Journal of Applied Research and Technology

www.jart.icat.unam.mx

Journal of Applied Research and Technology 22 (2024) 846-862

Original

Intrusion detection system with an ensemble DAE and Bi-LSTM in the fog layer of IoT networks

G. F. Edakulathur[a]* • S. Sheeja[b] • A. John[c] • J. Joseph[d]

[a]*Department of Computer Science, Karpagam Academy of Higher Education, Coimbatore, India*
[b]*Department of Data Science, Sri Krishna Adithya College of Arts and Science, Coimbatore, India*
[c]*Department of Mathematics, St. Thomas College (autonomous), Thrissur, India*
[d]*Department of Mathematics, Carmel College (autonomous), Mala, India*

Received 03 08 2024; accepted 08 29 2024
Available 12 31 2024

Abstract: The world is rapidly arriving at the period of the IoT, which connects all types of technology to digital services and provides us with great ease. As the quantity of IoT-connected equipment increases rapidly, there may be a rise in network vulnerabilities, leading to an increase in network threats. Fog computing seems to be a distinctive paradigm that includes the cloud's network's edge, including practical computation and vital infrastructure. As a result of easy access to resources, the fog layer renders the system susceptible to several threats. Tackling these challenges entails detecting intrusions and tracing the route leading to the source of the threat. The objective of this study is to offer a security mechanism and demonstrate how an intrusion detection system can guarantee the integrity of *IoT* networks. Based on deep learning (*DL*) approaches, several promising intrusion detection systems (*IDSs*) have been presented, however, they need time-consuming parameter adjustment in various situations. To address this issue, this study suggests a hybrid Deep Auto Encoder (*DAE*) and *Bi-LSTM* for item installation in the fog due to the need to safeguard essential infrastructure against prompt and efficient identification of malicious threats. Further sparrow search optimization algorithm is proposed for parameter tuning. Utilizing IoT-based data, the effectiveness of the suggested model is assessed. The outcome of the experiment obtained by analyzing the suggested *IDS* utilizing CICIDS2017 and Bot-IoT datasets attested to their supremacy over modern systems that are currently available in terms of precision, accuracy, false alarm rate, and detection rate. To learn more about how well this model works, two additional metrics are added: Cohen's Kappa coefficients and Mathew correlation. The outcomes of our experiments and simulations showed that the suggested approach was stable and reliable across a variety of performance criteria and has achieved and accuracy of 98.7%. The experimental outcomes show that the proposed system can effectively describe normal activity inside fog nodes and identify various kinds of attacks such as Benign, Port Scan, DDoS, DoS GoldenEye, DoS Hulk and DoS Slowhttp.

Keywords: Intrusion detection, deep autoencoder, bidirectional LSTM, sparrow search algorithm, multiple attacks detection

*Corresponding author.
E-mail address: edakulathur@hotmail.com (G. F. Edakulathur).
Peer Review under the responsibility of Universidad Nacional Autónoma de México.

1. Introduction

Cyber-attacks are now becoming highly complex, making it more difficult to accurately identify intrusions. The reliability of security services, such as data protection, reliability, and accessibility, could suffer from an inability to halt intrusions. (Abeshu & Chilamkurti, 2018; Samy et al., 2020) To tackle hazards to computer security, various detection techniques have been developed. One of the security techniques that may be used to identify breaches at any layer of an *IoT* architecture and prevent security problems is intrusion detection. (Ferrández-Pastor et al., 2018; Mohammad et al., 2012; Vinayakumar et al., 2019). The majority of invasions are started by attackers or unauthorized users. An attacker may try to use the Internet to obtain remote access to a system or deactivate a service. Accurate intrusion detection required knowledge of how to effectively attack a system. Numerous techniques exist for detecting intrusions, such as methods that rely on statistical methods, cluster analysis, deep learning, or artificial neural networks. An *IDS* is a proactive intrusion detection technology used to quickly identify and categorize assaults, intrusions, security policies, and violations at the host and network levels of infrastructure. *IoT-IDS* will react in time to stop the assault when the attack happens. As a security measure for *IoT* networks, this technique can prevent attacks before they happen. (Kasongo & Yanxia, 2020; Suhaimi etal., 2019; Vinayakumar et al., 2019). The technology related to the *IoT* is referred to as the fourth industrial revolution. It is a system made up of interconnected software, computer devices, sensors, mechanical & digital machines, and some other techniques that allow connectivity and data sharing with various systems and devices across the Internet without requiring any interaction between people or computers. *IoT* devices are easily attacked by hackers because they have access to the Internet and lack adequate security safeguards.

Offering a productive structure to serve *IoT* and fog computing has evolved as a development of cloud computing. Fog serves as a facilitator by localized processing of the endpoint user's demands and reducing communication lag times between the final user and the cloud through the fog. Because of this, the receiving network activity on the fog node devices must be authentic. These systems are exposed to multiple intrusions. Fog Computing has evolved as a development of cloud computing by offering a productive infrastructure to serve *IoT*. Fog is regarded as an evolution of the cloud model from the network core to the network edge. It is a platform that offers a high degree of virtualization, in the words of Cisco, full cloud computing is often referred to as fogging or edge computing, which simplifies computer operations, and networking services between traditional cloud servers, and end devices. Devices storing data at cloud centres, end devices, and services connected to network fog devices make up fog computing data storage. (Hassen et al., 2020; Ijaz et al., 2021). Fog computing refers to a layer (fog) that sits close to the edge between the cloud and end users. It has appeared as a response to the issues with cloud computing's high latency and high energy usage. Fog devices are vulnerable to attack from malicious network entities because they have few resources, such as processors and memory. (Benrazek et al., 2020; Hussain & Beg, 2019; Liao et al., 2020). A hacker or intruder could sneak into the network and damage user information. An *IDS* is a strong tool for identifying intruders in a network. *IDS* is a practical method for enhancing fog computing privacy. In numerous sorts of research, an *IDS* based on fog computing is developed to identify and prevent external threats. *IDS*'s main purpose is to develop an equal response strategy according to the actions of the attacker. The performance of *DL*-based intrusion detection is superior to other methods among them. (Raza et al., 2013; V. Kumar et al., 2021).

In an existing model, a hybrid binary *kNN-DNN* classification algorithm is proposed. (Khan et a., 2021). The method is based on *DN* Network and the k Nearest Neighbour (*k-NN*) technique. It's perfect for putting together the initial phase of the two-stage detection technique used in the proposed design. (Roy et al., 2022). Another study suggests a unique intrusion detection model that is built on a classifier and a dimension reduction technique that may be utilized as an online machine learning approach. To scale down the dataset's dimensions from its many different attributes to a select few, the suggested model employs Principal Component Analysis. (Pajouh et al., 2016; Salo et al. 2019). An existing technique called Nave Bayes by anomaly Detect and Genetic Algorithm Based Wrapper feature selection Model in fog computing eliminates unnecessary qualities to cut down on processing time while also creating an improved model that can forecast outcomes utilizing the Knowledge Discovery Dataset for the Security Laboratory. (Onah et a., 2021). Figure 1 portrays the position of fog layer in IoT networks.

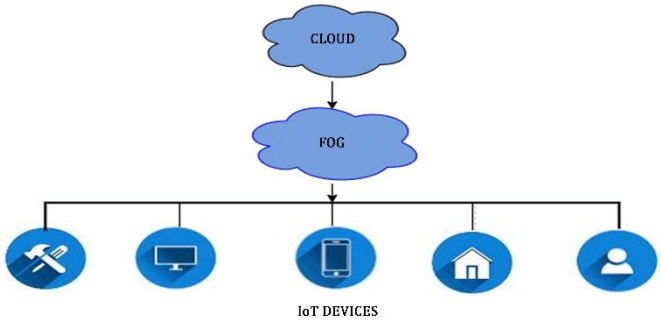

Figure 1. Fog computing.

There have been many deep learning-based intrusion detection systems suggested for the Internet of Things. It has merged hundreds of billions of items from various platforms with the internet. *IoT* networks have been the target of numerous hackers as a result of this convergence because it combines the digital and physical worlds. Combining deep learning with knowledge-based systems may help to solve the issue. Therefore, a hybrid model may be the next development for FOG-cloud *IDS*. Hence this paper aims to develop a method for detecting intrusion in the fog computing model using ensemble classification. In this paper, a hybrid deep autoencoder (*DAE*) and *Bi-LSTM* device architecture that uses the fog's advantages to deploy a timely and precise detection of harmful behaviors for the *IoT* network is suggested. The primary contributions of this study are as follows:

- A novel approach is suggested that combines the hybrid deep autoencoder (*DAE*) and *Bi-LSTM* for system implementation in the fog layer to protect essential infrastructure from accurate and timely detection of multiple attacks.
- This method is presented a sparrow search optimization algorithm (*SSOA*) for parameter tuning.
- In the proposed model the performance of our proposed methodology is assessed using IoT-based data.
- In comparison to current Models, our obtained results analyzing the suggested *IDS* using Bot-IoT and CICIDS2017 datasets prove their supremacy in terms of false alarm rate, accuracy, precision, false alarm rate, error rate, and detection rate.
- In order to better comprehend how this model performs, two additional metrics: Cohen's Kappa coefficients and Mathew correlation are considered. The results of our simulations and experiments demonstrated the stability and reliability of the suggested framework in terms of according to a range of performance metrics.
- Utilizing comparison and analysis of the suggested *IDS* with different solutions from the literature using the CICIDS2017 and Bot-IoT datasets.

The structure of the manuscript is as follows: A brief of the current *IDS* and related works utilizing various recurrent deep learning models is given in Section 2. Section 3 suggested the *IDS*'s specific processes. Section 4 outlines the system implementation, the experiments' justifications, and the processing of the data set. The results and analyses are further explained in Section 5 and the work is indicated and highlighted in Section 6.

2. Literature survey

A system that detects intrusions, which is essential for a system, acts as a clear defense line against cyber threats in the cyber security area. Intrusion detection systems (*IDS*) are a powerful method to find intruders in a network. Few research articles have been validated by the material, discussed in this study.

Sharma et al. (2023) innovative anomaly-based *IDS* for *IoT* networks make use of *DL* methods. In particular, they created a *DNN* model using filter-based FS that eliminates strongly correlated information. Additional parameters and hyperparameters were added to the model to fine-tune it. They employed the UNSW-NB15 database, which consists of four threat classes, for their study.

A feature extraction method called "k-means clustering," which comes from signal processing, was introduced by Shanker et al. (2023). It divides a collection of \(n\) observations into \(k\) clusters, each of which is centered around the observation with the closest mean. They used Python and the KDDcup99 dataset in their work to use the k-means algorithm and explore its possibilities. The efficacy of the outcomes proved how successful their strategy was in comparison to other publicly accessible options. They have created a web-based system that can detect network assaults by examining real network traffic packets.

Shandilya et al. (2023) presented a method for data collection that entails the use of Wireshark and tcpdump to record the network traffic of configured virtual computers. They built up a dataset with 10 machines trying to take advantage of one another while connected to Router1 on VLAN 1 in a Docker Bridge network to examine the effects of different cyberattack scenarios. The collection contains actions like online surfing and downloading foreign programs, some of which may be dangerous. Furthermore, a variety of attack techniques were used to compromise services including *FTP* and *SSH*.

Gudla et al. (2024) have presented a fog-based Internet of Things attacks prediction system using *DL* models. The best deep learning (*DL*) model with extreme precision is then predicted for placement at the fog layer after evaluation of many Deep Learning models, such as *CNN + LSTM, GRU, Bi-LSTM, LSTM, HEM,* and *DNMLP*. To discover flaws in *NW* security, this framework uses the *LSTM DL* model. According to this study, the *LSTMDL* techniques surpass *DNMLP* regarding properly predicting the attackers, although it requires more time to detect activity (*CBDT*) than other models. In order to evaluate the behavior of end *IoT* systems, the *LSTMDL* concept is configured in such a fog nodes computer module. *SDN* is taken into consideration while experimenting on the Anaconda platform.

Banaamah and Iftikhar (2022) have suggested DL-based detection techniques built on Gated Recurrent Units (*GRU*s), *CNN*, and *LSTM*. This study applied an established dataset *IoT* for intrusion detection called Bot-IoT. The raw dataset was processed as part of the pre-processing to ensure it was suitable for a DL approach. Standardization, normalization, and data cleaning are all part of this process. In the feature

selection process, four functionalities for Bot-IoT are selected. To categorize the attacks, they used three different kinds of neural networks. For the experiments, the datasets were split into training and testing datasets. This dataset was compiled using a realistic network configuration with both botnet and regular system traffic. They built our classifiers using the Kera library, which employed TensorFlow as its backend.

Ramkumar et al. (2022) have discussed using a fog computing platform to implement a hybrid ensemble classifier driven by optimization. The fog, cloud, and endpoint layers in fog computing are used as a trio to perform all of the processing. Three operations, including data transformation, feature selection, and classification, were executed in the cloud layer. By using log transformation, data is transformed. A feature is determined by utilizing Smirnov- Kolmogorov filter based on correlation. After that, data is classified using ensemble classifiers such as Shepard *CNN*, deep Neuro-Fuzzy Network (*DNFN*), and RideNN. The created Rider Sea Lion Optimized (*RSLO*) technique is used to tune the ensemble classifier. By combining the optimized Rider algorithm and Sea Lion Optimization, the RSLO algorithm has been developed.

According to Reddy et al. (2021) the device is exposed to multiple attacks due to the fog layer's quick access to resources. A unique breed of services that serve a broad range of *IoT* device applications is made possible by fog computing. Their research aims to provide a security mechanism and ensure that *IoT* networks are operating truthfully through the use of an intrusion detection framework. For device implementation in a fog based on the Exact Greedy Boosting ensemble technique, Reddy et al. demonstrated a system for detecting network intrusions. This suggested model investigates the monitoring of network traffic new intrusion Database 2020 data by determining and categorizing the kind of attacks depending on deviations from typical behavior. Utilizing a variety of machine learning and upgrading classifiers with XGBoost-based network *IDS* (*NIDS*) under fine-tuning of hyperparameters to detect anomalies. The fundamental machine learning technique is employed with several distributions sequentially to find the defective rule and rectify its existing one, resulting in the creation of a novel weak prediction. The suggested NIDS model is trained using the XGBoost approach and is superior to traditional ML algorithms.

P. Kumar et al. (2021) have presented *IDS* based on an ensemble distributed design that incorporates XG Boost, Gaussian naive Bayes, and k-nearest neighbors, as the first individual learners and also is based on fog computing. Random forest uses the first-level prediction findings to inform the final categorization at the second level. Pre-processing is done on the data from the training dataset during the first phase. It contains feature mapping, which turns categorical variables into numerical features, mutual information-based feature selection, imputation of missing values, and standard scalar feature normalization to create an optimum feature set. A mapping strategy is utilized that combines one hot label encoding (*OHLE*) and one hot encoding (*OHE*). This method uses a feature selection mechanism based on clear understanding. They are subjected to a test using UNSW-NB15, and the suggested method is verified by the use of an authentic DS2OS Iot-based dataset.

Yazan et al. (2022) have proposed DL-based *IDS* on deep learning to identify security risks in *IoT* environments. To obtain the best detection and recognition, our suggested module combines the Deep Stacked Polynomial Network (*DSPN*) with the spider monkey Algorithm (*SMA*) technique; in the data sets, *SMO* chooses the best characteristics, while *SDPN* categorizes the data as normal or anomalous. Pre-processing is done in the beginning to get rid of uncertainties in the normalized data set. Redundancy removal and missing value replacement are two efficient methods that are applied during pre-processing. The pre-processing step was first compared using similarity measures in the data of the data set which then calculated the distance between each pair of data using the Minkowski distance. Best features are chosen from the dataset at this stage after the pre-processing step, which removes any uncertainties from the dataset. The *SDPN* classifier's learning time is reduced mostly by best feature selection. The *SMO* model is modified: a fission-fusion social system-inspired optimization algorithm based on food foraging, to select the best features. The technique has been employed in many technical fields since it is simpler to use in complex optimization problems and requires fewer control parameters.

Zeeshan et al. (2021) have presented a Deep Intrusion Detection Based Protocol (*DIDBP*) method, comparing characteristics of the Bot-IoT and UNSWNB15 data sources based on *TCP* and flow to create a data set of packets of *IoT* traffic. Both sets of data's common properties for flow and *TCP* category are examined and integrated with *PB-DID*. The *LSTM*-based deep model with 26 features is created using the entire set of data from the Bot-IoT and UNSW-NB15 data sets via the suggested *DIDBP* architecture (by merging them to produce a single customized data set). The problem of data imbalance was also overcome during the integration of the data sets.

Intrusion detection is one of the primary security methods and aims to pinpoint attempted assaults findings from any of the research on intrusion detection in the Literature Survey are not satisfactory. Table 1 provide the accuracy obtained by the existing methods reviewed in this manuscript. Additionally, they have a high probability of false alarms and poor detection. In order to prove their effectiveness in real-time deployments, they must first be trained and assessed on realistic datasets. Due to the non-representative nature of the dataset used for training and evaluating the underlying models, several methods suggested in the literature have been observed to have low accuracy and be unsuccessful in

real applications. In order to stop multiple attacks in cloud networks, a distributed ensemble design-based *IDS* is suggested in this research employing fog computing.

Table 1. Datasets and accuracy of existing models.

Model	Dataset	Accuracy
(Sharma et al., 2023)	UNSW-NB15	91.00%
(Shanker et al., 2023)	KDDcup99	98.00%
(Shandilya et al., 2023)	Generated Dataset	NA
(Gudla et al., 2022)	DDoS-SDN	99.70%
(Gudla et al., 2022)	NSL-KDD	99.12%
(Gudla et al., 2022)	UNSW-NB15	94.11%
(Gudla et al., 2022)	IoTID20	99.88%
(Banaamah & Iftikhar, 2022)	Bot-IoT	99.60%
(Ramkumar et al., 2022)	UNSW-NB15	97.2%
(Reddy et al., 2021)	IoTID20	99.4%
(P. Kumar et al., 2021)	UNSW-NB15	93.21%
(P. Kumar et al., 2021)	DS2OS	99.17%
(Yazan et al., 2022)	NSL-KDD	99.02%
(Zeeshan et al., 2021)	UNSW-NB15	96.3%

3. Methodology

Fog computing is a methodology that expands the cloud to networks to address issues specific to clouds. Fog technology is a more recent type of computing with a variety of features that set it apart from cloud computing. As an outcome of insufficient resources, the fog nodes are vulnerable to cyber-attack. In order to improve service quality, *IDS* are a crucial part of the security of the fog network. *IDS* are an effective technique for locating network intruders. The conventional security of a network is difficult to prevent multi-source intrusion and cross-domain, much like physical security technologies. Since identifying malicious traffic is crucial for network security, so a system is recommended for intrusion detection should be used in fog circumstances in our research. In this instance, the fog nodes are in charge of gathering data from various network sensors and devices and conducting preliminary processing on it.

The steps that follow are included in the proposed method: Data collection, pre-processing, and feature extraction and categorization are the initial main steps. The network data are initially gathered from two benchmark datasets. The following phase then focuses exclusively on data pre-processing. A successful learning process depends on this step. Standardization, data cleansing, and normalization are the three stages of data pre-processing. Following pre-processing of the data, features are retrieved using Deep Autoencoder (*DAE*) and converted and classified into a format that can be utilized to instruct a deep learning technique.

The *Bi-LSTM* method is then trained using the features that were extracted for the multi-classification. The hybrid deep autoencoder (*DAE*) and *Bi-LSTM* architecture suggested is seen in Figure 2. The suggested hybrid model has decent accuracy. The steps that follow are included in the proposed method: Data collection, pre-processing, and feature extraction and categorization are the initial main steps. The network data are initially gathered from two benchmark datasets. The following phase then focuses exclusively on data pre-processing. A successful learning process depends on this step. Standardization, data cleansing, and normalization are the three stages of data pre-processing. Following pre-processing of the data, features are retrieved using Deep Autoencoder (*DAE*) and converted and classified into a format that can be utilized to instruct a deep learning technique. The *Bi-LSTM* method is then trained using the features that were extracted for the multi-classification. The hybrid deep autoencoder (*DAE*) and *Bi-LSTM* architecture suggested is seen in Figure 2. The suggested hybrid model has decent accuracy.

3.1. Data sets

For training and testing, two raw network packet databases namely CICIDS2017 and Bot-IoT are employed. Unlike other research using a portion of the dataset, entire data from the datasets are used for model training. A brief explanation of both the datasets is given below.

The CICIDS-2017 was launched by the Canadian Institution of Cyber force (*CIC*) in 2017. It includes updated real global attacks as well as the usual flows. It includes updated real-world attacks as well as common processes. The information premised on protocols, destination IP addresses, source timestamps, and attacks are used by CICFlowMeter to evaluate the network traffic. Additionally, this dataset includes the most current cyber-attacks, including Port Scan, Infiltration, DDoS, DoS, SQL injection, BoT, and Brute Force. 2,830,743 records from the CICIDS-2017 are contained in eight files with 78 unique features per record. It also includes research from a CICFlowMeter network traffic analysis study, which comprised flows labelled according to the date, destination ports, assaults, destination IP addresses, and source and protocols. Kurniabudi et al. (2020). Table 2 displays the CICIDS2017 distribution such as types of attacks in dataset, number of records on each attack and their percentage.

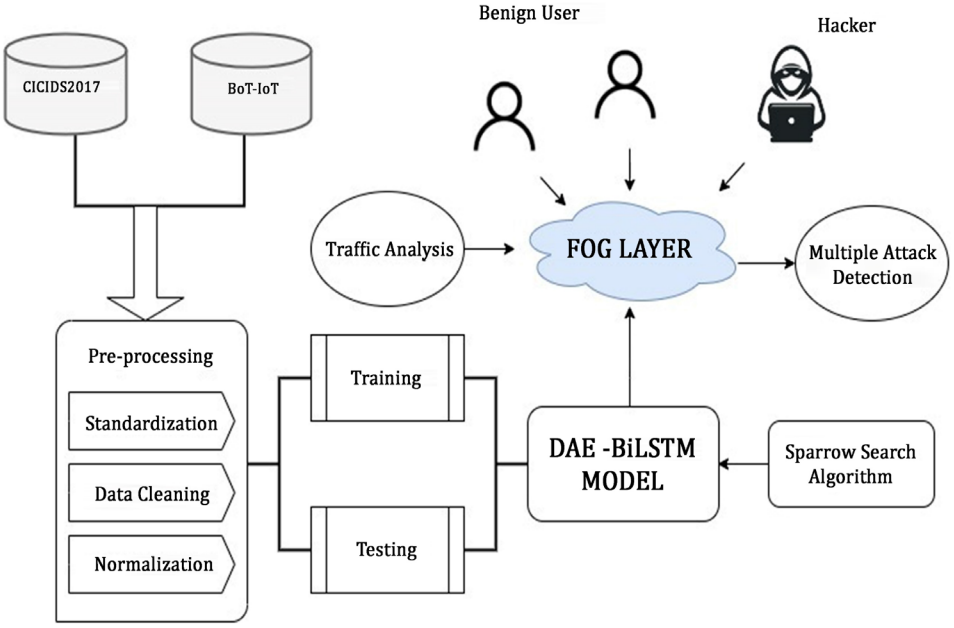

Figure 2. Architecture of the proposed model.

Table 2. CICIDS2017 dataset.

Flow Type		Number of Records	Percentage
Benign (Normal)		339621	61.32%
DoS GoldenEye		7458	1.35%
DoS Hulk		14108	2.55%
DoS Slowhttp		4216	0.76%
DoS Slowloris		3869	0.70%
SSH Patator		2511	0.45%
FTP Patator		3907	0.71%
Web Attacks	SQL Injection	12	0.00%
	XSS	631	0.11%
	Brute Force	1353	0.24%
BotNet		1441	0.26%
Port Scan		158673	28.65%
DDoS		16050	2.90%

The Bot-IoT data was created by the Cyber Range Lab of the Australian Centre for Cyber Defence. This data set is one of the newest datasets in the sector. Koroniotis et al. released the database in 2018 (Koroniotis et al., 2019). It has over 72 million recordings and combines simulated and real-time settings. The Bot-IoT dataset is the appropriate dataset for the tests since it includes *IoT*-generated network traffic as well as a wide variety of attacks, and regular updates, and can be used to distinguish points from new datasets. A recently developed dataset is used for Bot-IoT assault identification in an *IoT* network context. The four types of attack records are listed here: Reconnaissance, theft, *DoS*, and *DDoS*. The subcategories of *DoS* and *DDoS* include *UDP TCP* and *HTTP*. The dataset comprises traffic flows from botnet attacks as well as regular traffic on the Internet of Things and many more cyber-attacks. The realistic testbed is utilized to generate this dataset with efficient information features in order to track correct traffic. Table 3 displays the Bot-IoT Dataset distribution such as types of attacks in dataset, number of records on each attack and their percentage.

Table 3. Bot-IoT dataset.

Flow Type	Number of Records	Percentage
Benign (Normal)	9543	0.013%
Information Gathering	1821639	2.480%
DDoS	38532480	52.500%
DoS	33005194	45.000%
Information Theft	1587	0.002%

3.2. Data pre-processing

On the CICIDS-2017 and Bot-IoT datasets for training and testing, the pre-processing steps below were used. Pre-processing is a helpful technique that aims to clean it up data and remove noise. The raw dataset is processed in this step to make it acceptable for this algorithm. This is the method's initial stage. (Yin et al., 2023). Processing data aim to modify unstructured database into a simpler and more practical format for further processing stages. The step is divided into three sub-steps including standardization, data cleaning, and normalization. Dataset standardization is the initial stage. This process is crucial because it makes sure that the data are all scaled equally, and their normally distributed value lies between 0 and 1. The normalization of data is the second phase. The process of normalisation entails data transformation. Negative values, which neural networks rejected, must be avoided using this procedure. Each data element in the dataset was normalised between 0 and 1. In the third and final step, known as data cleaning, unnecessary data, such as NaN and null values, are eliminated. (Jairu & Mailewa, 2022) Following are the various data pre-processing steps.

Standardization is technique that implies that the datasets will be compressed to fall between zeros (0s) and ones (1s) using Batch Normalization, which tackles the issues of gradients that expand or collapse due to rapid learning rate in a conventional deep network. Normalizing network activations avoids small parameter changes from amplifying into larger and less variable gradient activation. i.e., it keeps the training from becoming trapped in nonlinearities' saturated regimes. The Min-Max Rescaling method can be used to achieve this. Every value in a column will be replaced by a new value using Eq. (1) below when using min-max scaling:

$$m = \frac{x - x_{min}}{x_{max} - x_{min}}$$
$$Where: m = new\ value,$$
$$x = original \text{ cell value},$$
$$x_{min} = \text{minimim value of the column},$$
$$x_{max} = \text{maximum value of the column}. \quad (1)$$

The database was examined by the data cleaning method to see if any columns or rows contained blank or unutilized values. Then, the means of the values that were empty or unused before and after the sample are estimated. Finally, the null values are replaced with the mean. As a result, outliers were removed, and more consistent data were generated.

In order to scale the data into a particular range [0, 1], normalization was used without altering the normalcy of the data behaviours. By tackling local optimal issues, this stage facilitates the convergence of the statistical method and deep learning techniques and helps them fulfil their goals. To enhance data quality, this method focuses on pre-processing and normalization. The normalization technique map characteristics between 0 and 1 using the min-max normalization in Eq. (2). The minimum value for each feature is set to 0, the max value to 1, and any other value is transformed into a decimal number between 0 and 1.

$$X[i] = (X[i] - X_{min}) / (X_{\max} - X_{min}) \quad (2)$$

3.3. Features extraction

An intrusion prevention method for the Fog computer system is suggested in this work. It is essential for protection to recognize harmful traffic networks. This article describes our Deep Auto Encoder (*DAE*) and Bidirectional *LSTM* (*BiLSTM*)-based proposed technique for intrusion detection. Although *DAE* achieves decent accuracy on its own, *Bi-LSTM* is used to improve the accuracy further. This section contains the method of feature extraction utilizing a Deep Autoencoder (*DAE*).

3.3.1. Deep autoencoder

An autoencoder is a traditional unsupervised neural network that uses repeated back propagation to attempt to set its target values to equal its inputs. An autoencoder is a type of neuronal structure that encodes input base data for output data reconstruction. The autoencoder must first learn to recognize the key input features in order to start this procedure. The deep autoencoder is a powerful unsupervised feature representation method with several buried layers. The neural idea of learning data is motivated by the fact that the characteristics of hidden nodes are automatically learned from the input data and are not manually generated. During transformation, the deep features in high dimensions are compressed to low dimensions with very small distortion. The deep characteristics of the series of frames are learned and retrieved, together with its underlying patterns and shape changes utilizing a reliable four-layer stacked autoencoder design, as shown in Figure 2. The first layer conveys 8000 neurons to a 15000-dimensional feature map, followed by reductions in dimensions of 4000, 2000, and 1000, respectively. To make the autoencoder's temporal complexity less complicated, high-dimensional data is reduced by a half factor. High computational complexity is produced by compressing high dimensional data with few deep layers and simple techniques. In the input data, the DAE learns "part-whole decomposition" or "hierarchical grouping" (Ullah et al.,

2019; Badr & Samma, 2022; Tong et al., 2020). The early stages of the Deep stacked autoencoder collect the changes and initial order characteristics in the raw original input data. The second-order features that correspond to the patterns observed in the first-order features are taught to the intermediate layers on the other side. (Zhou et al., 2020). The AE comprises two phases: encoding, in which weights and biases are added to the data, and some non-linearity function, such as the sigmoid and *relu* has given in Eq. (3) Following that, the data is decoded to the same number of inputs as in Eq. (4). To bring the mean squared error close to zero; the weights are modified using a backpropagation algorithm.

$$h(x) = sigm(Wx + b) \tag{3}$$

$$\hat{x} = sigm(W(h(x)) + b) \tag{4}$$

In the deep stacked autoencoder, the initial hidden layer receives input x, meanwhile, the other receives information from the prior hidden layer, as shown in Eq. (5) and Eq. (6). Here $x^l, W^l and\ b^l$ are the data, ‘‘n’’ represents the number of encoding layers, the biases of the relevant layer, and the weights, respectively.

$$h(x)^{(l+1)} = sigm(W^l x^l + b^l) \tag{5}$$

$$\hat{X}^{(n+l+1)} = sigm(W^{(n-l)} h(x)^{(n+l)} + b^{(n-l)}) \tag{6}$$

The suggested *DAE* can be developed for 400 epochs. The L2 weights regularisation is used to reduce the "falling into local minima" and over-fitting problems. Additionally, a sigma value 0.05 is used for the sparsity regularisation, which results in an average output of 0.5 for each hidden layer neuron over the training data.

In order to fine-tune the *DAE* weights, mean squared fault (*MSE*) with L2 regularisation and sparsity adjustment is utilized as a cost. The mistake is reduced up to $\mathbf{10^{-2}}$in 300 epochs, and during the final epoch of the training phase, it was 0.0077. Testing and training are the two stages of the *DAE-IDS*. In the training phase, the system builds a model based on the suggested *DAE* model using a training dataset. In the testing phase, the system uses the model for detecting the label of unseen data (test dataset) to estimate how well it will function if deployed online. Input, hidden, and output layers are the three types of layers that make up *DAE*. The training dataset serves as the input layer. The 117 features from the CICIDS2017 and Bot-IoT data base are all represented in the input layer of our DAE model. (Huang et al., 2015).

3.3.2. Bidirectional LSTM (BiLSTM)

The proposed DAE-based strategy for extracting the feature is described here. Furthermore, *Bi-LSTM* is used to classify the data. Data are grouped by classification based on a label or target class. The techniques for solving classification issues belong to the category of supervised learning. In this model, the *Bi-LSTM* classification algorithms are employed to assess how the suggested framework improves classification performance. A new recurrent neural network learning design is suggested to address this demand, which can improve the structure's temporal organization. (Zhu & Nasser, 2021). At the next time stamp, the output can be instantaneously fed back into itself. *RNN* is a model that is frequently used in deep learning. Deep neural networks have successfully learned the hierarchical aspects of natural language in recent sentiment analysis experiments. However, *RNN* suffers from a slant disappearing gradient exploding problem, whereas the A memory module in the *LSTM* has the ability to select which data should be stored in memory and when it should be deleted. As a result, *LSTM* can successfully address the issues of gradient disappearance and training challenges by mining time series with delays in the time series and comparatively large intervals. There are three layers in a standard *LSTM* network framework: input, hidden loop, and output. The cyclic hidden layer, in contrast to the conventional recurrent neural network, mostly consists of neuron nodes. Memory modules serve as the foundational building block of *LSTM* cyclic hidden layers. Output gate, forget gate, and Input gate are the three adaptive multiplication gating units contained in this memory module. The *LSTM*'s each neuron nodes carry out the following calculation: The input gate is set at time *t* in accordance with the output outcome $h_{t-1}$of the unit at the time in question and is given in Eq. (7). The input x_t at that precise instant depends on whether to do a calculation to update the current data in the cell.

$$i_t = sigmoid\ (W_t \cdot [h_{t-1}, x_t] + b_t) \tag{7}$$

A forget gate is used to determine whether to keep or put away the information depending on the most recent hidden layer output and the current time input and is given in Eq. (8).

$$f_t = sigmoid\ (W_f \cdot [h_{t-1}, x_t] + b_f) \tag{8}$$

The previous output result h_{t-1} of the hidden layer *LSTM* cell determines the value of the current candidate memory cell and the current input data x_t. At this instant, Character $*$ is the element-wise matrix multiplication, the memory cell state value C_t modifies the current candidate cell C_t and its own state C_{t-1}input gate and forget gate. These memory cell state values are given in Eq. (9) and Eq. (10).

$$\bar{C}_t = tanh\ \ (W_C \cdot [h_{t-1}, x_t] + b_C) \tag{9}$$

$$C_t = f_t * C_{t-1} + i_t * \bar{C} \tag{10}$$

Output gate o_t is determined as shown in Eq. (11) and it is utilised to regulate the value of the cell status. The result of the final cell is h_t, which can be written as Eq. (12).

$$o_t = sigmoid\,(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{11}$$

$$h_t = o_t * tanh(\,C_t) \tag{12}$$

The forward *LSTM* network and backward *LSTM* network make up the *Bi-LSTM*. Both the forward and the backward *LSTM* hidden layers are in charge of extracting features; the forward layer extracts features in the forward direction. The *Bi-LSTM* model can be used to take into account the effects of each characteristic both before and after the sequence data. As a result, more detailed feature information is acquired. *Bi-LSTM*'s current state contains both forward and backward output and they are given in Eq. (13), Eq. (14) and Eq. (15)

$$h_t^{forward} = LSTM^{forward}(h_{t-1}, x_t, C_{t-1}) \tag{13}$$

$$h_t^{backward} = LSTM^{backward}(h_{t-1}, x_t, C_{t-1}) \tag{14}$$

$$H_T = h_t^{forward}, h_t^{backward} \tag{15}$$

Therefore, the proposed model uses a Deep Autoencoder to extract the feature and a Bidirectional *LSTM* (*Bi-LSTM*) to classify the data. The Sparrow Search Optimization Algorithm (*SSA*) will then be used to implement the tuning parameter. By selecting a range of values and doing the experiments, the parameters are tuned. The ideal values are selected, resulting in the highest level of precision (Huang et al., 2015).

3.4. Sparrow search optimization algorithm

A novel version of the swarm intelligence algorithm called the Sparrow Search optimization Algorithm (*SSOA*), has been set to use in a variety of applications. Due to its distinctive qualities, including its worldwide search capabilities, a small set of tuning parameters, and clear structure. Around the world, farms and forests are home to a variety of little birds known as sparrows.

In the past, sparrows could be found all over Europe, as well as in some regions of North Africa and Asia. However, sparrows were brought to these regions by migrants from other continents, such as Australia and the United States, and now they are part of the local ecosystem. Among the omnivorous birds, sparrows mostly take seeds but can also eat small insects, berries, and fruits. Some sparrow species, including pigeons and house sparrows grown in captivity, are used to residing in urban areas. This little bird will eat absolutely everything. Currently, the house sparrow is the wild bird species with the greatest global distribution. Although this particular species of the sparrow is intimately tied to human surroundings, other sparrow species also frequented residential areas. Sparrows can be found in a wide range of climates and environments; however, they normally stay closer to inhabited cities and stay far away from meadows, deserts, and deep forests. This species has two different types of individuals: the scrounger and the producer, and while the producers search for sources of food, the scroungers get their food by nagging the producers. The birds frequently alternate scrounging and creating, as well as flexible interactive plans. It can also be said that sparrows typically employ both producer and scavenger strategies to find food. The studies suggested that each person in the group keeps an eye on one other's behaviour. In the meantime, the flock's predatory birds battle for the food sources of their friends who consume more than they do. Additionally, sparrows' energy stores play a key role in their decision-making process when it comes to hunting tactics; sparrows with insufficient energy collect more (Zhu & Nasser, 2021).

The sparrow search optimization method makes use of sparrows' predatory and anti-predatory behaviour as a novel method for swarm intelligence optimization. The SSA process is divided into the following steps, and Figure 3 depicts a flow diagram for its algorithm.

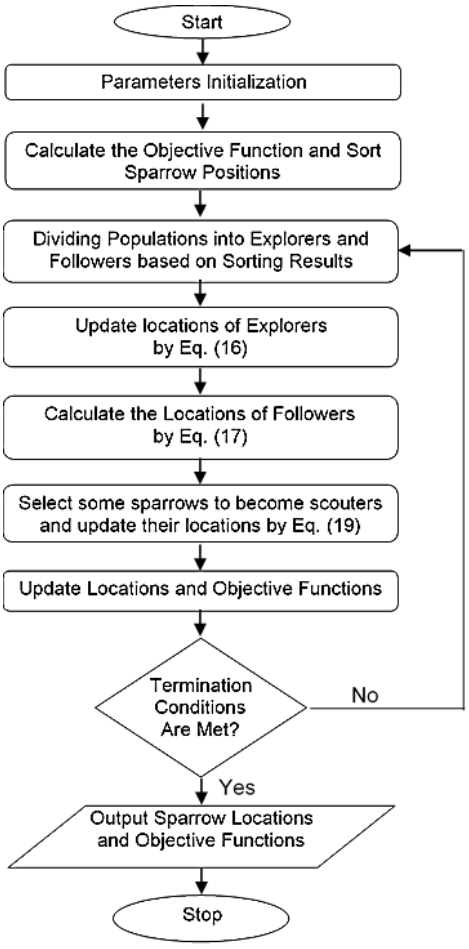

Figure 3. Sparrow search algorithm.

Step 1: number of explorers, the location of the sparrow, and the primary components of parameter initialization are setting the number of sparrows and the number of iterations. The quantity of n sparrows can be expressed as:

$$X = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^d \\ x_2^1 & x_2^2 & \cdots & x_2^d \\ \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^2 & \cdots & x_n^d \end{bmatrix} \quad (16)$$

where X_d^n specifies the location of the n^{th} sparrow in dimension d; d is the dimension of the variable to be optimized, and n is the population size.

Step 2: Establish the objective method and arrange the sparrow places. The following list identifies the i^{th} sparrow's objective function:

$$F_i = f\left(\begin{bmatrix} x_i^1 & x_i^2 & \cdots & x_i^d \end{bmatrix} \right) \quad (17)$$

where the objective function is denoted by *F*.

Step 3: Determine if the population is safe where it is at the present and alter the explorer's location.

$$X_{ij}^{t+1} = \begin{cases} X_{ij}^t \cdot exp\left(\frac{-i}{\alpha \cdot b}\right) & , R_2 < ST \\ X_{ij}^t + QL & , R_2 \geq ST \end{cases} \quad (18)$$

where X_{ij}^t denotes the value of the j^{th} dimension in the i^{th} sparrow in t^{th} iteration; max amount of iteration is represented by the constant b;A randomly distributed number, R_2 represents the warning factor in the range of [0, 1].; The safety threshold, denoted by the symbol ST, has values between [0.5, 1.0]; Q is an odd amount subjecting to standard distribution, L is $1 \times d$ dimensional matrix; When $R_2 > ST$ means the nearby area is considered secure, and the sparrow swarm departs in search of food. On the other hand, the current area is in danger, thus the explorer must lead the sparrows swarm together in quest of a fresh food supply.

Step 4: Obtain a position update and ascertain the follower's status. The following changes have been made to the area:

$$X_{ij}^{t+1} = \begin{cases} Q \cdot exp\left(\frac{X_w^t - X_{ij}^t}{i^2}\right) & , i > n/2 \\ X_p^{t+1} + \left| X_{ij}^t - X_p^{t+1} \right| \cdot A^+ \cdot L & , i \leq n/2 \end{cases} \quad (19)$$

where $A^+ = A^T(AA^T)^{-1}$, X_w is the worst position in the population of sparrows represents the worse position; X_p is in place of the ideal explorer; When $i > n/2$, it means that the follower its poor positioning and lack of access to food; A is a $1 \times d$ dimensional matrix with random generation of each dimensional value $[-1,1]$. Follower needs to go to different locations where it can receive more food. On the other hand, it keeps looking for food close to the explorer.

Step 5: When few of the sparrows notice a danger, they become scouters and update the locations as follows:

$$X_{ij}^{t+1} = \begin{cases} X_b^t + \beta \cdot \left| X_{ij}^t - X_b^t \right| & , f_i > f_g \\ X_{ij}^t + K \cdot \left(\frac{\left| X_{ij}^t - X_b^t \right|}{(f_i - f_w) + \varepsilon} \right) & , f_i = f_g \end{cases} \quad (20)$$

Wherein the population of sparrow X_b indicates the population's ideal position; f_g is the objective functions of the best value;$f_i$is the i^{th} sparrow objective function, and $f_w$is represents the worst range of the target value; K is a standardised odd amount with range of $[-1,1]$; β is a typical properly a widely spaced odd number. In order to keep the denominator from reaching zero, ε is a smaller value.

Step 6: Develop a distinct essential performance.

Step 7: Ascertain whether the iteration stop condition is attained; if it is not, repeat steps 3 through 6.

4. Result and analysis

A detailed discussion of the suggested method's evaluation criteria and specified dataset is provided in this part. To detect threats in the *IoT* network environment, first, the dataset and then the evaluation criteria are addressed. The Sparrow Search Optimization Algorithm is used to perform parameter adjustment after the data was classified by *BiLSTM*.

4.1. Dataset description

Two datasets, Bot-IoT and CICIDS-2017, were used to test the DAE-BiLSTM model and compare it to various techniques. The majority of researchers examine the efficacy of suggested systems using these datasets.

The Canadian Institution for Cyber-security provided the CICIDS-2017 database as an open-source intrusion detection dataset in 2017. The dataset includes safe and recent major strikes like DoS, Brute force, Web-based, infiltration, heart-bleed, BOT, and DDoS. The PCAP traffic data are analysed for network traffic using *CIC* Flow Meter to produce *CSV* files. The mot recent common attacks are listed in the program known as the *CIC* Flow Meter, which is available to the public on the *CIC* website (Canadian Institute for Cybersecurity, 2017). One of this dataset's features is its ability to produce realistic background traffic. The B-Profile creates naturalistic benign background traffic and is in charge of profiling the abstract behaviour of human interconnections. The B-Profile for CICIDS2017 uses *HTTPS*, *HTTP*, email protocol, *SSH*, and *FTP* to

extract the abstract behaviour of 25 users. In this work, a total of 2830743 instances and 80 features are employed, where there are 2273097 benign and 557646 malicious attacks, respectively. A ratio of 60% to 40% was used to divide it into training and test datasets.

Bot-IoT data made accessible by UNSW Cyber Range Lab in Canberra. Given that it was developed in an environment specifically designed for the Internet of Things and contains a sufficient number of records with diverse network profiles. This dataset provides a true representation of an *IoT* network. Over 72 million recordings of network activity in a simulated Internet of Things environment make up the Bot-IoT dataset. Additionally, a dataset contained 3.6 million records is used, for this research. The original dataset contains lists of the top 10 features, which were also utilized in this study. Each of the training and test datasets has five output classes that represent both the regular traffic and the four different kinds of attacks that were made against the *IoT*.

4.2. Data pre-processing

Data processing is necessary for all information retrieval processes, particularly network-based intrusion detection attempts to distinguish between regular and abnormal network traffic. Kurgan and Musilek evaluated the many formal process models that have been suggested for knowledge discovery and data mining (*KDDM*). These models predict that the data pre-processing phase requires 50% of the total process work, whereas the data mining activity only requires 10% to 20% of the total process effort. So, the data pre-processing stage is the main emphasis of this work. Standard pre-processing procedures include standardization, normalization, and data cleaning. Graphical representation of distribution of records of each type of attacks before and after data pre-processing is given in Figure 4 and Figure 5 for CICIDS2017 dataset and in Figure 6 and Figure 7 for Bot-IoT dataset.

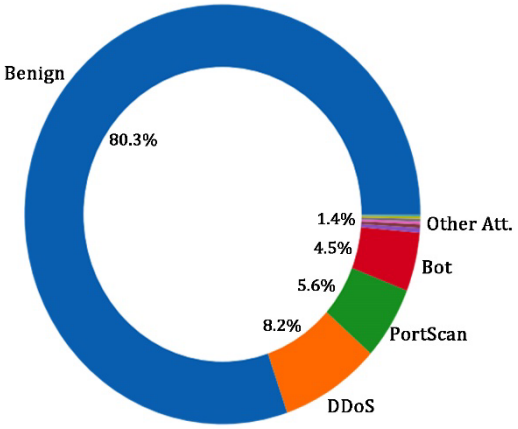

Figure 4. Before pre-processing data (CICIDS2017).

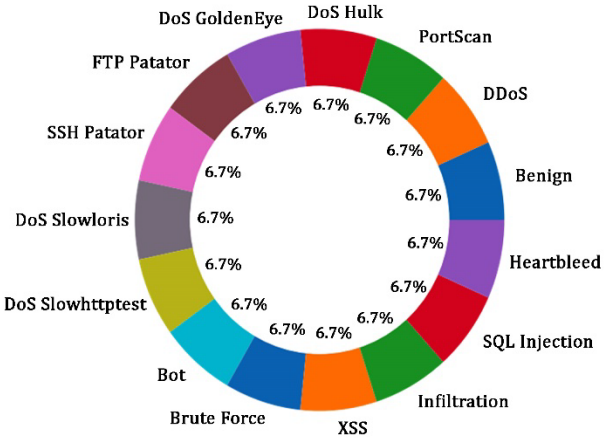

Figure 5. After pre-processing data (CICIDS2017).

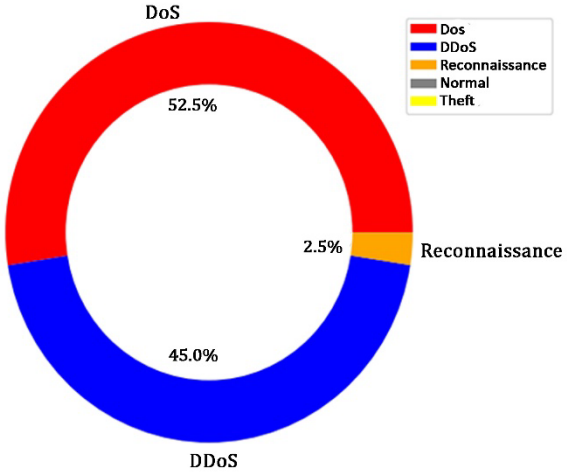

Figure 6. Before pre-processing data (Bot-IoT).

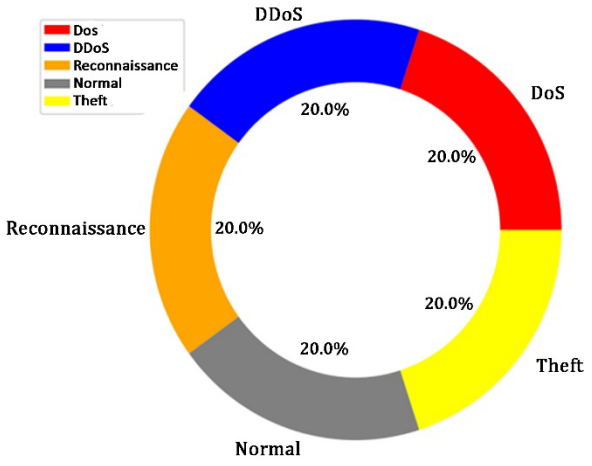

Figure 7. After pre-processing data (Bot-IoT).

4.3. Training and testing

To measure the model's functionality, the dataset was split into train and test sets. Data were used for training in the proportion of 80% and testing at 20%. Table 4 provide a brief description of splitting of dataset records for training and testing the proposed model.

Table 4. Summary of datasets.

Dataset	Attacks	Training Data	Testing Data
CICIDS2017	Normal	3,18,087	1,36,219
	Infiltration	5	1
	Web Attack	292	134
	Port Scan	22,324	9,558
	BoT	265	102
	DoS/ DDoS	53,427	23,018
	Brute Force	1,904	813
	Total	**3,96,304**	**1,69,845**
Bot-IoT	Normal	286	191
	DoS	1,46,293	97,529
	DDoS	1,63,287	1,08,858
	Reconnaissance	54,649	36,433
	Theft	47	32
	Total	**3,64,562**	**2,43,043**

Figure 8 and Figure 9 displays how DAE-BiLSTM performed for the CICIDS2017 dataset in terms of training and analysis accuracy. With a size of 10 epochs or greater batch, there is an improvement in the training and testing accuracy for multi-classification.

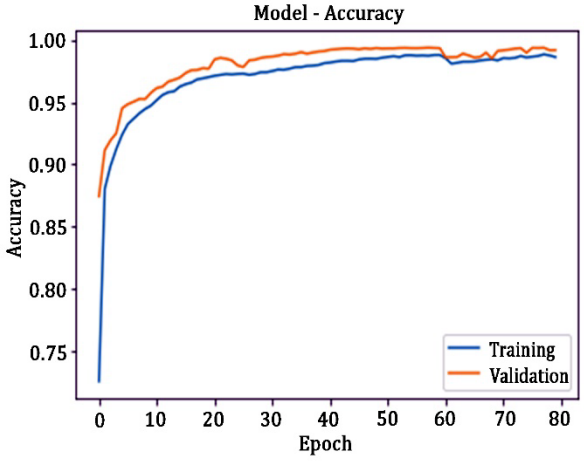

Figure 8. Model accuracy using CICIDS2017.

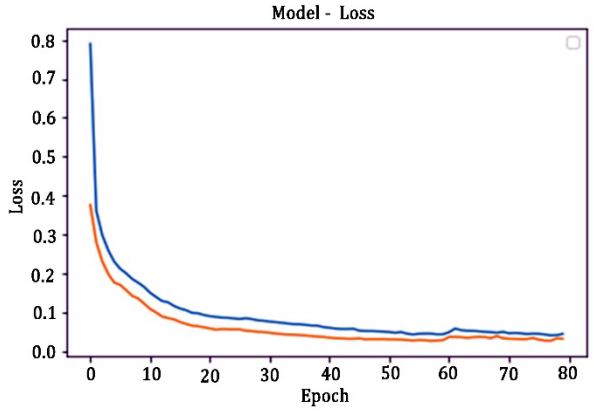

Figure 9. Model loss using CICIDS2017.

Figure 10 and Figure 11 displays the DAE-accuracy BiLSTMs in training and validation for the Bot-IoT dataset. An increase in epochs and a batch size of 10 for multi-classification indicates an improvement in training and testing accuracy.

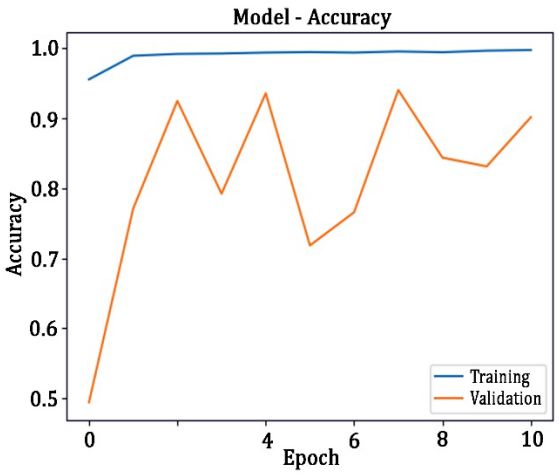

Figure 10. Model accuracy using Bot-IoT.

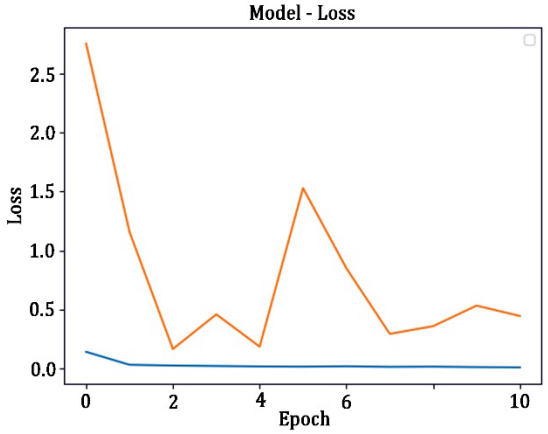

Figure 11. Model loss using Bot-IoT.

4.4. Performance evaluation

Along with various performance evaluation measures such as accuracy, Error rate, false alarm rate, Detection Latency, detection rate, and precision, two more performance measures are added: Matthew Correlation and Cohen's Kappa Coefficient. The primary motivation for this new adaption is monitoring the performance stability of recursive networks. Our proposed DAE-BiLSTM model's effectiveness is determined. These metrics are provided in the equations below:

The accuracy of a model's predictions is measured as a percentage of correct predictions. Based on its confusion matrix, a classification model's accuracy is measured. A balanced dataset is used to give a comprehensive evaluation of the model. It is characterized as the ratio of accurate predictions to all other predictions, and this can be calculated utilizing the Eq. (21).

$$Accuracy = \frac{TP+TN}{FP+FN+TN+TP} \tag{21}$$

where FN=false negatives, FP=false positives, TN=true negatives and TP=true positives

The detection rate is expressed as the discrepancy between the actual and anticipated numbers of anomalous samples. The *DR* represents the method's capacity to evaluate attacks, a crucial indicator in IDSs. The specific computation is stated as follows:

$$Detection\ rate(DR) = \frac{TP}{TP+FN} \tag{22}$$

The false alarm rate also referred to as the false positive rate, calculates the percentage of regular network traffic flows that are misclassified. The computation appears as this:

$$False\ Alarm = \frac{FP+FN}{TP+TN+FP+FN} \tag{23}$$

Another important parameter for assessing machine learning techniques is precision. The equation shows that this rate is the proportion of accurately predicted malware samples.

$$Pr\,e\,cision\ (PRC) = \frac{TP}{TP+FP} \times 100 \tag{24}$$

$$Error\ Rate = 100 - Accuracy \tag{25}$$

In machine learning, the *MCC* is used to assess the efficacy of binary (2-class) classification, which is typically utilized in binary classification. *MCC* measures the degree of agreement between the precise and anticipated binary classifications, which typically returns a value of 0 or 1. The *MCC* value thus provides a more accurate indication of the classification model. Meanwhile, this does not negate other performance criteria. Equation 26 is utilized to determine the *MCC* metric.

$$MCC = \frac{TP \times TN - FN \times FP}{\sqrt{(TN+FP)(TP+FP)(TN+FN)(TP+FN)}} \tag{26}$$

J.A. Cohen first established Cohen's Kappa statistic, or just Kappa (henceforth, also indicated by K), in the field of psychology as a measure of agreement between two judges. Later, it was used as a classification performance statistic in the literature. Kappa is a ratio of agreement between the observed and predicted or derived groups for cases in a testing dataset to put it more accurately. It is defined as:

$$K = \frac{Acc - P_e}{1 - P_e} \tag{27}$$

Performance measures in prediction modelling do not give a clear view of our categorization, especially the extremely balanced dataset used. It can effectively manage classes with imbalances. The mathematical representation of Cohen's Kappa (K) coefficient is as in:

$$K = \frac{Obs^{agree} - Expect^{agree}}{1 - Expect^{agree}} \tag{28}$$

$$where,$$

$$Obs^{agree} = accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)}$$

$$Expect^{agree} = \frac{A+B}{(TP+TN+FP+FN)}$$

$$Where,$$

$$A = \frac{(TP+FP)(TP+FN)}{(TP+TN+FP+FN)}$$

$$B = \frac{(TN+FP)(TN+FN)}{(TP+TN+FP+FN)}$$

Fog nodes located near network edges gathered information from *IoT* devices (detectors and sensors), analysed it, and recorded it utilizing network edge equipment in distant network areas. As a result, data flow across the network was dramatically decreased with high-quality, long-term, high-speed, high-quality, and localized endpoint services for real-time connection and low latency, particularly with time-sensitive or latency applications. Because the identification of intrusions is a latency-restricted application, the anomaly detection score must be calculated promptly. The time needed to compute the discrimination and reconstruction losses largely determines how long this period will last. To evaluate the detection latency of our proposed IDS, the discriminatory loss, total reconstructive loss, and a composite of both losses are taken into account. But when the reconstruction loss was also taken into account, the latent rose. This is due to the fact that calculating the reconstruction loss and identifying the latency demonstration of a sample both take time. The encoder within our architecture offers a significant reducing the time needed to identify intrusions

because it immediately maps patterns to their latent representation, whereas earlier research addressed optimization difficulties during intrusion detection. The detection of intruders in CPSs, for example, is a good example of an application where it is much more suitable. Since *ALAD* (Adversarially Learned Anomaly Detection) only uses fully connected and convolutional layers in neural networks, it does not actually experience the restricted parallelization that RNN-LSTM networks allow. Being less computationally intensive, it has a faster detection delay than our technique. Our *IDS* does certainly perform better than IDS-based *ALADs* at discovering breaches in cyber-physical systems. Figure 12 graphically represent the details of detection latency with Cumulative Distribution Function.

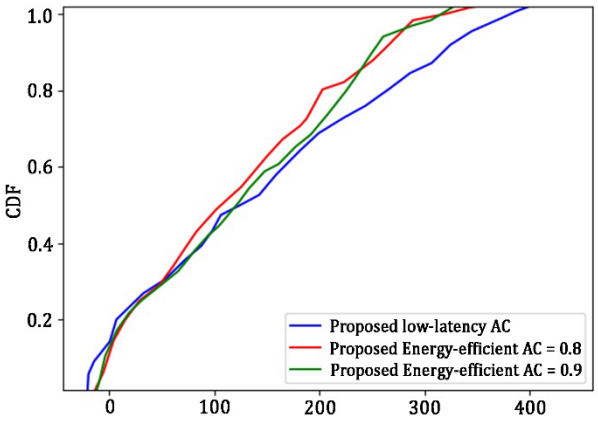

Figure 12. Detection latency.

Compared to alternative approaches like the Gaussian naive Bayes, K-NN, and Bayesian classifier, our proposed method has a lower error rate, and it is displayed in Figure 13.

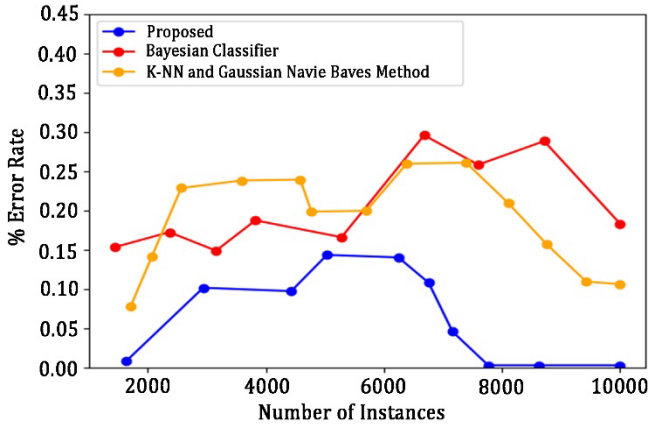

Figure 13. Error rate.

The matrix's diagonal shows the real detection numbers, while the other rows and columns display the inaccurate detection values. The scores the model produced and the different kinds of error. The weighted average accuracy of the *DAE - Bi-LSTM* model achieved the best result. Figure 14 and Figure 15 exhibits the confusion matrix of the results.

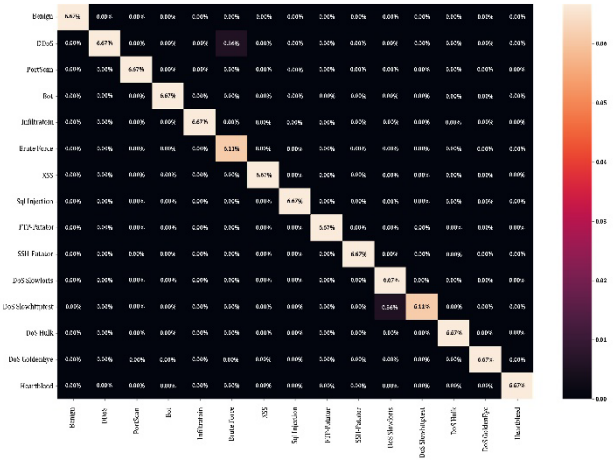

Figure 14. Confusion Matrix.

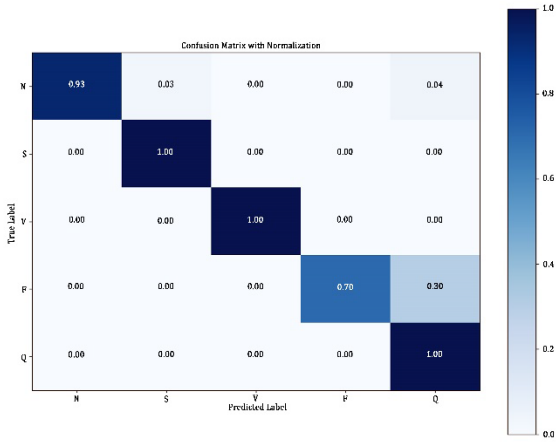

Figure 15. Confusion matrix with normalization.

4.5. Comparison with other state of the art methods

The necessity to detect intrusions in the contemporary cyber world has led to extensive research on the subject. For intrusion detection, researchers have used a wide range of powerful and sophisticated machine-learning techniques. In this part, the proposed method's accuracy over the CICIDS2017-BoTIoT datasets to various methodologies are evaluated for tool-based intrusion detection using DL and classic ML techniques. The model expresses excellent performance across all metrics in precision (0.98816), accuracy (0.987256), error rate (0.012744), *DR* - detection rate

(0.914526), *FAR* - false alarm rate (0.65536), MCC (0.941027), and Cohen's Kappa (0.892167). According to Table 5, approaches based on DAE-BiLSTM have shown better results than other methods. The techniques listed in Table 4 have been assessed utilizing CICIDS2017-BoTIoTTrain+ and CICIDS2017-BoTIoT Test+ datasets.

Table 5. Compare with other state-of-the-art methods.

Methods	Accuracy	Precision	DR	FAR
HDT	83.1485	97.2193	72.4694	73.0394
DT	80.9084	96.7753	68.7524	80.3918
KNN	79.1209	70.7361	89.5455	79.0371
SVM	78.5215	71.4286	85.2273	77.7202
Proposed	98.7	98.81	91.45	65.5

5. Conclusion

Intrusion detection is a vital security technology that guards computer systems and networks against unauthorized access and attacks. In this research, a viable intrusion detection method for the fog node to identify threats is constructed. In order to create an effective attack detection method, this research used a hybrid deep autoencoder (*DAE*) and *Bi-LSTM* model. After pre-processing of the data, features are retrieved using Deep Autoencoder (*DAE*) and converted and classified into a format that can be utilized to instruct a deep learning technique. The *Bi-LSTM* method is used to train the classification model using the features that were extracted. Finally, for optimization, to adjust model parameters, Sparrow Search Optimization Algorithm (*SSOA*) is used. The *DAE-BiLSTM* model was tested and compared with several other methods using two datasets, Bot-IoT and CICIDS-2017. The suggested model was found to be stable and robust based on the outcomes of the simulations and experiments, and for a more comprehensive view, the efficacy of our model was assessed using Mathew correlation, Cohen's Kappa coefficients, and a variety of common metrics. The model achieved an accuracy of 98.7% and precision of 98.81%. The results of the experiments demonstrate that the suggested system is capable of accurately describing typical activities within fog nodes and recognizing a variety of attack types, including DDoS, Port Scan, DoS GoldenEye, DoS Hulk, and DoS Slowhttp. The experimental findings demonstrate that the proposed system can identify various assaults and describe the usual activity occurring among fog nodes.

Conflict of interest

The authors do not have any type of conflict of interest to declare.

Funding

The authors did not receive any sponsorship to carry out the research reported in the present manuscript.

References

Abeshu, A., & Chilamkurti, N. (2018). Deep Learning: The Frontier for Distributed Attack Detection in Fog-to-things Computing. *IEEE Communications Magazine, 56*(2), 169-175. http://dx.doi.org/10.1109/MCOM.2018.1700332

Badr L., & Samma H. (2022). Optimized Deep Autoencoder Model for Internet of Things Intruder Detection. *IEEE Access*, *10*, 8434-8448. https://doi.org/10.1109/ACCESS.2022.3144208

Banaamah, A. M., & Iftikhar A. (2022), Intrusion Detection in IoT Using Deep Learning. *Sensors, 22*(21), 8417. https://doi.org/10.3390/s22218417

Benrazek A. E., Kouahla Z., Farou B., Ferrag M. A., Seridi H., & Kurulay M. (2020). An Efficient Indexing for Internet of Things Massive Data based on Cloud-fog Computing. *Transactions on Emerging Telecommunications Technologies*, *31*(3), 2161-3915. https://doi.org/10.1002/ett.3868

Canadian Institute for Cybersecurity (2017). *Intrusion Detection Evaluation Dataset* [Data set]. https://www.unb.ca/cic/datasets/ids-2017.html

Ferrández-Pastor, F.J., Mora H., Jimeno-Morenilla, A., & Volckaert, B. (2018). Deployment of IoT Edge and Fog Computing Technologies to Develop Smart Building services. *Sustainability, 10*(11), 3832. https://doi.org/10.3390/su10113832

Gudla, S. P. K., Bhoi, S. K., Nayak, S. R., Singh K. K., Verma A., & Izonin I., (2022). A Deep Intelligent Attack Detection Framework for Fog-Based IoT Systems. *Computational Intelligence and Neuroscience, 2022*(1). https://doi.org/10.1155/2022/6967938

Hassen, H. B., Ayari, N., & Hamdi, B. (2020). A Home Hospitalization System based on the Internet of Things, Fog Computing and Cloud Computing. *Informatics in Medicine Unlocked*, *20*, 100368. https://doi.org/10.1016/j.imu.2020.100368.

Huang, Z., Xu, W., & Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991.* https://doi.org/10.48550/arXiv.1508.01991

Hussain, M.M., & Beg, M.M.S. (2019). Fog Computing for Internet of Things (IoT)-Aided Smart Grid Architectures. *Big Data and Cognitive Computing, 3*(1), 8. https://doi.org/10.3390/bdcc3010008

Ijaz, M., Li, G., Lin, L., Cheikhrouhou, O., Hamam, H., & Noor, A. (2021). Integration and Applications of Fog Computing and Cloud Computing Based on the Internet of Things for Provision of Healthcare Services at Home. *Electronics*, *10*(9), 1077. https://doi.org/10.3390/electronics10091077

Jairu P., & Mailewa A. B. (2022). Network Anomaly Uncovering on CICIDS-2017 Dataset: A Supervised Artificial Intelligence Approach. In *Proceedings of the IEEE International Conference on Electro Information Technology (eIT).* (pp. 606-615). IEEE. https://doi.org/10.1109/eIT53891.2022.9814045

Kasongo, S. M., & Yanxia S. (2020). A Deep Learning Method with Wrapper based Feature Extraction for Wireless Intrusion Detection System. *Computers & Security, 92*, 101752. https://doi.org/10.1016/j.cose.2020.101752.

Khan, M. A., Khattak, M., Jan, S. U., Ahmad, J., & Jamal, S. S., Shah, A., Pitropakis, N., & Buchanan, W. (2021). A Deep Learning-based Intrusion Detection System for MQTT Enabled IoT. *Sensors, 21*(21), 7016. https://doi.org/10.3390/s21217016

Koroniotis N., Moustafa N., Sitnikova E., & Turnbull B. (2019). Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset. *Future Generation Computer Systems*, *100*, 779-796. https://doi.org/10.1016/j.future.2019.05.041

Kumar, V., Das, A.K., & Sinha, D. (2021). UIDS: a Unified Intrusion Detection System for IoT Environment. *Evolutionary Intelligence, 14*(1), 47-59. https://doi.org/10.1007/s12065-019-00291-w

Kumar, P., Gupta G. P., & Tripathi R., (2021). A distributed ensemble design based intrusion detection system using fog computing to protect the internet of things networks. *Journal of Ambient Intelligence and Humanized Computing, 12*(10), 9555-9572. https://doi.org/10.1007/s12652-020-02696-3

Kurniabudi, Stiawan D., Darmawijoyo, Idris M. Y. B., Bamhdi A. M. & Budiarto R. (2020). CICIDS-2017 Dataset Feature Analysis with Information Gain for Anomaly Detection. *IEEE Access*, *8*, 132911-132921. https://doi.org/10.1109/ACCESS.2020.3009843

Liao S., Wu J., Mumtaz S., Li J., Morello R., & Guizani M. (2020). Cognitive Balance for fog Computing Resource in Internet of Things: An Edge Learning Approach. *IEEE Transactions on Mobile Computing, 21*(5), 1596-1608. https://doi.org/10.1109/TMC.2020.3026580

Mohammad S. H., Abdul M., & Abu N. B. (2012). An Implementation of Intrusion Detection System using Genetic Algorithm. *International Journal of Network Security & Its Applications (IJNSA), 4*(2), 109-120. http://dx.doi.org/10.5121/ijnsa.2012.4208

Onah, J., Abdulhamid, S., Abdullahi, M., Hayatu H. I., & Al-Ghusham, A. (2021). Genetic Algorithm based Feature Selection and Naïve Bayes for Anomaly Detection in Fog Computing Environment. *Machine Learning with Applications, 6*. 100156. https://doi.org/10.1016/j.mlwa.2021.100156

Pajouh H. H., Javidan R., Khayami R., Dehghantanha A., & Choo K. K. R. (2016). A Two-layer Dimension Reduction and Two-tier Classification Model for Anomaly-based Intrusion Detection in IoT Backbone Networks. *IEEE Transactions on Emerging Topics in Computing, 7*(2), 314-323. https://doi.org/10.1109/TETC.2016.2633228

Ramkumar M. P., Daniya, T., Mano P., & Rajakumar, S. (2022). Intrusion Detection using Optimized Ensemble Classification in Fog Computing Paradigm. *Knowledge-Based Systems*, *252*(2), 109364. https://doi.org/10.1016/j.knosys.2022.109364

Raza, S., Linus W., & Thiemo V. (2013). SVELTE: Real-time Intrusion Detection in the Internet of Things. *Ad Hoc Networks, 11*(8), 2661-2674. https://doi.org/10.1016/j.adhoc.2013.04.014

Reddy D. K. K., Behera H. S., Nayak J., Naik B., Ghosh U., Sharma P. K., (2021). Exact Greedy Algorithm based Split Finding Approach for Intrusion Detection in Fog-enabled IoT Environment. *Journal of Information Security and Applications*, *60*. 102866.
https://doi.org/10.1016/j.jisa.2021.102866

Roy S., Li J., Choi B., & Bai Y. (2022). A Lightweight Supervised Intrusion Detection Mechanism for IoT Networks. *Future Generation Computer Systems, 127*(2), 276-285.
https://doi.org/10.1016/j.future.2021.09.027

Salo, F., Ali B., & Aleksander E. Dimensionality Reduction with IG-PCA and Ensemble Classifier for Network Intrusion Detection. (2019). *Computer Networks, 148*, 164-175.
https://doi.org/10.1016/j.comnet.2018.11.010

Shanker, R., Agrawal, P., Singh, A., & Bhatt, M. W. (2023). Framework for Identifying Network Attacks through Packet Inspection using Machine Learning. *Nonlinear Engineering*, *12*(1), 20220297.
https://doi.org/10.1515/nleng-2022-0297

Shandilya, S. K., Ganguli, C., Izonin, I., & Nagar, A. K. (2023). Cyber Attack Evaluation Dataset for Deep Packet Inspection and Analysis. *Data in Brief*, *46*, 108771.
https://doi.org/10.1016/j.dib.2022.108771

Sharma, B., Sharma, L., Lal, C., & Roy, S. (2023). Anomaly based Network Intrusion Detection for IoT Attacks using Deep Learning Technique. *Computers and Electrical Engineering*, *107*, 108626.
https://doi.org/10.1016/j.compeleceng.2023.108626

Samy, A., Yu, H., & Zhang, H. (2020). Fog-based Attack Detection Framework for Internet of Things using Deep Learning. *IEEE Access, 8*, 74571-74585.
http://dx.doi.org/10.1109/ACCESS.2020.2988854

Suhaimi, H., Suliman, S., Musirin, I., Harun, A., & Mohamad, R. (2019). Network intrusion detection system by using genetic algorithm. *Indonesian Journal of Electrical Engineering and Computer Science*, *16*(3), 1593-1599.
http://doi.org/10.11591/ijeecs.v16.i3.pp1593-1599

Tong J., Luo J., Pan H., Zheng J., & Zhang Q. (2020). A Novel Cuckoo Search Optimized Deep Auto-Encoder Network-Based Fault Diagnosis Method for Rolling Bearing. *Shock and Vibration*, *2020*(1).
https://doi.org/10.1155/2020/8891905

Ullah A., Muhammad K., Haq I. U., & Baik S.W. (2019). Action Recognition using Optimized Deep Autoencoder and CNN for Surveillance Data Streams of Non-stationary Environments. *Future Generation Computer Systems, 96*, 386-397.
https://doi.org/10.1016/j.future.2019.01.029

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access*, *7*, 41525-41550.
https://doi.org/10.1109/ACCESS.2019.2895334

Vinayakumar, R., Soman, K.P., Poornachandran, P., Alazab, M., Jolfaei, A. (2019). DBD: Deep Learning DGA-Based Botnet Detection. In: Alazab, M., Tang, M. (eds) *Deep Learning Applications for Cyber Security*. Advanced Sciences and Technologies for Security Applications. Springer, Cham.
https://doi.org/10.1007/978-3-030-13057-2_6

Yazan O., Liu D., & Nayak A. (2022). DL-IDS: a Deep Learning–based Intrusion Detection Framework for Securing IoT. *Transactions on Emerging Telecommunications Technologies*, *33*(3).
https://doi.org/10.1002/ett.3803

Yin Y., Jang-Jaccard J., Sabrina F., & Kwak J., (2023). Improving Multilayer-Perceptron (MLP)-based Network Anomaly Detection with Birch Clustering on CICIDS-2017 Dataset. In *Proceedings of the 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD).* (pp. 423-431). IEEE.
https://doi.org/10.1109/CSCWD57460.2023.10152640

Zeeshan M., Riaz Q., Bilal M. A., Shahzad M. K., Jabeen H., Haider S. A., & Rahim A. (2021). Protocol-based Deep Intrusion Detection for DoS and DDoS Attacks using UNSW-NB15 and Bot-IoT Data-sets. *IEEE Access*, *10*, 2269-2283.
https://doi.org/10.1109/ACCESS.2021.3137201

Zhou Q., Yong B., Lv Q., Shen J., & Wang X. (2020). Deep Autoencoder for Mass Spectrometry Feature Learning and Cancer Detection. *IEEE Access, 8*, 45156-45166.
https://doi.org/10.1109/ACCESS.2020.2977680

Zhu, Y., & Nasser Y. (2021). Optimal Parameter Identification of PEMFC Stacks using Adaptive Sparrow Search Algorithm. *International Journal of Hydrogen Energy*, *46*(14), 9541-9552.
https://doi.org/10.1016/j.ijhydene.2020.12.107