5 had 90.10%, k-nearest neighbors (KNN) reached 90.30%, and random forest (RF) was 90.80% [28]. Kudugunta and Ferrara [29] reported that they used the LSTM algorithm on the Cresci dataset to detect bots. racted contextual features from user metadata as additional inputs for the LSTM network, showing an accuracy of 96.33%. On the other hand, Saurabh *et al.* [30] created a network intrusion detection systems (NIDS) model using stacked and bidirectional LSTM variants, as well as the UNSW\_NB15 and BoT-internet of things (IoT) datasets. Their model was 97% accurate.

## 2.1. Botnet

A botnet is a network of computers infected by malware and controlled remotely by an unauthorized individual, known as a botmaster [31]. Botnets pose a severe threat to internet security because they can obtain data for illegal activities like DDoS attacks, spamming, data theft, and malware distribution. The ability to control botnets through a C&C infrastructure sets them apart from other types of malware [32], [33]. Due to differences in time zones, languages, and applicable laws, cyber security authorities find it challenging to trace and handle the distributed infected hosts of botnets worldwide. The flexibility, scalability, and evasion methods employed by botnets present a significant threat that necessitates international cooperation and coordinated efforts to counter. Moreover, Figure 1 shows an illustration of a botnet lifecycle.

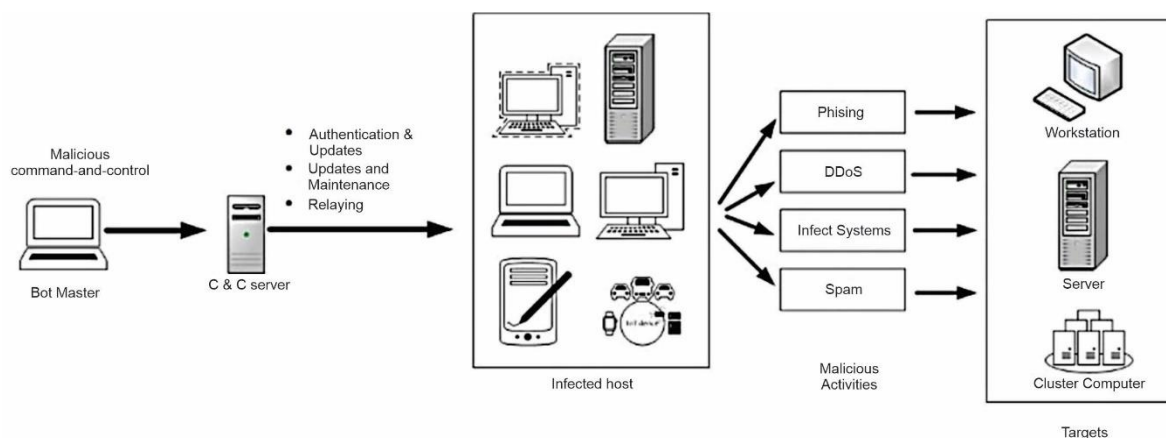


Figure 1. Botnet lifecycle

The botnet has several possible C&C communication topologies that affect how the botmaster communicates with infected hosts. These include the centralized C&C model, which has a central point for managing and relaying messages between bots and the botmaster. The model is relatively easy to implement and customize, but due to the identifiable central point, it is vulnerable to detection as an attack. Furthermore, there is the peer-to-peer (P2P)-based C&C model, which is an evolution from the centralized model. An unstructured C&C model is also available, where bots do not actively contact other bots or the botmaster. This model monitors each incoming connection from the botmaster and randomly forwards encrypted messages to other bots. Based on the method, the botmaster can remain effectively hidden, although greater effort is required in monitoring and managing incoming communications. Understanding the various suitable C&C models becomes crucial in combating botnets to facilitate the development of effective detection and protection strategies [34].

## 2.2. Long short-term memory

LSTM as shown in Figure 2, is a variant of recurrent neural network (RNN) designed to address the issues of vanishing and exploding gradients in RNN [35]. It uses hidden units that function to store input over long periods [36] and comprises three gates, namely the input, the forget, and the output. Specifically, the forget gate is responsible for discarding old memories, the input gate receives new data by determining memory updates. The output gate combines short-term and long-term memory to produce the current memory output [35], [37]. The equations for the LSTM gates are expressed as in (1) to (6) [37]. For the input gate  $i_t$ , the

equation is detailed in (1). Where  $x_t$  represents the input value at time  $t$ ,  $W$  is the weight matrix at each gate,  $h_{t-1}$  is the hidden state at the previous time step, and  $b$  is the bias value for each gate. For the forget gate  $f_t$ , the equation is detailed in (2). For the output gate  $o_t$ , the equation is detailed in (3). Where  $\tilde{c}_t$  is the cell candidate computed at each time step to update the memory cell, the equation is detailed in (4). Therefore, from these (1) to (4), the current cell memory ( $c_t$ ), and hidden state ( $h_t$ ) can be detailed in (5) and (6).

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{1}$$

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3}$$

$$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{4}$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \tag{5}$$

$$h_t = o_t * \tanh c_t \tag{6}$$

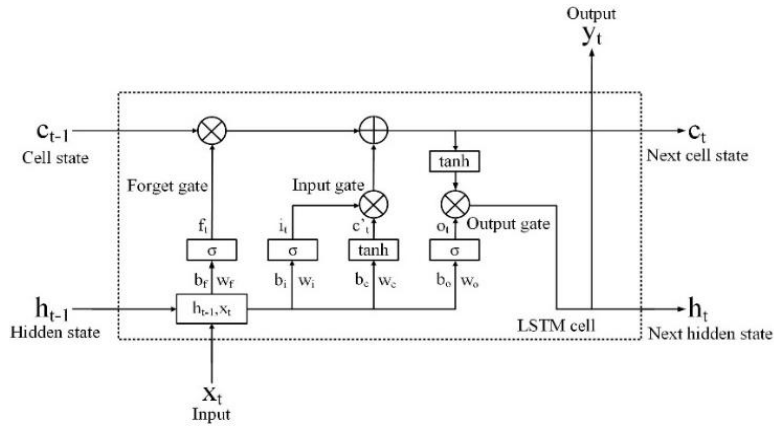


Figure 2. LSTM architecture [21], [38], [39]

### 3. METHOD

#### 3.1. Proposed models

In this study, the models were tested using the CSE-CIC-IDS2018 dataset, which was filtered to extract botnet attack traffic and benign. The SMOTE method was applied to address the imbalance in the amount of data. Furthermore, the data was divided into training and testing datasets with a certain ratio after completing the preprocessing process. After the preparation stage, the study proceeded with the construction and testing of the models by processing data using LSTM, and their performance was evaluated based on specific parameters. The final stage included analyzing the results of the previous evaluations and the conducted process, as shown in Figure 3.

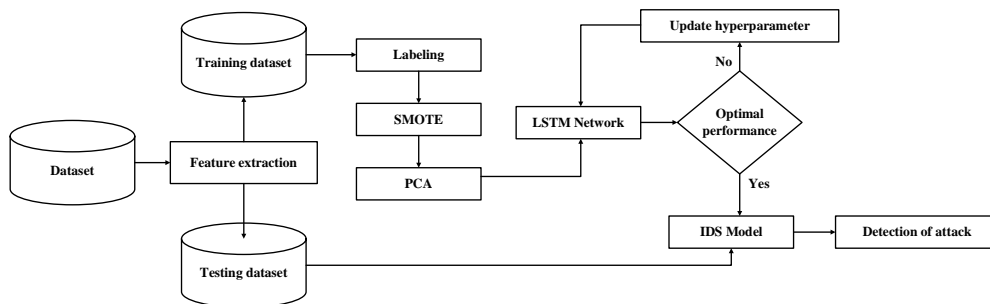


Figure 3. Study method

**3.2. Dataset**

This study uses the CSE-CIC-IDS 2018 dataset from the University of New Brunswick (UNB) or the Canadian Institute for cyber security. The dataset comprises seven categories, namely brute force, web attack, DoS attack, DDoS attack, botnet, infiltration, and benign. The attack infrastructure consisted of 50 machines, while the victims included 5 departments with 420 machines and 30 servers. This dataset comprises each machine network traffic and system logs [40], including a list of attack types and their duration. All attack scenarios were collected in Pcaps files and event logs from each machine. This data extracted 83 network traffic features, as shown in Table 2. To view the details of the feature list in the dataset, please refer to CSE-CIC-IDS2018 on web [40].

Table 2. Network traffic features dataset CSE-CIC-IDS 2018 [25], [41]

Feature Number	Description
01 to 04	Fundamental characteristics of network connections
05 to 16	Characteristics of network packets
17 to 22	Attributes of network flows
23 to 45	Statistics of network flows
46 to 63	Traffic features related to content
64 to 67	Characteristics of network subflows
68 to 79	General traffic features
80 to 83	Fundamental features of network connections

**3.3. Data preprocessing**

Preprocessing stage, including data cleaning and transformation, is carried out to prepare data effectively for model training and testing contexts. Specifically, data cleaning is performed to ensure data integrity, consistency, and relevance within a dataset. This process includes identifying and correcting errors as well as removing meaningless data that could impact the analysis and models. The initial step in data cleaning is removing duplicates, where entries with identical values are identified and eliminated to maintain model accuracy. Additionally, inaccurate or irrelevant data (noisy data) is removed, focusing on identifying entries that do not fit the dataset structure. Data cleaning serves as a crucial step to ensure the data quality used in analysis and producing accurate results. This is followed by data transformation, which includes two important steps, namely data encoding and normalization.

**3.4. Classification model**

Classification model predicts the class of data, categorizing attack as binary or multi-class to distinguish between benign and malicious network traffic in the context of intrusion detection system (IDS). This complexity puts pressure on algorithms in terms of computational power and time. Each dataset is evaluated and categorized as benign or botnet in the classification process, which is used to identify unusual patterns, detect anomalies, and recognize misuse. In this study, the architecture of the model is stacked LSTM, with the design presented in Figure 4. Specifically, LSTM layers are stacked to ensure that the lowest and highest hidden layers are fully connected in a feedforward relationship.

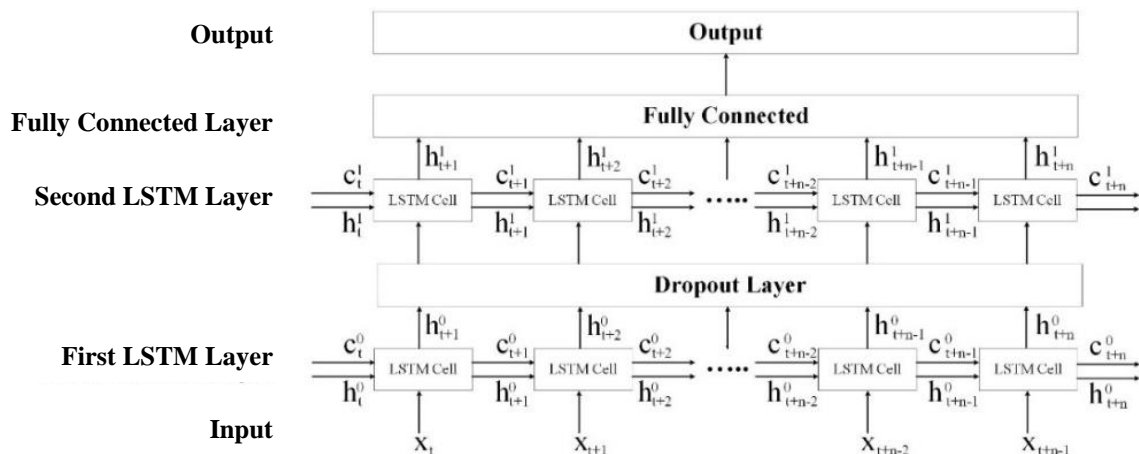


Figure 4. Stacked LSTM architecture design

Stacked LSTM is designed based on the principle that multiple layers of nonlinear mapping between input and output are used for learning hierarchical systems. In Figure 4, the output from the hidden layers propagates forward, serving as one of the inputs to the next LSTM layer. Specifically, each layer receives input from the previous and produces output that becomes the input for the next layer. Through this process, the model can extract increasingly complex feature representations from sequential data. The main advantage of stacked LSTM is the ability to handle the processing of long and complex sequence data effectively. Furthermore, it has the capability to learn more abstract patterns, perform better prediction, classification, or sequential data analysis. The general formula for stacked LSTM is not significantly different from a single LSTM but considers the relationships between each stacked layer. The formula can be represented in terms of forget gate, input gate, output gate, candidate cell state, cell state, and hidden state. The step-by-step process is as follows, the input at time  $t(x_t)$  is fed into the first LSTM layer. Moreover, the 3 main gates that control the flow of information gate in each layer include the forget, input, and output. In (8), the forget gate determines which information needs to be discarded from the previous cell state ( $c_{t-1}$ ). This is performed by multiplying the previous cell state by the output of a sigmoid function using weights  $W_f$ ,  $U_f$ , and bias  $b_f$ , which applied to input  $x_t$  as well as the previous hidden state ( $h_{t-1}$ ) (8). In (9), the input gate determines the new information stored in the cell state. This is performed by computing the sigmoid function of weights  $W_i$ ,  $U_i$ , and bias  $b_i$ , applied to the input  $x_t$  and the previous hidden state (9). In (10), the candidate cell value ( $\tilde{c}_t^{(l)}$ ) is calculated using the tanh function, which combines the input  $x_t$  and the previous hidden state with weights  $W_c$ ,  $U_c$ , and bias  $b_c$ . In (11), the cell state is updated by combining the previous cell state modified by the forget gate with the candidate cell value modified by the input gate. In (12), the output gate determines the output of the hidden state based on the updated cell state. This is carried out by computing the sigmoid function of weights  $W_o$ ,  $U_o$ , and bias  $b_o$ , in the input  $x_t$  and the previous hidden state. In (13), the hidden state ( $h_t$ ) is obtained by multiplying the output gate with the tanh function of the updated cell state.

After the process in the first layer is completed, the obtained hidden state ( $h_t$ ) is used as the input ( $x_t$ ) for the next LSTM layer, which is repeated for others. The lower layers capture simpler and more local patterns, while the upper layers capture more complex and global patterns. Through these stages, stacked LSTM can handle long and complex data sequences more effectively compared to single LSTM. Additionally, stacked allows each layer to process and extract higher-level features, enhancing the model's ability to understand and predict sequential data.

$$f_t^{(l)} = \sigma(W_f^{(l)} x_t + U_f^{(l)} h_{t-1}^{(l)} + b_f^{(l)}) \quad (8)$$

$$i_t^{(l)} = \sigma(W_i^{(l)} x_t + U_i^{(l)} h_{t-1}^{(l)} + b_i^{(l)}) \quad (9)$$

$$\tilde{c}_t^{(l)} = \tanh(W_c^{(l)} x_t + U_c^{(l)} h_{t-1}^{(l)} + b_c^{(l)}) \quad (10)$$

$$c_t^{(l)} = f_t^{(l)} * c_{t-1}^{(l)} + i_t^{(l)} * \tilde{c}_t^{(l)} \quad (11)$$

$$o_t^{(l)} = \sigma(W_o^{(l)} x_t + U_o^{(l)} h_{t-1}^{(l)} + b_o^{(l)}) \quad (12)$$

$$h_t^{(l)} = o_t^{(l)} * \tanh(c_t^{(l)}) \quad (13)$$

## 4. EXPERIMENTS

### 4.1. Experiment setup

This study used the CSE-CIC-IDS 2018 dataset shown in Figure 5(a), which includes various network attacks, including botnet. The primary focus of data processing was to handle missing values, feature normalization, and encoding categorical variables. Relevant features were selected for botnet detection using PCA and the dataset was split into training and testing sets, with a standard ratio of 80:20. This division maintains the distribution of botnet attack instances to avoid class imbalance.

The botnet dataset was obtained from CSE-CIC-IDS2018, consisting of 286,191 data samples shown in Figure 5(b). The benign data samples were selected from the dataset and combined with botnet data samples. This combination was performed in a balanced proportion between benign and botnet data to ensure that the resulting dataset could be effectively used for training and evaluating botnet detection models. Additionally, the study applied the SMOTE to increase the number of samples in the minority class, which is the botnet data. This process ensures balancing the proportion between the two classes to avoid an imbalance that could affect the model's performance in detecting botnet attack. The experiments were conducted under

several scenarios and on hardware with specifications detailed in Table 3. The first scenario includes selecting input features to be used, while other scenarios were designed based on the combination of hyperparameter configurations for stacked LSTM.

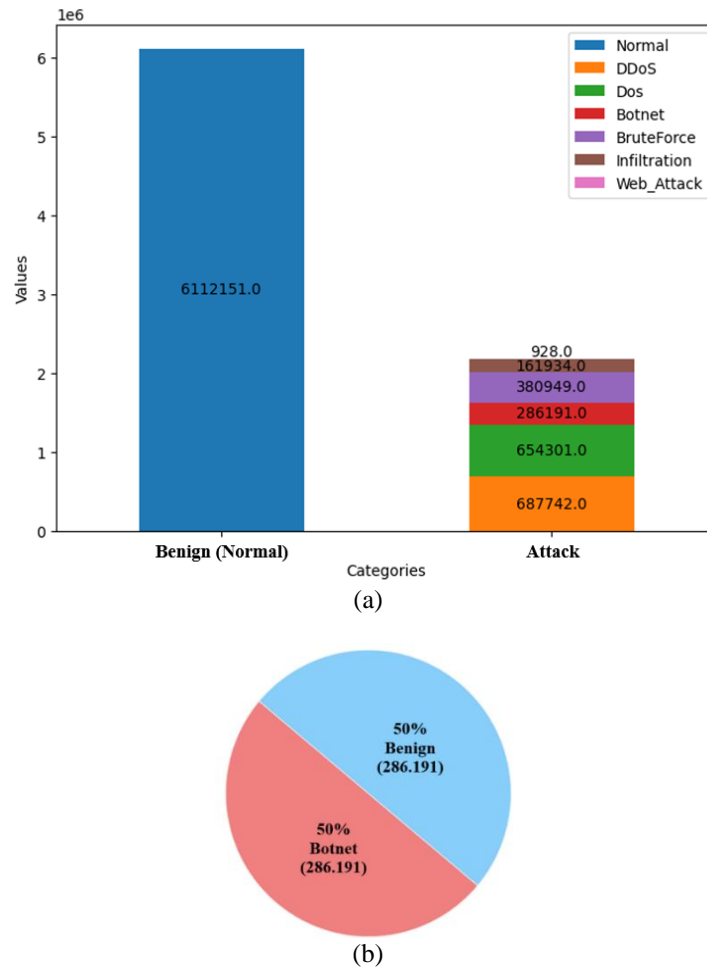


Figure 5. Dataset overview and statistics: (a) distribution of class data in CSE-CIC-IDS2018 and (b) comparison of benign and botnet

Table 3. Hardware specifications for processing stacked LSTM

Component	Specification
CPU	16 Core
Memory	32 GB
Hardisk	1 TB
Networks	1 Gbps

The hardware infrastructure described in this study ensured optimal performance for processing stacked LSTM models, using high computational capabilities and fast data access speeds. Python was used for data processing tasks, model training, and evaluation. Data manipulation and analysis were facilitated using Pandas, primarily for dataset handling, while NumPy performed numerical operations and array handling. Scikit-learn assisted in data preprocessing tasks, including scaling, dataset splitting, and evaluation metrics. Data visualization and graph construction were based on Matplotlib and Seaborn libraries, while TensorFlow and Keras were used for building, training, and evaluating stacked LSTM models. Interactive development and visualization of results were supported by Jupyter Notebook, while Anaconda simplifies the management of the Python environment and its dependencies. This integrated method ensures the efficient development and evaluation of stacked LSTM models for various applications.

#### 4.2. Tuning hyperparameter

In the design of stacked LSTM network, this study explores various configurations to determine the most optimal model. In Table 4, several LSTM parameters were used such as nodes, dropout, batch size, learning rate, and epochs. Methods such as grid search were used for hyperparameter optimization to determine the best combination for stacked LSTM model with the highest performance in botnet detection tasks.

Table 4. Hyperparameter configuration

Hyperparameter	Value
LSTM Node	32, 64, 128, 256, 512
Dropout	0.1, 0.2, 0.3, 0.4, 0.5
Batch Size	128, 256, 512, 1,024, 2,048
Learning Rate	0.01, 0.001, 0.0001, 0.00001, 0.000001
Loss function	Binary cross entropy
Activation Function	Sigmoid
Number of epochs	10, 20, 30, 40, 50

Testing was conducted by inputting parameters into the model to be tested to determine the best hyperparameter, which would be used as the main parameters. From the experiments with tuning hyperparameters shown in Table 5, it was observed that increasing the number of node units in the neural network (LSTM) model leads to decreased prediction accuracy. The results suggested that increasing the model complexity by adding more node units did not improve performance, showing the potential to cause decreased accuracy and overfitting on the training data. This showed the need for adjusting and managing the model complexity to avoid excessively capturing noise signals from irrelevant training data. Subsequently, testing was conducted on the dropout parameter to examine its impact on model performance, with the results listed in Table 6.

Table 5. Tuning hyperparameters of number of nodes

Unit, banodes	Dropout	Activation Function	Learning rate	Batch size	Epoch	Accuracy (%)
32	0.1	Sigmoid	0.01	128	10	94.34
64	0.1	Sigmoid	0.01	128	10	96.49
128	0.1	Sigmoid	0.01	128	10	96.29
256	0.1	Sigmoid	0.01	128	10	95.82
512	0.1	Sigmoid	0.01	128	10	95.52

Table 6. Tuning hyperparameter dropout

Unit nodes	Dropout	Function activation	Learning rate	Batch size	Epoch	Accuracy (%)
32	0.1	Sigmoid	0.01	128	10	91.34
32	0.2	Sigmoid	0.01	128	10	92.62
32	0.3	Sigmoid	0.01	128	10	94.34
32	0.4	Sigmoid	0.01	128	10	94.12
32	0.5	Sigmoid	0.01	128	10	94.21

The impact of dropout on model performance requires consideration, as a regularization method used to prevent overfitting by randomly deactivating some neurons during the training process. In this dataset, dropout was increased from 0.1 to 0.5 alongside a rise in the number of node units. The results showed that higher dropout led to increased accuracy, where the highest value was achieved in the third experiment with a dropout of 0.3 (94.34%). This showed that higher dropout usage could help reduce overfitting and improve the model generalization.

In Table 7, several experiments were conducted with various learning rate values to determine the optimal learning rate for the neural network model. Meanwhile, other parameters, such as the number of node units, dropout rate, activation function, batch size, and number of epochs, were kept constant. The results showed significant variation in model accuracy. In the first experiment, with a learning rate of 0.01, the model achieved the highest accuracy of 94.34%. However, when the learning rate was reduced to 0.001 in the second experiment, the accuracy significantly dropped to 89.65%. Further reduction in the learning rate to 0.0001 and 0.00001 led to model accuracies of 61.62% and 69.00%, respectively, showing that a learning rate excessively small did not sufficiently optimize weight updates in the network. At the lowest learning rate



of 0.000001, the model accuracy further decreased to 55.10%, emphasizing that a very small learning rate was ineffective in training the model. A learning rate that is too high can cause the model to be unstable and fail to achieve optimal convergence, while an excessively high value can lead to slow convergence and failure to effectively learn patterns in the data. The experimental results showed that a learning rate of 0.01 provided the best performance in this scenario, making it the optimal value for training the neural network model.

Table 7. Tuning hyperparameter learning rate

Unit nodes	Dropout	Function activation	Learning rate	Batch size	Epoch	Accuracy (%)
32	0.3	Sigmoid	0.01	128	10	94.34
32	0.3	Sigmoid	0.001	128	10	89.65
32	0.3	Sigmoid	0.0001	128	10	61.62
32	0.3	Sigmoid	0.00001	128	10	69.00
32	0.3	Sigmoid	0.000001	128	10	55.10

In Table 8, an experiment was conducted to test the impact of varying batch sizes on the neural network model, while keeping other parameters constant to determine the most suitable batch size. The results showed that a batch size of 128 provided the highest accuracy at 94.34%. Increasing the batch size to 256 caused a significant drop in accuracy to 89.58%, and a further decrease to 88.23% was observed at 512. At a batch size of 1,024, accuracy slightly improved to 89.70%, while size 2,048 had 90.43%, showing improvement over the previous batch size. These results showed that a smaller batch size of 128 allowed the model to learn more effectively, updating weights more frequently and in greater detail. Larger batch sizes reduced the frequency of weight updates, leading to poorer generalization from the training data. Therefore, a batch size of 128 was the most optimal selection for the neural network model with this configuration. The optimal parameters obtained from the tuning results were used as the main hyperparameters, as shown in Table 9.

Table 8. Tuning hyperparameter batch size

Unit nodes	Dropout	Function activation	Learning rate	Batch size	Epoch	Accuracy (%)
32	0.3	Sigmoid	0.01	128	10	94.34
32	0.3	Sigmoid	0.01	256	10	89.58
32	0.3	Sigmoid	0.01	512	10	88.23
32	0.3	Sigmoid	0.01	1,024	10	89.70
32	0.3	Sigmoid	0.01	2,048	10	90.43

Table 9. Optimal hyperparameter

Parameter	Value
Number of unit nodes	32
Activation function of stacked LSTM layer	Sigmoid
Batch size	128
Loss function	Binary crossentropy
Learning rate	0.01
Optimizer function	Sigmoid
Dropout	0.3
Epoch	10

### 4.3. Discussions

After obtaining optimal hyperparameters for the LSTM and stacked LSTM models, the next step was the implementation on the testing data for performance evaluation. In this study, the main evaluation metrics used are: accuracy, precision, recall, and F1-score, to gain a comprehensive understanding of the model effectiveness in handling the dataset. The results provided valuable insights into assessing whether the developed model performs adequately for the intended application purposes or requires further adjustments. Therefore, the use of LSTM and stacked LSTM after tuning ideal hyperparameters would provide deep insights into the model capability to handle classification tasks on the given dataset.

The training and testing performances in Figure 6(a) showed stable accuracy values at epoch 10, indicating that the model had reached a sufficient convergence point at that stage. Convergence suggests that the model had learned well from the training data and could consistently make predictions on the testing data. At this point, the incremental increase in accuracy between subsequent epochs is no longer significant,

showing that additional training iterations may not significantly improve model performance. Although the experiment was set to run for 50 epochs, achieving stabilization by epoch 10 suggested that further training iterations might not provide significant benefits. Therefore, stopping the training at epoch 10 could be considered sufficient, saving time, and computational resources needed for the training process. The training and testing graphs in Figure 6(b) showed stable loss values at epoch 10, despite the experiment running up to 50 epochs. This suggested that the model might have reached convergence relatively early at an optimal loss value, specifically at epoch 10. The results also showed that the model had effectively adjusted weights to the training data and achieved an optimal learning rate. At epoch 10th, the stable results also suggested that additional epochs did not cause a significant improvement in model performance.

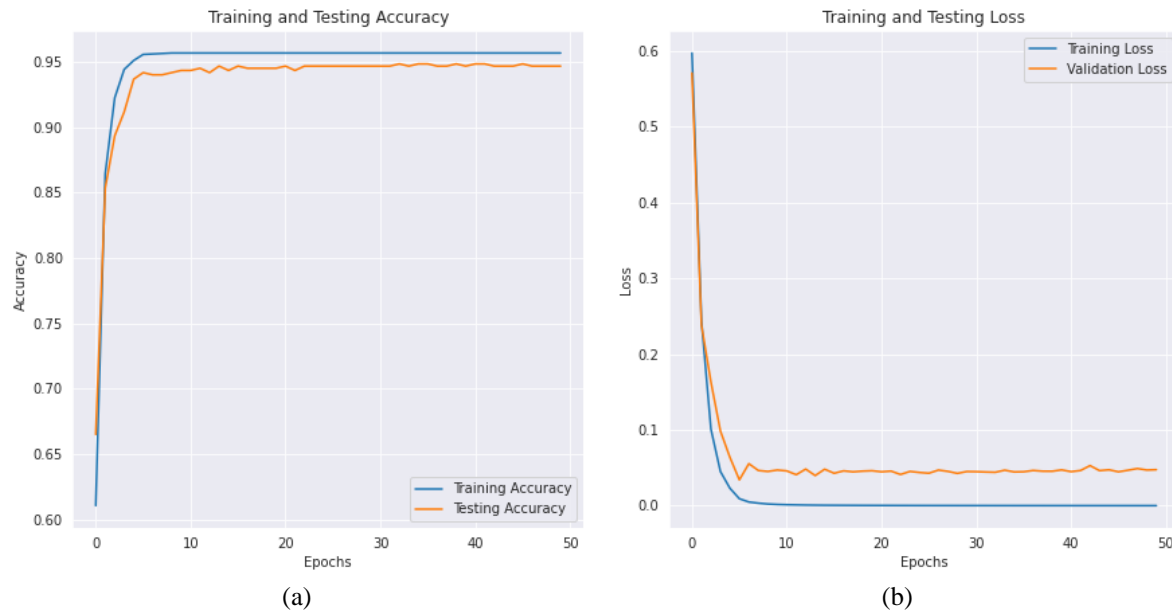


Figure 6. Performance during training and testing phases: (a) accuracy graph and (b) loss graph

Based on the data obtained from tuning hyperparameters of a single LSTM, this study developed several models, namely PCA+LSTM, stacked LSTM, and PCA+Stacked LSTM. Table 10 and Figure 7 show that using different methods to construct LSTM models for classification leads to significant differences in performance and computational resources. The single LSTM model shows moderate performance with an accuracy of 94.34% but can be improved, particularly in precision and recall. Using the PCA+LSTM model slightly improves performance with an accuracy of 96.24%, although there is a need for more computational resources for execution. The stacked LSTM model shows slightly better performance in accuracy and other metrics compared to Single LSTM but has longer execution time. The PCA+Stacked LSTM model achieves the best performance with an accuracy of 96.77%, with significant improvements in precision and recall. However, PCA+Stacked LSTM model has a higher CPU load and the same execution time as stacked LSTM. When selecting the appropriate model, there is a need to consider a balance between performance and available resources.

Table 10. LSTM performance

Indicator	Single LSTM	PCA+LSTM	Stacked LSTM	PCA+ Stacked LSTM
Accuracy (%)	94.34	96.24	96.37	96.77
Precision (%)	82.29	83.68	84.29	88.95
Recall (%)	79.86	81.43	82.36	88.58
F1-Score (%)	80.48	82.05	82.83	88.64
Time (Minute)	5	6	7	7
CPU (%)	80	82	82	90

Table 11 shows the results of previous studies, indicating the variation in accuracy based on the method and dataset. According to Jagadeesan and Amutha [26], the ESVNN (M1) method applied to the

CTU-13 dataset achieved an accuracy of 86.84%, while Bijalwan *et al.* [27] the use of ensemble (M2) method on ISCX dataset obtained 93.37%. Hoang and Nguyen [28] that used the T1 training set, the naïve Bayes (M3), C4.5 (M4), KNN (M5), and RF (M6) methods achieved accuracies of 86.50%, 90.10%, 90.30%, and 90.80%, respectively. Furthermore Kudugunta and Ferrara [29], the LSTM algorithm (M7) applied to the Cresci dataset showed an accuracy of 96.33%. The proposed method (M8), namely stacked LSTM was applied to the CSE-CIC-IDS2018 dataset and achieved an accuracy of 96.77%. This significant increase showed that the use of stacked LSTM can provide better performance in detecting botnet threats compared to several previous methods. Table 11 shows the comparison of accuracy performance among various methods, as shown in Figure 8.

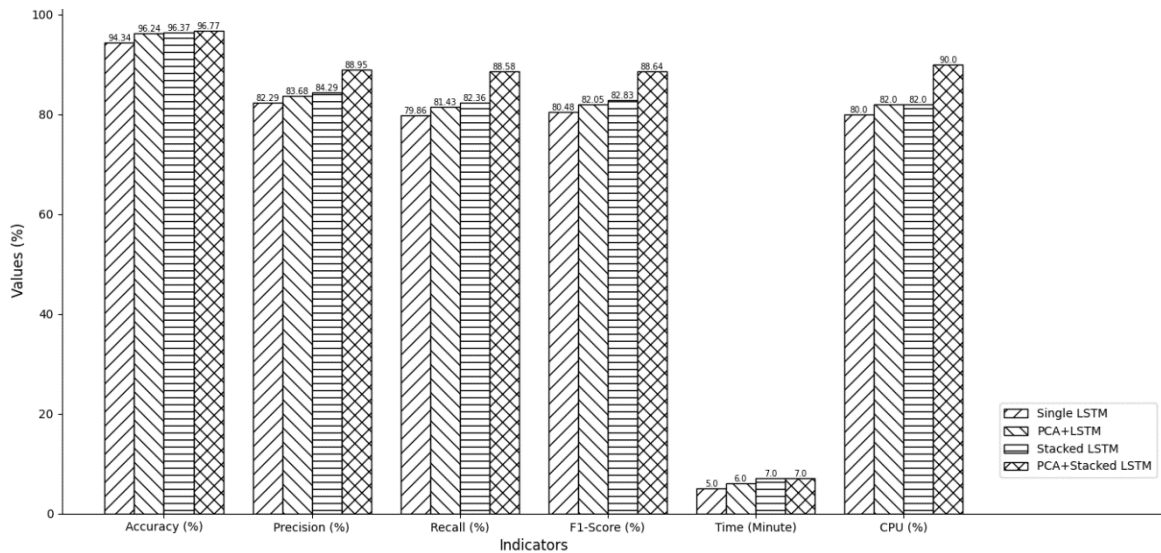


Figure 7. LSTM performance comparison

Table 11. Comparison of the accuracy between proposed and previous studies

Reference	Dataset	Method	Accuracy (%)
[26]	CTU-13	ESVNN	86.84
[27]	ISCX	Ensemble	93.37
[28]	Domain name (T1 training set)	Naïve Bayes	Naïve Bayes = 86.50
		C4.5	C4.5 = 90.10
		KNN	KNN = 90.30
		RF	RF = 90.80
[29]	Cresci	LSTM	96.33
The proposed method	CSE-CIC-IDS2018	PCA+Stacked LSTM	96.77

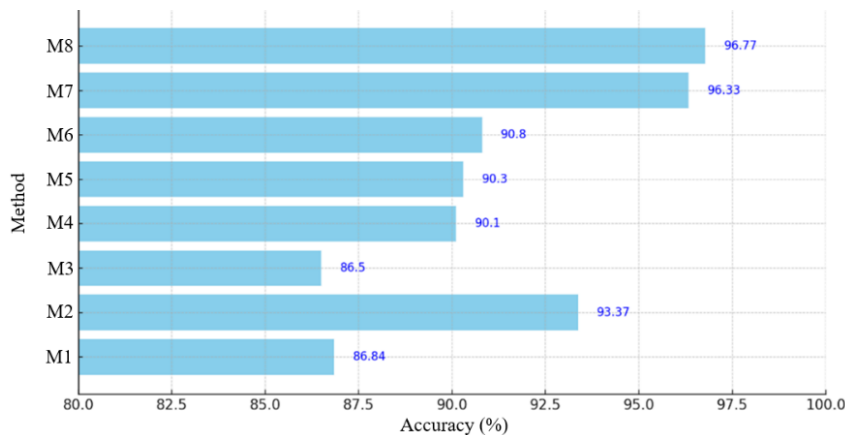


Figure 8. Comparison of accuracy across various models

## 5. CONCLUSION

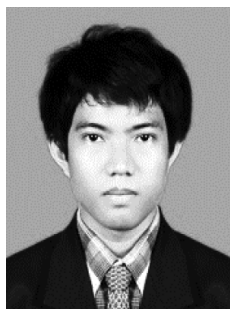
In conclusion, this study compared methods for classifying botnet attacks using the stacked LSTM model. The evaluation was based on several quantitative metrics, including accuracy, precision, recall, and F1-score. The results showed that the stacked LSTM model performed optimally compared to other evaluated methods, achieving an accuracy of 96.77%, precision of 88.95%, recall of 88.58%, and F1-score of 88.64%. This model showed consistent execution times, with an average of 7 minutes for data processing, but required approximately 90% more CPU usage. Based on these quantitative evaluation results, the stacked LSTM model was considered effective in detecting botnet attacks with good performance, although it required higher computational resources. This study could be further developed by integrating CNN-LSTM to enhance botnet detection. While LSTM handles temporal relationships, CNN captures spatial features from network data, potentially making this combination more effective in identifying complex botnet patterns. To lower overfitting and boost detection accuracy, it was thought that more research into hyperparameter optimization and the use of regularization methods like dropout on the CNN-LSTM model was necessary. Furthermore, we anticipated that integrating CNN-LSTM would significantly enhance botnet detection performance.




## REFERENCES

- [1] H. A. Sukhni, M. A. Al-Khasawneh, and F. H. Yusoff, "A systematic analysis for botnet detection using genetic algorithm," in *2021 2nd International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, 2021, pp. 63–66, doi: 10.1109/ICSCEE50312.2021.9498109.
- [2] R. Vinayakumar, K. P. Soman, P. Poornachandran, M. Alazab, and A. Jolfaei, "DBD: Deep learning DGA-based botnet detection," *Advanced Sciences and Technologies for Security Applications*, pp. 127–149, 2019, doi: 10.1007/978-3-030-13057-2\_6/COVER.
- [3] V. Kanimozhi and T. P. Jacob, "Artificial intelligence-based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing," *ICT Express*, vol. 5, no. 3, pp. 211–214, 2019, doi: 10.1016/j.icte.2019.03.003.
- [4] M. D. Hossain, H. Ochiai, F. Doudou, and Y. Kadobayashi, "SSH and FTP brute-force attacks detection in computer networks: Lstm and machine learning approaches," *2020 5th International Conference on Computer and Communication Systems, ICCCS 2020*, pp. 491–497, 2020, doi: 10.1109/ICCCS49078.2020.9118459.
- [5] S. Haq and Y. Singh, "Botnet detection using machine learning," *PDGC 2018 - 2018 5th International Conference on Parallel, Distributed and Grid Computing*, pp. 240–245, 2018, doi: 10.1109/PDGC.2018.8745912.
- [6] M. Stevanovic and Pedersen, "On the use of machine learning for identifying botnet network traffic," *Journal of Cyber Security and Mobility*, vol. 4, no. 2, pp. 1–32, Jan. 2016, doi: 10.13052/JCSM2245-1439.421.
- [7] A. Azab, M. Alazab, and M. Aiash, "Machine learning based Botnet identification traffic," in *2016 IEEE Trustcom/BigDataSE/ISPA*, 2016, pp. 1788–1794, doi: 10.1109/TrustCom.2016.0275.
- [8] I. F. Kilincer, F. Ertam, and A. Sengur, "Machine learning methods for cyber security intrusion detection: datasets and comparative study," *Computer Networks*, vol. 188, 2021, doi: 10.1016/j.comnet.2021.107840.
- [9] D. Wu, Z. Jiang, X. Xie, X. Wei, W. Yu, and R. Li, "LSTM learning with Bayesian and Gaussian processing for anomaly detection in industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5244–5253, 2020, doi: 10.1109/TII.2019.2952917.
- [10] M. S. Elsayed, N. A. L. -Khac, S. Dev, and A. D. Jurcut, "Network anomaly detection using LSTM based autoencoder," *Q2Swinet 2020 - Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, pp. 37–45, 2020, doi: 10.1145/3416013.3426457.
- [11] C. Wei, G. Xie, and Z. Diao, "A lightweight deep learning framework for botnet detecting at the IoT edge," *Computers and Security*, vol. 129, 2023, doi: 10.1016/j.cose.2023.103195.
- [12] M. Sudha, V. M. K. Reddy, W. D. Priya, S. M. Rafi, S. Subudhi, and S. Jayachitra, "Optimizing intrusion detection systems using parallel metric learning," *Computers and Electrical Engineering*, vol. 110, 2023, doi: 10.1016/j.compeleceng.2023.108869.
- [13] H. N. Bhandari, B. Rimal, N. R. Pokhrel, R. Rimal, and K. R. Dahal, "LSTM-SDM: An integrated framework of LSTM implementation for sequential data modeling," *Software Impacts*, vol. 14, 2022, doi: 10.1016/j.simpa.2022.100396.
- [14] S. Nayyar, S. Arora, and M. Singh, "Recurrent neural network based intrusion detection system," *Proceedings of the 2020 IEEE International Conference on Communication and Signal Processing, ICCSP 2020*, pp. 136–140, Jul. 2020, doi: 10.1109/ICCSP48568.2020.9182099.
- [15] S. Sen and A. Raghunathan, "Approximate computing for long short term memory (LSTM) neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2266–2276, 2018, doi: 10.1109/TCAD.2018.2858362.
- [16] Y. Yang, S. Tu, R. H. Ali, H. Alasmay, M. Waqas, and M. N. Amjad, "Intrusion detection based on bidirectional long short-term memory with attention mechanism," *Computers, Materials and Continua*, vol. 74, no. 1, pp. 801–815, 2023, doi: 10.32604/cmc.2023.031907.
- [17] 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, 2021, doi: 10.1155/2021/3806459.
- [18] S. I. Popoola, B. Adebisi, R. Ande, M. Hammoudeh, K. Anoh, and A. A. Atayero, "SMOTE-DRNN: a deep learning algorithm for botnet detection in the internet-of-things networks," *Sensors*, vol. 21, no. 9, 2021, doi: 10.3390/s21092985.
- [19] L. Zhou, C. Zhao, N. Liu, X. Yao, and Z. Cheng, "Improved LSTM-based deep learning model for COVID-19 prediction using optimized approach," *Engineering Applications of Artificial Intelligence*, vol. 122, 2023, doi: 10.1016/j.engappai.2023.106157.
- [20] Y. Baek and H. Y. Kim, "ModAugNet: a new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module," *Expert Systems with Applications*, vol. 113, pp. 457–480, 2018, doi: 10.1016/j.eswa.2018.07.019.
- [21] A. Nazir *et al.*, "A deep learning-based novel hybrid CNN-LSTM architecture for efficient detection of threats in the IoT




- ecosystem,” *Ain Shams Engineering Journal*, vol. 15, no. 7, 2024, doi: 10.1016/j.asej.2024.102777.
- [22] S. A. -Okyere, C. Shen, Y. Y. Ziggah, M. M. Rulegeya, and X. Zhu, “Principal component analysis (PCA) based hybrid models for the accurate estimation of reservoir water saturation,” *Computers and Geosciences*, vol. 145, 2020, doi: 10.1016/j.cageo.2020.104555.
- [23] B. S. Raghuvanshi and S. Shukla, “SMOTE based class-specific extreme learning machine for imbalanced learning,” *Knowledge-Based Systems*, vol. 187, 2020, doi: 10.1016/j.knsys.2019.06.022.
- [24] F. E. Laghrissi, S. Douzi, K. Douzi, and B. Hssina, “Intrusion detection systems using long short-term memory (LSTM),” *Journal of Big Data*, vol. 8, no. 1, 2021, doi: 10.1186/s40537-021-00448-4.
- [25] L. Liu, P. Wang, J. Lin, and L. Liu, “Intrusion detection of imbalanced network traffic based on machine learning and deep learning,” *IEEE Access*, vol. 9, pp. 7550–7563, 2021, doi: 10.1109/ACCESS.2020.3048198.
- [26] S. Jagadeesan and B. Amutha, “An efficient botnet detection with the enhanced support vector neural network,” *Measurement: Journal of the International Measurement Confederation*, vol. 176, 2021, doi: 10.1016/j.measurement.2021.109140.
- [27] A. Bijalwan, N. Chand, E. S. Pilli, and C. Rama Krishna, “Botnet analysis using ensemble classifier,” *Perspectives in Science*, vol. 8, pp. 502–504, 2016, doi: 10.1016/j.pisc.2016.05.008.
- [28] X. D. Hoang and Q. C. Nguyen, “Botnet detection based on machine learning techniques using DNS query data,” *Future Internet*, vol. 10, no. 5, 2018, doi: 10.3390/fi10050043.
- [29] S. Kudugunta and E. Ferrara, “Deep neural networks for bot detection,” *Information Sciences*, vol. 467, pp. 312–322, 2018, doi: 10.1016/j.ins.2018.08.019.
- [30] K. Saurabh *et al.*, “LBDMIDS: LSTM based deep learning model for intrusion detection systems for IoT networks,” in *2022 IEEE World AI IoT Congress (AIoT)*, 2022, pp. 753–759, doi: 10.1109/AIoT54504.2022.9817245.
- [31] R. Mannikar and F. Di Troia, “Enhancing botnet detection in network security using profile hidden Markov models,” *Applied Sciences*, vol. 14, no. 10, 2024, doi: 10.3390/app14104019.
- [32] H. El-Sofany, S. A. El-Seoud, O. H. Karam, and B. Bouallegue, “Using machine learning algorithms to enhance IoT system security,” *Scientific Reports*, vol. 14, no. 1, 2024, doi: 10.1038/s41598-024-62861-y.
- [33] U. H. Garba, A. N. Toosi, M. F. Pasha, and S. Khan, “SDN-based detection and mitigation of DDoS attacks on smart homes,” *Computer Communications*, vol. 221, pp. 29–41, 2024, doi: 10.1016/j.comcom.2024.04.001.
- [34] Y. Xiao, J. Liu, K. Ghaboosi, H. Deng, and J. Zhang, “Botnet: classification, attacks, detection, tracing, and preventive measures,” *Eurasip Journal on Wireless Communications and Networking*, vol. 2009, 2009, doi: 10.1155/2009/692654.
- [35] H. Liu and B. Lang, “Machine learning and deep learning methods for intrusion detection systems: A survey,” *Applied Sciences*, vol. 9, no. 20, 2019, doi: 10.3390/app9204396.
- [36] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.
- [37] S. Atef and A. B. Eltawil, “Assessment of stacked unidirectional and bidirectional long short-term memory networks for electricity load forecasting,” *Electric Power Systems Research*, vol. 187, 2020, doi: 10.1016/j.epr.2020.106489.
- [38] P. S. Muhuri, P. Chatterjee, X. Yuan, K. Roy, and A. Esterline, “Using a long short-term memory recurrent neural network (LSTM-RNN) to classify network attacks,” *Information*, vol. 11, no. 5, 2020, doi: 10.3390/INFO11050243.
- [39] T. H. Le, J. Kim, and H. Kim, “An effective intrusion detection classifier using long short-term memory with gradient descent optimization,” *2017 International Conference on Platform Technology and Service, PlatCon 2017*, doi: 10.1109/PlatCon.2017.7883684.
- [40] CSE-CIC-IDS2018, “CSE-CIC-IDS2018 on AWS: A collaborative project between the communications security establishment (CSE) & the Canadian Institute for Cybersecurity (CIC),” *Canadian Institute for Cybersecurity*, 2018. Accessed: May 11, 2024. [Online]. Available: <https://www.unb.ca/cic/datasets/ids-2018.html>
- [41] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” *ICISSP 2018 - Proceedings of the 4th International Conference on Information Systems Security and Privacy*, vol. 2018-Janua, pp. 108–116, 2018, doi: 10.5220/0006639801080116.

## BIOGRAPHIES OF AUTHORS






**Ahmad Heryanto**    received the M.Eng. degree in electrical engineering from the Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia, in 2014. He is currently pursuing the Ph.D. degree with Sriwijaya University, Indonesia. His research interests include parallel processing, distributed computing, software security, and network intrusion detection. He can be contacted at email: [hery@unsri.ac.id](mailto:hery@unsri.ac.id).






**Deris Stiawan**    received the Ph.D. degree in Computer Engineering from Universiti Teknologi Malaysia, Malaysia. He is currently a Professor at Department of Computer Engineering, Faculty of Computer Science, Universitas Sriwijaya. His research interests include computer network, intrusion detection/prevention system, and heterogeneous network. He can be contacted at email: [deris@unsri.ac.id](mailto:deris@unsri.ac.id).






**Adi Hermansyah**    received the M.Eng. degree in electrical engineering from the Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia, in 2019. He is currently a Lecturer with Faculty of Computer Science, Universitas Sriwijaya. His research interests include computer network, intrusion detection/prevention system, and heterogeneous network. He can be contacted at email: [adihermansyah@unsri.ac.id](mailto:adihermansyah@unsri.ac.id).






**Ricy Firnando**    received the M.Kom. degree in computer science from the Universitas Sriwijaya, Palembang, Indonesia. He is currently a Lecturer with Faculty of Computer Science, Universitas Sriwijaya. His research interests include information system, and informatic. He can be contacted at email: [ricyfirnando@unsri.ac.id](mailto:ricyfirnando@unsri.ac.id).






**Hanna Pertwi**    received her S.Kom. degree in computer science from Sriwijaya University, Palembang, Indonesia, in 2022. Her research interests include computer networks, intrusion detection/prevention systems, and heterogeneous networks. She can be contacted at email: [hannapertwi961@gmail.com](mailto:hannapertwi961@gmail.com).



**Mohd Yazid Bin Idris**    is an Associate Professor at School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia. He obtained his M.Sc. and Ph.D. in the area of Software Engineering, and Information Technology (IT) Security in 1998 and 2008 respectively. In software engineering, he focuses on the research of designing and development of mobile and telecommunication software. His main research activity in IT security is in the area of intrusion prevention and detection (IPD). He can be contacted at email: [yazid@utm.my](mailto:yazid@utm.my).



**Rahmat Budiarto**    received the B.Sc. degree from the Bandung Institute of Technology, in 1986, the M.Eng. and Dr.Eng. degrees in computer science from the Nagoya Institute of Technology, in 1995 and 1998, respectively. He is currently a Full Professor with the College of Computer Science and IT, Albaha University, Saudi Arabia. His research interests include intelligent systems, brain modeling, IPv6, network security, wireless sensor networks, and MANETs. He can be contacted at email: [rahmat@bu.edu.sa](mailto:rahmat@bu.edu.sa).