SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V11I11P111

Volume 11 Issue 11, 118-132, November 2024
© 2024 Seventh Sense Research Group®

Original Article

Attention-Based BI-LSTM Model to Detect Botnet Attacks over Internet of Things (IoT) Environments

Swapna Thota[1], D. Menaka[2]

[1,2]Noorul Islam Centre for Higher Education, TamilNadu, India.

[1]Corresponding Author : swapnathota@gmail.com

Received: 12 September 2024 Revised: 11 October 2024 Accepted: 10 November 2024 Published: 03 December 2024

Abstract* - *Detecting an IoT-botnet attack involves monitoring network traffic, identifying unusual behaviour, and implementing security measures to prevent and mitigate the impact of the attack. These days, hackers use botnets, a network of computational devices, to illegally access distributed resources and launch cyber-attacks against the "Internet of Things (IoT)". A variety of "Machine Learning (ML) and Deep Learning (DL") techniques have recently been developed to identify botnet assaults in IoT networks. The six main stages of the proposed paradigm are Botnet Attack Mitigation, Feature Extraction, Feature Selection, and Data Augmentation. First, data cleaning and data normalization (min-max normalization) are used to preprocess the raw data that has been gathered. The "Synthetic Minority Oversampling Technique (SMOTE)" method is then used to enrich the pre-processed data to address the class imbalance problem. Then, the supplemented data retrieved characteristics like Measure of Dispersion (Skewness, Variance, IQR), Central tendency (Generalized mean, Winsorized mean, Median, standard deviation, and variance), and Information Gain. The best features are selected using the extracted features. CUGOA stands for Clan Updated Grasshopper Optimization Algorithm, a hybrid optimization model. The Grasshopper Optimization Algorithm (GOA) and Elephant Herding Optimization (EHO) are combined to create the proposed CUGOA model. Next, the DCNN, Attention-based Bi-LSTM, and optimized RNN are all included in the new ensembled-deep-learning model, which detects Botnet Attacks. The chosen optimal features are used to fine-tune the DCNN and Attention-based Bi-LSTM. The improved RNN model receives the output of DCNN and Attention-based Bi-LSTM as input. The final detected outcome regarding the presence/ absence of a botnet attack is acquired from the optimized RNN model, whose bias function is fine-tuned using the new Hybrid optimization model. Once the attacker is found to be present in the network, it is mitigated using the new Botnet Traffic Filter (BTF). Thus, the network becomes highly reliable. The proposed model outperforms existing models regarding "accuracy, sensitivity, specificity, and precision".

Keywords* - *Botnet, Internet of Things, Deep Learning cyber security, Intrusion detection.

1. Introduction

Hackers have been attacking IoT devices a lot in the past few years. As a result, creating and implementing an efficient detection system is required. Due to the advent of new malware variants, even with a large number of prior detection systems, it is insufficient to identify every sort of assault adequately. They are often divided into two groups, such as anomaly-based detection systems and abused systems.

A host-based detection system and a network-based detection system can classify it in accordance with the detection architecture. With 50 billion devices expected to be connected to the Internet by 2020, the Internet of Things (IoT) is expected to usher in a new era of increased connectivity [1]. The IoT's primary goal is to link previously disconnected objects to the Internet [2] to create intelligent gadgets that can gather Data that can be stored and shared without the need for human intervention [3]. Many of these Internet of Things Devices are marketed to customers who prioritise affordability and ease of use over security. These market forces have compelled IoT manufacturers to innovate and produce vast quantities of unsafe "Internet-connected devices", like "IP cameras and Digital Video Recorder (DVR) boxes", while leaving out essential security measures. Using default credentials, unsecured protocols, and inherent computational limits are frequently the sources and symbols for such vulnerabilities and exploits. An ever-growing pool of attack resources is available due to the fast spread of unsecured IoT devices and the simplicity with which attackers may find them utilizing web services like shodan [4]. Attackers can now carry out massive attacks against Internet resources, including "spamming, phishing, and Distributed Denial of Service (DDoS)", by exploiting and compromising a huge number of these vulnerable IoT devices [5]. Likely, the growth in DDoS attacks based on IoT will continue unless IoT manufacturers take ownership and build security features into their products.

This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

Until then, there are several difficulties with the IoT since it could become the new cyber-attack hotspot. The problems previously mentioned, As more “DDoS attacks”, many of these Internet of Things devices are aimed at consumers who take priority affordability and ease of use over security. Because of these market forces, IoT manufacturers have been forced to innovate. This problem is exacerbated by the fact that many consumer IoT devices lack a user interface, making it nearly impossible for users to identify and be aware of attacks on home networks. The detection of botnets in IoT networks is a classification problem. [6] Each sample in a network traffic packet is categorized as harmless or malicious based on a set of predetermined criteria using binary classification. On the other hand, multi-class classification identifies the precise kind of botnet assault. Currently, AI algorithms have illustrated strong performance in classification tasks across many application domains, including, for example, voltage stability evaluation in power systems [7].

Several IoT networks categorise network traffic data, and "Machine Learning (ML) and Deep Learning (DL)" models have been developed. Using various architectures, such as “Random Forest (RF), Support Vector Machine (SVM), Deep Neural Network (DNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU)”, These models learn well how distinguish between harmless as well as malicious traffic presents thorough evaluations as well as surveys on machine learning and deep learning applications throughout intrusion detection to gain a detailed understanding. It is challenging to classify data from extremely unbalanced network traffic. When there are more than 1:10 examples when compared to samples from the minority group, the data are considered to be severely unbalanced [8].

In minority classes, High-class imbalance degrades both Machine Learning and Deep Learning model classification performance. Small “disjunctions, noise, and overlap in network traffic samples”, in addition to the class imbalance issue, can potentially cause poor classification performance [9]. The volume of malicious traffic created in common botnet attack scenarios, including DDoS, is Typically, the amount of harmless traffic generated by genuine devices in an IoT network is far greater. This indicates that there are much fewer samples in the normal class than there are in the attack class.

Modern DL models in this situation have the propensity to favor the majority (of the attack) class, which raises the “False Positive (FP) rate” [10]. Deploying cutting-edge application of “ML and DL” models in real-world IoT networks cause a significant portion of network traffic data to be misclassified from minority classes, which may result in privacy violations, the loss of sensitive data, lost revenue from unavailable applications and services, and even fatalities in critical IoT systems.

The key contribution of this research work is:

- To select the optimal features using the new hybrid optimization model. The prospect hybrid optimization model- Clan Updated Grasshopper Optimization Algorithm (CUGOA) is the combination of the Elephant Herding Optimization (EHO) and Grasshopper Optimization Algorithm (GOA).
- To introduce a new ensembled-deep-learning model for IoT-BOTNET attack detection with DCNN, Attention-based Bi-LSTM, and optimized RNN.
- To fine-tune the bias function of the RNN model using the new hybrid optimization model.

2. Related Work

An effective technique for identifying botnet attacks is Deep Learning (DL). However, the amount of memory space and network traffic needed is typically substantial. Therefore, deep learning approaches are extremely difficult to implement in low-memory IoT devices. In this study, We reduce the feature dimensionality of the "Long Short-Term Memory Auto encoder" during the encoding phase of “large-scale IoT network traffic data (LAE)” [11]. This study investigates the problem of estimating multiple pitches and Multichannel harmonic sinusoidal signal Directions-of-Arrival (DOA). After defining a uniform linear array spatiotemporal matrix signal model, the ESPRIT method, which is focused on subspace techniques and takes advantage of time domain invariance, is used to calculate the multi-pitch frequencies of a number of harmonic signals [12]. We have seen exceptional productivity as a result of the large number of Internet of Things (IoT) applications that are widely used and have made daily life easier.

Because these technologies are insecure, vulnerable computers can be found all around us in our daily lives, allowing the Internet of Things to be used to launch numerous attacks using massive botnets. These atrocities' heinous goals have also been achieved. To keep devices safe, a solid identification strategy is essential. The purpose of this research is to develop and deploy a model for anomaly-based intrusion detection in IoT networks that uses a "Convolutional Neural Network (CNN)" and a “Gated Recurrent Unit (GRU)” to identify and classify binary and multiclass IoT network data. The suggested model is validated using the “BoT-IoT, IoT Network Intrusion, MQTT-IoT-IDS2020, and IoT-23” datasets. The “accuracy, precision, recall, and F1 score” of our proposed binary and multiclass classification model would be extremely high [13]. The proposed feature extraction and classification phases of this detection model are divided into two steps. In the first phase, known as feature extraction, each application's data is processed by combining its existing features with statistics and higher-order statistical features. Based on these retrieved features, the classification technique is developed using an optimal “Deep Convolutional Neural Network (DCNN)” model. A new “Modified Algorithm of the

Innovative Gunner (MAIG)", an improved version of the AIG method, is used to optimise the convolution layer's number of filters and filter size, as well as the activation function [14]. The "Internet of Things (IoT)" is increasingly many of these devices have Internet access, but many of them lack basic security, leaving the Internet vulnerable to a variety of attacks. Mirai, as well as other botnets, have launched "Distributed Denial of Service (DDoS)" attacks against critical Internet infrastructure using insecure consumer IoT devices [16]. The number of "Internet of Things (IoT)" devices deployed globally is rapidly increasing, and the frequency of such attacks has reached an all-time high; it is critical to quickly identify "Distributed Denial-of-Service (DDoS)" attacks in order to mitigate security threats. By accelerating warnings and disconnections originating from a network of infected IoT devices, instant identification helps network security by halting the spread of the botnet and averting further attacks [17].

"Signature-based detection" and "anomaly-based detection" are the two detection methods. For attacks with a known pattern, "signature-based detection" is used. "Anomaly-based detection" is utilized concurrently for both unidentified and recognized attack patterns. Besides that, "Network-based Intrusion Detection Systems (NIDS)" rely on traffic identification, that is, the flow of information. Machine learning algorithms are capable of extracting critical characteristics required to classify traffic records as malicious or legitimate activity [18]. Due to its extensive deployment and difficulties, the "Internet of Things (IoT)" has grown to be a particularly prominent subject of research [22]. Security, though, is the main issue as its scale and applications continue to expand quickly.

It is a laborious process to individually install security features in each IoT device and to update them in response to evolving threats. Furthermore, machine learning models are best suited to exploiting massive amounts of data created by IoT devices [21]. Numerous DL-based approaches have been proposed in recent literature to detect assaults in IoT; Botnet detection in IoT networks has been proposed by researchers in a variety of ways using machine language and deep learning model architectures. Although LSTM was proposed [15], No experiment was conducted to validate the effectiveness of the suggested "DDoS attack detection strategy in web servers". Because the GRU model performed better than the "Artificial Neural Network (ANN) and the Long Short-Term Memory (LSTM) models".

The performance evaluation, on the other hand, had been solely based on accuracy [19] suggested Stacked RNN (SRNN), which cascades different RNN layers as opposed to the "k-Nearest Neighbour (kNN), Logistic Regression (LR), SVM, Multi-Layer Perceptron (MLP), and Decision Tree (DT) models, the RF model" outperformed them all. "The Edge-based Graph Sample and Aggregate (E-GraphSAGE) model" was suggested, and it performed better than the DT and "Extreme Gradient Boosting (XGBoost) models" [19] because it outperformed the "RF, Extra Tree (ET), Gradient Boost (GB), and XGBoost models". The "convolutional Neural Network (CNN) model" fared better than the "RNN, LSTM, and GRU" models [20]. An edge-cloud DNN model with a low level of complexity was proposed, and it outperformed the "kNN, DT, RF, and SVM models" in terms of performance.

3. Proposed Iot-Botnet Attack Detection and Mitigation Method

The following stages are part of the proposed model: [Figure 1]

Step : 1 Data Pre-processing: the collected raw data $S_i^{inp}; i = 1,2, ... N$ is cleaned and normalized using data cleaning and min-max normalization techniques. The data acquired after pre-processing is denoted as S_i^{pre}.

Step : 2 Data Augmentation: The "Synthetic Minority Oversampling Technique (SMOTE)" is used to deal with the issue of class imbalances S_i^{pre}. The augmented data is denoted as S_i^A.

Step : 3 Feature Extraction: the features like Central tendency (The generalised mean, the Winsorized mean, the median, the standard deviation, and the variance), Measure of Dispersion (Skewness, Variance, IQR), and Information Gain are extracted from the augmented data S_i^A. The extracted features are denoted as F_i.

Step : 4 Feature Selection: Then, from the extracted features F_i, the optimal features are selected using the new "Hybrid optimization model (Elephant Herding Optimization (EHO) and Grasshopper Optimization Algorithm (GOA))". The extracted optimal features are denoted as F_{i*}.

Step : 5 Botnet attack detection via Ensembled-Deep-Learning-Model: The ensembled-deep-learning-model consists of three deep learning models: DCNN, Attention-based Bi-LSTM, and optimized RNN. The DCNN and Attention-based Bi-LSTM are trained using the selected optimal features F_{i*}. The outcome acquired from DCNN is Out^{DCNN} Attention-based Bi-LSTM $Out^{Bi-LSTM}$. They are fed as input to optimized RNN, whose bias function is fine-tuned using the Hybrid optimization model. The final detected outcome regarding the presence/absence of a botnet attack is acquired from the optimized RNN.

Step : 6 Botnet Attack Mitigation: If the presence of a botnet attack is detected, then it is mitigated using the new Botnet Traffic Filter (BTF); otherwise, the data is routed via the normal data routing mechanism.

Step : 7 Evaluation: Evaluate the proposed model using various evaluation metrics such as accuracy, sensitivity, specificity, and precision.

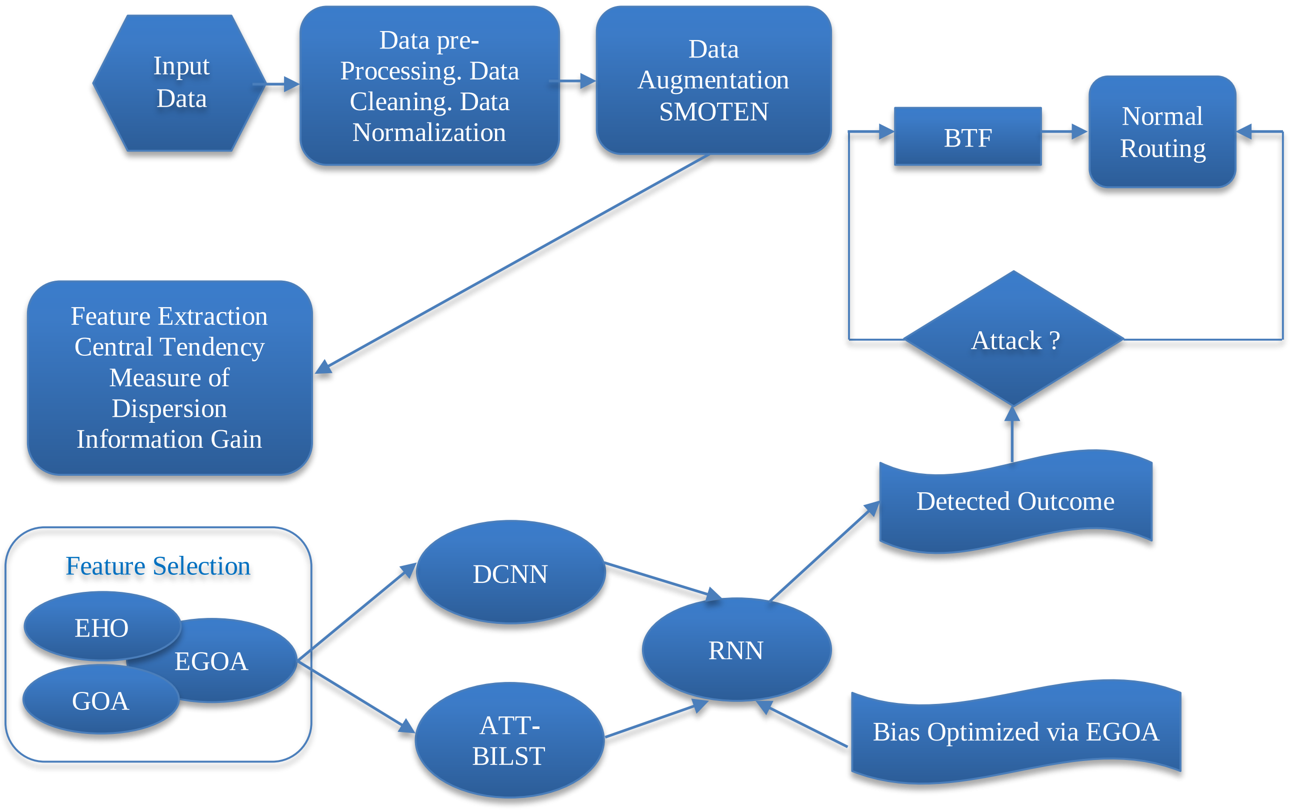

Fig. 1 Overall proposed diagram

3.1. Data Pre-Processing

Preprocessing refers to transforming raw data S_i^{inp} into a format suitable for analysis and modeling. In this research work, the data is pre-processed via data cleaning and data normalization.

3.1.1. Data Cleaning

The input to "data cleaning" is the raw data S_i^{inp}. The goal of data cleaning, a crucial step in data preparation, is to identify and correct errors, inconsistencies, and missing values in a dataset. This step is essential to guarantee the accuracy and dependability of the data used for analysis and decision-making. Outlier detection, imputation of missing values, and duplication removal are common tasks involved in data cleaning, which can be carried out manually or with the aid of automated tools. The data acquired after the cleaning mechanism is denoted as S_i^{clean}.

3.2. Data Normalization

Data normalization is a common pre-processing technique in machine learning that brings different features to a similar scale. Normalization is applied to the collected data in the context of IoT-BOTNET attack detection to ensure that all features are on the same scale and to avoid issues caused by different magnitudes of data. The normalization process is performed on the raw data before any further analysis or feature extraction is performed. The network traffic values S_i^{clean} were scaled to have values between 0 and 1, as determined. Equation (1) describes the min-max normalisation method.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

Where x is a feature vector representing the network traffic of S_i^{clean}, and x_{min} and x_{max} represents its minimum and maximum values, respectively. An additional dimension representing the facilitation of the creation of the DRNN model, a time step of one unit (i.e., t = 1), was added to the feature set. The data acquired after data normalization is denoted as S_i^{pre}.

3.3. Data Augumentation

The process of adding new, synthetic data samples to an existing dataset to increase its size and diversity is known as data augmentation. In this research, the"Synthetic Minority Oversampling Technique (SMOTE)" is used to address the problem of class inequality in S_i^{pre}.

Synthetic Minority Over-sampling Technique (SMOTE) is used in machine learning as a data augmentation technique to focus on the problem of class disparity. In the context of IoT-BOTNET attack detection, the class imbalance problem arises because botnet traffic is often much less prevalent than normal traffic in the dataset. SMOTE generates synthetic data points for the minority class (in this case, the botnet traffic) by interpolating between neighboring data points. This creates new representative data points and balances the number of samples in each class.

3.3.1. "Synthetic Minority Oversampling Technique"

In the case of an 11-class classification, the training set's high-class imbalance problem was addressed using the SMOTE algorithm in contrast to the method used in this study, which uses replacement to oversample minority classes. The method used in this study generates synthetic examples using strategies such as rotation and skew to achieve class balance. These fictitious data on network traffic were generated in the following manner connecting the minority classes' k closest neighbors, where k = 3. As a result, neighbors from the three closest neighbors were chosen randomly. Algorithm 1 presents the SMOTE process step-by-step. The number of "minority class samples (T), oversampling rate (N%), and number of nearest neighbours all had an effect on the production of synthetic samples (S) in the minority classes (k)".Minority class samples are randomized if N is less than 100%. We only calculate the k nearest neighbors for each minority class. The current sample of The integral multiples of 100 in $N(j)$, the minority class I, and an array of randomly generated integers (an array) are all functions of N. Z is V is a set of synthetic samples, whereas A is a set of original minority class samples. The augmented data is denoted as S_i^A.

"Algorithm I: SMOTE algorithm"

```
Input: T,N,K
Output: S
Initialization: k=3,q=37,r=0
If N<100 then
    Randomise the T minority class samples
    S=(N x 100) x T
    N=100
end
N=(j)(N/100)
For i=1 to T do
    Compute k nearest neighbours for i
        While N 6≠ 0 do
            nn=random(1,k)
                for c=1 to q do
                    f=Z[nn_arry[nn]][c]-Z[i][c]
                    g=random(0,1)
                        V[r][c]=Z[i][c]+(gxf)
                end
                r=r+1
                N=N-1
                end
end
```

3.4. Feature Extraction

Features like Central tendency (Generalized mean, Winsorized mean, Median, Standard deviation, variance), Measure of Dispersion (Skewness, Variance, IQR), and Information Gain, are extracted from the augmented data S_i^A.

3.4.1. Central Tendency

A statistical characteristic known as central tendency indicates where a dataset's center or middle is located. In other words, it measures a data set's typical or average value. By identifying a single value that serves as the data center, the central tendency can be used to summarise and describe the distribution of a dataset.

Mean

The mean is the most commonly used and well-known measurement method for"central tendency (or average)". It can be used with discrete and continuous data but is most often used in continuous data. Divide the total number of values in the data set by the number of values in the data set to determine the mean. As a result, if a data set contains values, the sample mean, as shown by Equations (2) and (3).

$$\bar{a} = \frac{a_1+a_2+\cdots+a_m}{m} \tag{2}$$

Typically, this formula is written slightly differently and begins with the "Greek capital letter", which means "sum of..." and is pronounced "sigma."

$$\bar{a} = \frac{\sum a}{m} \tag{3}$$

Median

The median is the middle score in a data set sorted by magnitude. Outliers and inconsistencies have less of an impact on the median. If the number of values or observations is odd, the $[(m+1)/2]th$ observation is used to calculate the median. If the number of observations or values is even, the median is determined by averaging the $(m/2)th$ and $[(m/2)+1]th$ observations in the given data set. The formula below can be used to determine the median for grouped data. This can be mathematically given as per Equation (4).

$$median = k + \left(\frac{\frac{M}{2}-du}{u}\right) \times g \tag{4}$$

Winsorized Mean

A central tendency measurement that is less susceptible to outliers than the standard mean. The highest and lowest values from a data set are swapped out for less extreme values to produce a Winsorized mean. The values in the modified set are then added up, and the average is determined. This can be mathematically given as per Equation (5).

$$Winsorized\ Mean = \frac{a_m \cdots a_{m+1} + a_{m+2} \cdots a_m}{M} \tag{5}$$

Here, M is the overall number of data points.

Standard Deviation

The standard deviation attempts to measure the dispersion or spread of a set of data values around a central tendency (mean, median, or model). It evaluates the degree of variation or departure from the mean. The standard deviation is low when the values are close to the central value; when it is high, the values are more dispersed. This can be mathematically given as per Equation (6).

$$\sigma = \sqrt{\frac{\sum_{j=0}^{m}(a_j - \bar{a})^2}{m}} \tag{6}$$

Here, a_j is one of the data points, $\bar{a}$ is the mean value and m is the data point total number. The extracted Central tendency-based features are denoted as f^{cen}.

3.4.2. Measure of Dispersion

A statistical term known as "measure of dispersion" is a term that describes the range or variation of a collection of data values. It measures how much a dataset's values deviate from its central tendency (mean, median, or mode). The terms "range, interquartile range, variance, and standard deviation" are some examples of "measures of dispersion". These measurements help identify outliers and skewness in the data and provide crucial information about the distribution of the data.

Skewness

"Skewness" is a measure of an asymmetric or distorted distribution. This is used to calculate how far the distribution of a given random variable deviates from a symmetric distribution, such as the normal distribution. A normal distribution has no skewness because it is symmetric on both sides.

There are several methods for calculating skewness, but the two most common are "Pearson mode skewness and Pearson median skewness". The "Pearson mode skewness" is used when the sample data has such a strong mode. If the data has more than one mode or a weak mode, "Pearson's median skewness" has been used.

"Pearson mode skewness" calculation is shown in Equation (7).

$$Skewness = \frac{\bar{A} - N_p}{t} \tag{7}$$

As per Equation (7), A is the mean value, $N_p$is the mode value and t is the sample data's "standard deviation".

Variance

Deviation from the mean squared is described as variance. Measures the deviation of each data point in a dataset from the mean. This can be mathematically given as per Equation (8).

$$Variance = t^2 = \frac{\Sigma(a_j - \bar{a})^2}{m-1} \tag{8}$$

Equation (8) $\bar{a}$ is the mean sample, where m denotes the total number of data points.

Interquartile Range

The middle 50% of the data values in a dataset are spread across the "Interquartile Range (IQR)", a measure of dispersion. It represents the middle half of the data range and is calculated as the difference between the data's 75th and 25th percentiles. In statistical modeling and data analysis, the IQR is a reliable measure of dispersion unaffected by outliers. This can be mathematically given as per Equation (9).

$$IQR = R3 - R1 \tag{9}$$

The extracted Measure of Dispersion based features is denoted asf^{dis}.

3.4.3. Information Gain

The reduction in entropy or uncertainty brought about by partitioning the data based on a particular feature is measured using the concept of information gain in decision tree learning and feature selection algorithms. It is computed as the difference between the weighted average of the entropies of the feature's partitions and the entropy of the original dataset. To create decision trees that successfully predict the target variable, the most informative features are found. This can be mathematically given as per Equation (10).

$$Gain(T, X) = Entrophy(T) - \sum_{f \in values(X)} \frac{T_f}{T}\, Entrophy(T_f) \tag{10}$$

As per Eqn. (10) values(X) is the all-possible values for the attribute X and T_f is the subset of T for which X has value f. The extracted features are denoted asf^{gain}. The extracted features are fused (f^{gain}+$f^{dis} + f^{cen}$). The extracted features are denoted as F_i.

3.5. Feature Selection

In a hybrid optimization model, feature selection entails selecting relevant features from a larger set of features. The best features will be chosen in this study using the new Hybrid optimization model, which is a hybrid of the "Elephant Herding Optimization (EHO) and the Grasshopper Optimization Algorithm (GOA)". The GOA Model is introduced within the EHO model in this research work. Grasshoppers are insects well-known for interfering with and harming agricultural and crop production. There are two stages in their life cycle: "nymph and adulthood". Long-distance and abrupt movements distinguish the adult phase, whereas small steps and slow movements distinguish the nymph phase.

Mathematically, the proposed model can be given as shown below:

Step 1: Initialization N- the count of search agents is initiated.
Step 2: Fitness Evaluation – The fitness of every search agent is computed as per the objective function of this research work. Mathematically, the fitness function can be given as $\mathrm{Obj} = \mathrm{Max}(A)$. , Here, A denotes the accuracy. Based on the computed fitness function, the search agents' agents
Step 3: Clan updating Operator-based GOA (proposed)- A matriarch from each clan serves as the head of the entire elephant population. Therefore, the matriarch j has an impact on the subsequent position of each elephant in clan ci. It can be changed as needed for the clan dj s elephant i. In this phase, the clan is updated based on the GOA swimming behavior. The following mathematical model illustrates grasshopper swarming behaviour. This can be mathematically given as per Equation (11).

$$O_j = T_j + H_j + X_j \tag{11}$$

Where O_j denotes the position of the $j - clan$, T_j refers to the social interaction between the matriarch and herd, Hj denotes the distance between the matriarch and herd. Eqn. (12) can be revised to read as follows to cause clans to behave randomly.

$$O_j = q_1 T_j + q_2 H_j + q_3 X_j \tag{12}$$

Where [0, 1] is the range for the random numbers q1,q2, and q3.

The following is a definition of social interaction between matriarch and herd Si. This can be mathematically given as per Eq. (13).

$$T_j = \sum_{\substack{i=1 \\ i\neq 1}}^{M} t(c_{ji})\widehat{c_{ji}} \tag{13}$$

N indicates the total number of grasshoppers, $c_{ji} = |O_j - O_i|$ indicates the "Euclidean" distance between $j\ and\ i$, and $\widehat{c_{ji}} = \frac{O_i - O_j}{c_{ji}}$

From the $j - th$ (monarch) to the $i - th$ elephant in the herd, c_{ji} is a unit vector, and s stands for the social forces created by Eq. (14).

$$t(q) = u\, exp^{\frac{-q}{k}} - exp^{-q} \tag{14}$$

Where k represents the attraction length scale and u represents the attraction intensity. Attraction and repulsion are two ways that elephants communicate with one another. the interaction of monarchy and herd regarding their personal space.

The following equation provides the distance H_j. This can be mathematically given as per Eq. (15).

$$H_j = -h\hat{e}_h \tag{15}$$

Where $\hat{e}_h$is h is the gravitational constant and a unit vector pointing towards the centre of the group.

The following equation yields the wind advection X_j. This can be mathematically given as per Equation (16).

$$X_j = f\hat{e}_b. \tag{16}$$

Where $\hat{e}_b$is a unit vector pointing in the wind's direction and u denotes the drift consistency t. The following equation can be obtained by changing the S, G, and A values.

$$C_j = \sum_{\substack{i=1 \\ i\neq 1}}^{M} t\left(\left|O_i - O_j\right|\right)\frac{O_i - O_j}{cji} - h\hat{e}_h + f\hat{e}_b \tag{17}$$

This equation is given in an improved form as

$$O_j^c = d\left(\sum_{\substack{i=1 \\ i\neq 1}}^{M} d\,\frac{ub_c - lb_c}{2}\ t\left(\left|O_j^c - O_i^c\right|\right)\frac{O_i - O_j}{c_{ji}}\right) + \hat{S}_c \tag{18}$$

Where the lower and the upper bounds of the c-th dimension are denoted by lb_c and ub_c respectively. The symbol represents the best solution found thus far in the $c - th$ dimension space.

$\hat{S}_c$.The repulsion zone, attraction zone, and comfort zone between monarchy and herd are reduced according to the number of iterations using the parameter $d2$. The following equation represents the parameters $d1$ and $d2$ as a single parameter. This can be mathematically given as per Equation (19).

$$d = d_{max} - s\,\frac{d_{max} - d_{min}}{s_{max}} \tag{19}$$

Where s is the current iteration and s_{max} is the maximum number of iterations possible, andd_{max} and d_{min} stand for d's maximum and minimum values, respectively.

A grasshopper's position is updated based on its present location, the world's best location, and the locations of other grasshoppers in the swarm. This makes it easier for elephants to avoid becoming stuck in local optima.

3.5.1. Separating Operator

When they reach puberty, male elephants in the elephant group will leave their family group and live alone. When attempting to solve optimization problems, this separating process can be modelled as a separating operator. Let's assume that the elephant individuals with the worst fitness will use the separating operator at each generation, as shown in Equation (20), to further enhance the searchability of the EHO method.

$$a_{worst,dj} = a_{min} + (a_{max} - a_{min} + 1) \times rand \quad (20)$$

Where the elephant individual's position's upper and lower bounds are, respectively, denoted by a_{max} and a_{min}. $rand \in [0,1]$ is a uniform stochastic distribution with a range of [0, 1].$a_{worst,dj}$ is the worst elephant member of clan ci. The EHO method is created according to the descriptions of the clan updating and separating operators, and its mainframe can be summarised.

- Termination- The best solution recorded is the optimal features. The optimal features are pointed out as F_{i*}.

3.6. IoT-BOTNET Attack Detection via Ensembled-Deep-Learning-Model

The ensembled-deep-learning model consists of three deep-learning models: DCNN, Attention-based Bi-LSTM, and optimized RNN. The DCNN and Attention-based Bi-LSTM are trained using the selected optimal features F_{i*}.

The outcome acquired from DCNN is Out^{DCNN}, Attention-based Bi-LSTM $Out^{Bi-LSTM}$. They are fed as input to optimized RNN, whose bias function is fine-tuned using the Hybrid optimization model. The final detected outcome regarding the presence/absence of a botnet attack is acquired from the optimized RNN.

3.6.1. “Deep Recurrent Neural Network”

“Recurrent Neural Network”features a hidden state, in contrast to Feedforward Neural Networks (FNN), which aids in modeling the temporal dynamics of input data. Equation (21) states that the RNN with trainable parameters transforms Xk denotes the temporal dynamics of a mini-batch of highly unbalanced network traffic features extracted from input data and an initial hidden state:

$$h_{1k} = \sigma_h(w_x x_k + w_h h_{init} b_h) \quad (21)$$

When RNN is trained using the kth mini-batch, the weights Wx and Wh are now applied for the linear transformation of Xk and hint; The bias is bh and $h1k$ is the new hidden state. Equations (22)-(23) are applied to the RNN layer output to produce the “Deep Recurrent Neural Network” layer output.

The presentation of “Deep Recurrent Neural Network” information in Algorithm 2 is comprehensive. Equation (22) is used to determine the four dense hidden layers' hidden states:

$$h_{mk} = \sigma_h(w_{hm} h_{(m-1)} + b_{mh} \quad (22)$$

Where $h1k = hk, hmk$is the hidden layer's mth hidden state, and $m = [2,3,4,5]$; The weight to significantly modify the previously hidden state, h(m1)k, linearly is Whm, while the bias of the mth hidden layer is $bmh. and\ sh$ is an activation function for a “Rectified Linear Unit (ReLU)”, as shown by Equation (26):

$$\sigma_h\ (a) = max(0, a) \quad (23)$$

The function returns 0 if an is a negative number; otherwise, it returns the same a when an is a positive number.

"Algorithm 2: DRNN algorithm".

```
Input: X
Target: y
Output: y
$H_0 = h_{int}$
for e 1 to u do
for k= 1 to n do
$h_{1k} =\sigma_h(W_x 1x_k+W_h lh_0 + b_{1h})$
for m = 2 to (d + 1) do
$h_{mk} =\sigma h(W_y h_{mk}(m-1)k+b_{mh})$
end
$\underline{yk} = \sigma y(W_y h_{mk} + by)$
$Lk =\theta(y_k, \underline{yk})$
End
$L =\frac{1}{n}\sum_{k=1}^{n} L_k$
W(.), b'(.)=ϕ(w(), b())
end
```

According to Eqs. (24), the dense output layer transforms the fourth dense layer's hidden state $h5k$:

$$yk = \sigma y(Wyh5k + by) \quad (24)$$

Where n $i$ndicates the X mini-batch sample size, “$n = p/\mu$ is the batch size, and $\mu = 512$"; Wy is the weight applied to h_{5k}'s linear transformation; b_y as the σy are the dense output layer's activation function and bias, respectively. σy is a softmax function in the scenario of multi-class classification and is denoted by Equation (25):

$$\underline{y}_{k}=\frac{e^{(W_y h_{5k}+b_y)}}{\sum_{w=1}^{r} \quad e^{(W_y h_{5k}+b_y)}} \tag{25}$$

γ is the number of classes present in y, and Equation (26) the categorical "cross-entropy loss function" (θc)

calculates the difference between $\tilde{y}$ and y:

$$L=\theta_c(y_k, \underline{y}_k = -\frac{1}{n}\sum_{\tau=1}^{n} \ \sum_{w=1}^{\gamma} \ [y\tau\omega log(\underline{y}\tau, \omega) \tag{26}$$

To validate the DRNN's performance, previously unidentified highly unbalanced network traffic data, Xva, and the ground-truth labels associated with it yva were used. Adam's effective first-order stochastic gradient descent method can reduce training and validation losses in mini-batches spanning u epochs [8].

Equation (27) represents the densely connected DL model's trainable parameters.

$$\Phi = [w_{(.)}, h_{(.)}, b_{(.)}] \tag{27}$$

The Adam optimizer, φ, modifies Φ to minimize L for each epoch, as shown in Equation (28):

$$\Phi' = \varphi(\Phi, L, \alpha, \beta_1 \beta_2) \tag{28}$$

3.6.2. Convolutional Neural Network

The field of object recognition and prediction has given a lot of attention to CNN, an ANN-based on deep learning theory. Images with a 2D grid can automatically be processed using CNN to extract spatial characteristics. The fully connected layer, pooling layer, convolutional layer, and activation layer function make up the majority of CNN. Figure 2 depicts the CNN layer.

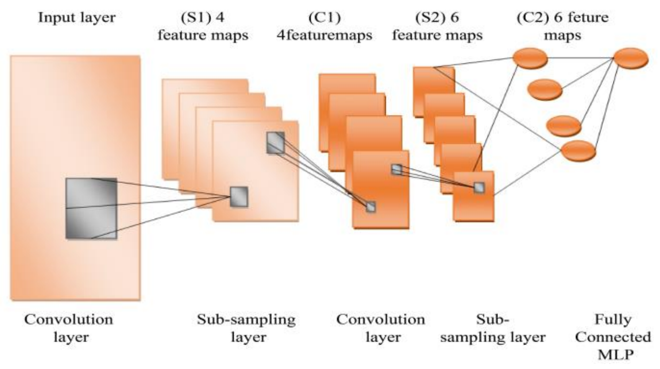

Fig. 2 CNN layer

Convolutional Layer

The convolutional layer's convolution process, which is used to extract visual information and learn the map between the input and output layers, replaces the matrix multiplication process in the traditional neural network.

By sharing parameters as a result of the convolutional process, the network can only learn one set of parameters, substantially lowering the number of parameters and improving computing performance. A convolution operation is described as per Eq. (29).

$$v_{i,j} = \sum_{q=0}^{C} \sum_{n=0}^{C} l_{q,n} s_{k+q,b+n} \tag{29}$$

Where $l_{q,n}$ is the convolutional kernel's scale, $s_{k,b}$ is the image's pixel value at k and b.

Activation Function

CNN frequently employs rectified linear unit activation functions to avoid vanishing gradients and speed up training. The following is a description of ReLU's goal. This is mathematically shown in Eqn. (30).

$$ReLU(r) = \begin{cases} r & r > 0 \\ 0 & r \leq 0 \end{cases} \tag{30}$$

Pooling Layer

While lowering the complexity of the computer network, the pooling layer can concentrate the specifics in feature maps. The typical pooling layer is called max pooling. This is mathematically shown in Eqn. (31).

$$MaxPool(e_\circ, m_\circ) = \begin{cases} e_\circ = floor\left(\frac{(e_i+2p-o)}{c} + 1\right) \\ m_\circ = floor\left(\frac{(m_i+2p-o)}{c} + 1\right) \end{cases} \tag{31}$$

Where $Floor(S)$ was the function of bringing together. A feature map's output height and width are denoted by the variables $e_\circ$ and $m_\circ$, respectively. The feature map parameters are as follows: $e_i$is the intake height, m_i is the input breadth, p is the "padding", o is the "kernel size", and c is the kernel stride of max pooling.

The weight of CNN is adjusted using a hybrid optimization model to improve the projected UHSC flexural strength prediction model's prediction accuracy.

The proposed model is an automated sign language recognition tool that utilizes a CNN architecture with various optimizers (Adam, Adadelta, Adagrade, RMS prop, and SGD) to detect sign language. This model aims to improve communication for deaf and mute individuals and bridge the gap created by traditional communication methods by providing a higher-performance recognition tool.

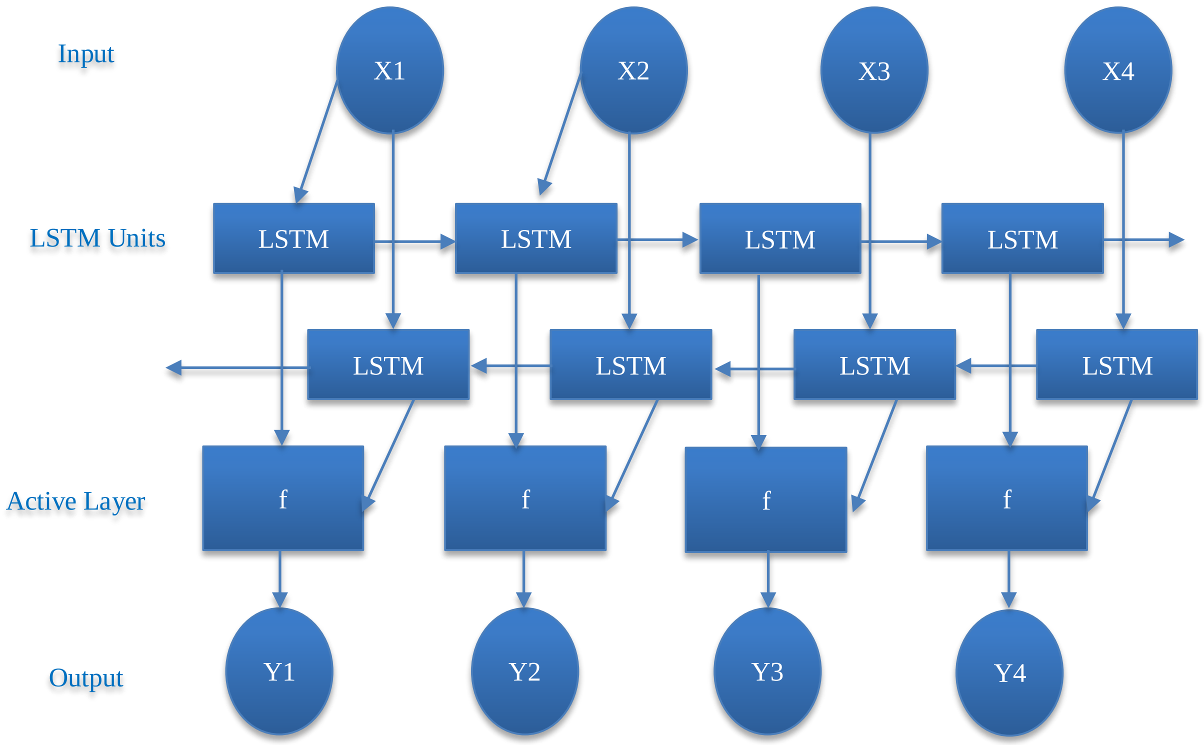

Fig. 3 Flowchart – bi-directional LSTM network

3.6.3. Bidirectional LSTM Algorithm

By combining past and future input data sequences, the bi-LSTM network is produced. Two linked layers are used to process the input data. Bi-directional LSTM predicts or tags the sequence of each element based on the context of items from the past and future using a finite sequence. This Bi-LSTM results from two LSTMs and is intended to train a network with input data sequences from the past and the future. Two linked layers are used to process the input data. The bi-directional LSTM uses a constrained sequence to anticipate or tag the sequence of each element in the past and future.

Two LSTMs resulted in this.

$$x_g^s = \sum_{k=1}^{k} a_1^s \, b_{kg} + \sum_{g's>0}^{G} w_{g'}^{s-1} \, bg'g \qquad (32)$$

$$x_g^s = \theta_g \left(x_m^s \right) \qquad (33)$$

Two distinct LSTM networks make up the bidirectional LSTM, one of which processes the sequence from beginning to end and the other from end to end. As an input to the following layer or for prediction, the outputs from these two LSTMs are then combined. In several NLP applications, including sentiment analysis, machine translation, and text classification, bidirectional LSTMs have been effectively used. These tasks can benefit from better performance by capturing contextual information from both sides, as shown in Figure 3 Flowchart – Bi-Directional LSTM Network.

3.6.4. RNN

The DCNN and Attention-based Bi-LSTM are trained using the selected optimal features F_{i*}. The outcome acquired from DCNN is Out^{DCNN}, Attention-based Bi-LSTM $Out^{Bi-LSTM}$. They are fed as input to optimized RNN, whose bias function is fine-tuned using the Hybrid optimization model. The term "recurrent neural network (RNNa type of neural network that actually works well with sequential data. Unlike feedforward neural networks that process data in a single pass, RNNs have loops that allow information to persist from one time step to the next. This makes RNNs particularly useful for processing sequential data where the current output depends on not only the current input but also previous inputs. In the context of botnet attack detection, RNNs can be used to analyze network traffic patterns over time and detect anomalous behavior that may indicate the presence of a botnet. For example, an RNN could be trained on a time series of network traffic data and learn to identify patterns of traffic that are characteristic of botnet activity. Once trained, the RNN could be used to monitor network traffic in real time and raise an alert when it detects traffic patterns that resemble those seen in the training data. The bias of the RNN is tuned using the new hybrid optimization method to improve the model's detection accuracy.

3.7. Botnet Attack Mitigation via Botnet Traffic Filter

A botnet is an infected network of devices that can be remotely controlled to perform malicious tasks like distributed denial of service attacks, malware distribution, and data theft. Botnets pose a serious threat to network security and user privacy. A botnet traffic filter is a software tool designed to detect and block network traffic associated with botnets. The botnet traffic filter analyses incoming network traffic and looks for patterns and behaviors associated with botnet activity. It can be implemented at various points in a network, such as a firewall, network switch, or individual devices. When the filter detects traffic associated with a botnet, it can take actions such as blocking the traffic or alerting network administrators. Botnet traffic filters are essential in protecting networks and devices from the damage caused by botnets. Without a botnet traffic filter, botnets can easily compromise networks, steal sensitive data, and launch devastating attacks. Organizations can significantly reduce their risk of being targeted by botnets by implementing a botnet traffic filter and ensuring the network's and users' security and privacy.

3.8. Performance Metrics for Classification

We compared the classification performance of the "SMOTE-DRNN" model to that of the "DRNN" model and other state-of-the-art “ML/DL” models using “training loss, validation loss, accuracy, precision, recall, F1 score, FPR, NPV, area under the receiver operating characteristic curve (AUC), Geometric Mean (GM), Matthew’s Correlation Coefficient (MCC), training time, and testing time”. The performance of the “DRNN and SMOTE-DRNN” models is one of the goals of this article. The goal is to examine the performance impact of highly asymmetric network traffic data.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} + 100\% \tag{34}$$

$$precision = \frac{TP}{TP + FP} + 100\% \tag{35}$$

$$recall = \frac{TP}{TP+FN} + 100\% \tag{36}$$

$$F1 = \frac{2 \times TP}{(2 \times TP) + FP + FN} \times 100\% \tag{37}$$

$$FPR = \frac{FP}{FP+TN} \times 100\% \tag{38}$$

$$NPV = \frac{TN}{TN+FN} \times 100\% \tag{39}$$

$$AUC = \frac{1}{2}\left[\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right] \times 100\% \tag{40}$$

$$GM = \sqrt{\frac{TP}{TP+FN} + \frac{TN}{TN+FP}} \times 100\% \tag{41}$$

$$MCC = \frac{(TP\times TN)-(FP\times FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{42}$$

The ratio of correctly identified IoT botnet attacks is represented by TP. FP indicates the proportion of normal samples misrepresented as botnet attacks. TN is the rate of correctly identified normal traffic. FN is the percentage of misclassified IoT botnet attacks as normal traffic. The four values are used in a confusion matrix to evaluate machine learning model performance.

4. Experimental Result

4.1. Data about Network Traffic

The distribution of samples in the “training, validation, and test sets” across the 11 classes is shown in Table 1. Data from network traffic indicated a significant class imbalance. This frequently degrades ML and DL model classification performance in minority classes. Figure 4 shows samples of data from the test, training, and validation sets.

Table 1. Samples of data from the training, validation, and test sets

Class	Training	Validation	Testing
DDH	588	197	204
DDT	586,393	195,713	195,274
DDU	568,760	189,407	190,088
DT	369.965	122,861	122,974
DH	906	311	268
KL	48	14	11
DU	632,641	205,662	206,789
Norm	280	87	100
OSF	10.677	3438	3482
SS	42,865	13.659	14,413
DE	4	1	1

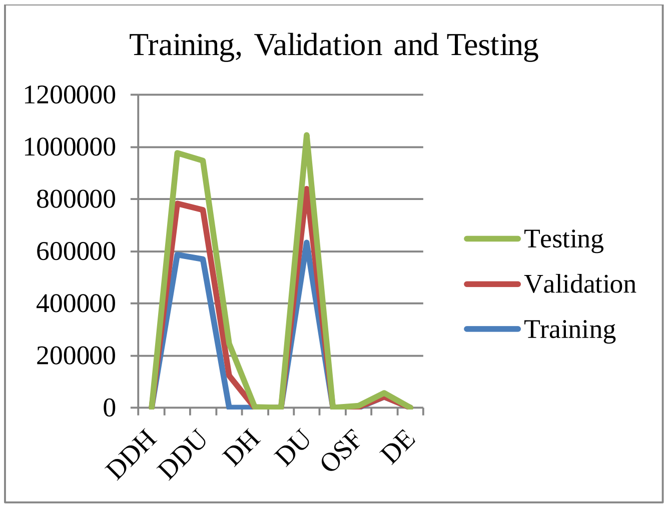

Fig. 4 Samples of data from the test, training, and validation sets

4.2. Model Underfitting and Overfitting Resistance

To assess the DRNN and SMOTEDRNN models' resistance to “underfitting and overfitting”, respectively, they investigate the “ross-entropy losses during training and validation”. The samples of data from the test, training, and validation sets are shown in Table 2.

Table 2. Samples of data from the test, training, and validation sets

	Training loss	
	DRNN	SMOTE
Epoch	0.08	0.04
	0.07	2
	0.06	3
	0.04	5
	0.02	7
	0.01	8

Losses in cross-entropy of the "DRNN and SMOTE-DRNN models" are shown in Figure 2 during training. "As the number of epochs increased from one to ten, "cross-entropy losses" decreased.

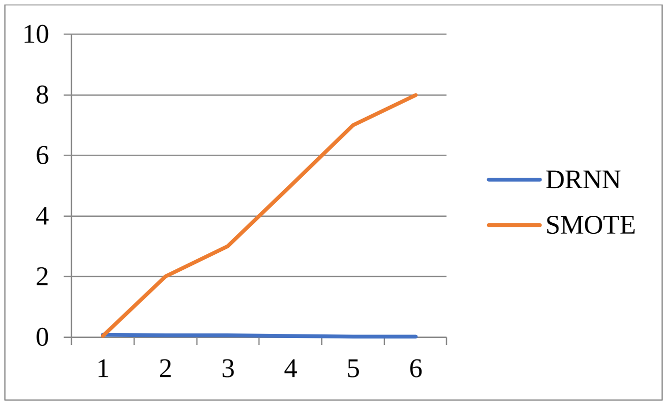

Fig. 5 Losses in training in the DRNN and SMOTE-DRNN models

The DRNN and SMOTE-DRNN model's cross-entropy losses during validation are shown in Figures 5 and 6. In general, cross-entropy losses decreased from 1 to 10 epochs. Table 3 shows the losses in validation for the DRNN and SMOTE-DRNN models.

Table 3. Losses in validation for the DRNN and SMOTE-DRNN models

Validation loss			
Epoch	DRNN	0.1	0.02
		0.6	0.04
		0.4	0.04
		0.2	0.06
	SMOTE-DRNN	1	0.005
		2	0.002
		5	0.06
		7	0.07
		8	0.08

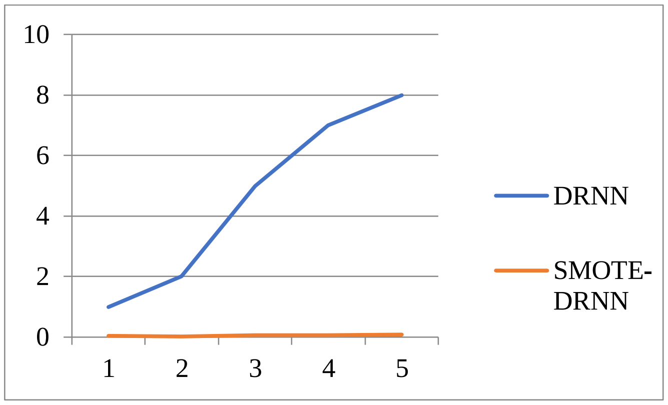

Fig. 6 Losses in validation in the DRNN &SMOTE-DRNN models.

Table 4. Classes' worth of ML and DL model recall (all metrics are in percent)

Model	DDH	DDT	DDU	DU	DT	OSF	DH	SS	KL	DE	Rank
RNN	100.00	100.00	100.00	100.00	100.00	93.23	100.00	88.92	77.91	98.76	2
SVM	62.24	54.86	76.06	100.00	68.56	69.86	72.75	71.65	87.09	88.98	13
DNN	82.86	84.42	67.92	81.96	59.35	81.76	74.86	68.86	78.04	86.87	15
RNN	95.62	95.62	95.62	64.86	96.12	67.76	96.45	96.75	98.42	66.54	12
CNN	97.01	97.01	97.01	98.54	97.34	97.76	98.69	95.65	98.04	100.00	16
DRNN	77.45	99.64	100.00	95.98	98.43	98.57	88.89	99.78	70.67	100.00	2
SMOTE-DRNN	99.02	99.02	100.00	100.00	99.65	100.00	99.68	99.76	100.00	100.00	4

Table 5. Overall classification performance of the ML and DL models (all metrics are in %)

Model	Acc	Prec	Recall	FI score	FPR	NPV	AUG	GM	MCC	T train(S)	T-test(S)
RNN	99.8	-	-	-	1.27	-	-	-	-	200.60	42.34
SVM	-	-	-	-	2.98	-	-	-	-	-	-
DNN	97.21	-	-	-	4.75	-	-	-	-	1400.60	-
RNN	97.53	-	-	-	1.47	-	-	-	-	1397.79	-
CNN	94.73	-	-	-	1.65	-	-	-	-	2800.28	-
SMOTE-DRNN	100.00	99.87	99.76	99.64	0.00	100	99.65	99.63	99.49	1138.73	9.81

The “SMOTE-DRNN” model's overall classification “performance, training time, and testing time” are shown in Table 5, along with those of the most advanced “ML/DL models”. Comparing the “SMOTE-DRNN” model to the most advanced “ML/DL models”, the “SMOTE-DRNN model” had a higher overall accuracy and a lower FPR. The most recent “Machine Learning and Deep Learning models” overall “precision, recall, F1 score, NPV, AUC, GM, and MCC” were not documented in the literature. Except for RNN, DNN, RDTIDS, and BLSTM, the SMOTE-DRNN model trained more quickly than other cutting-edge ML/DL models using the training set that contains network traffic samples.

The majority of the testing times for cutting-edge “Machine Learning and Deep Learning models” have not been documented in the literature. The test set's network traffic samples might be classified in less time than the RNN and BLSTM models. Show the given below in Figures 7 and 8.

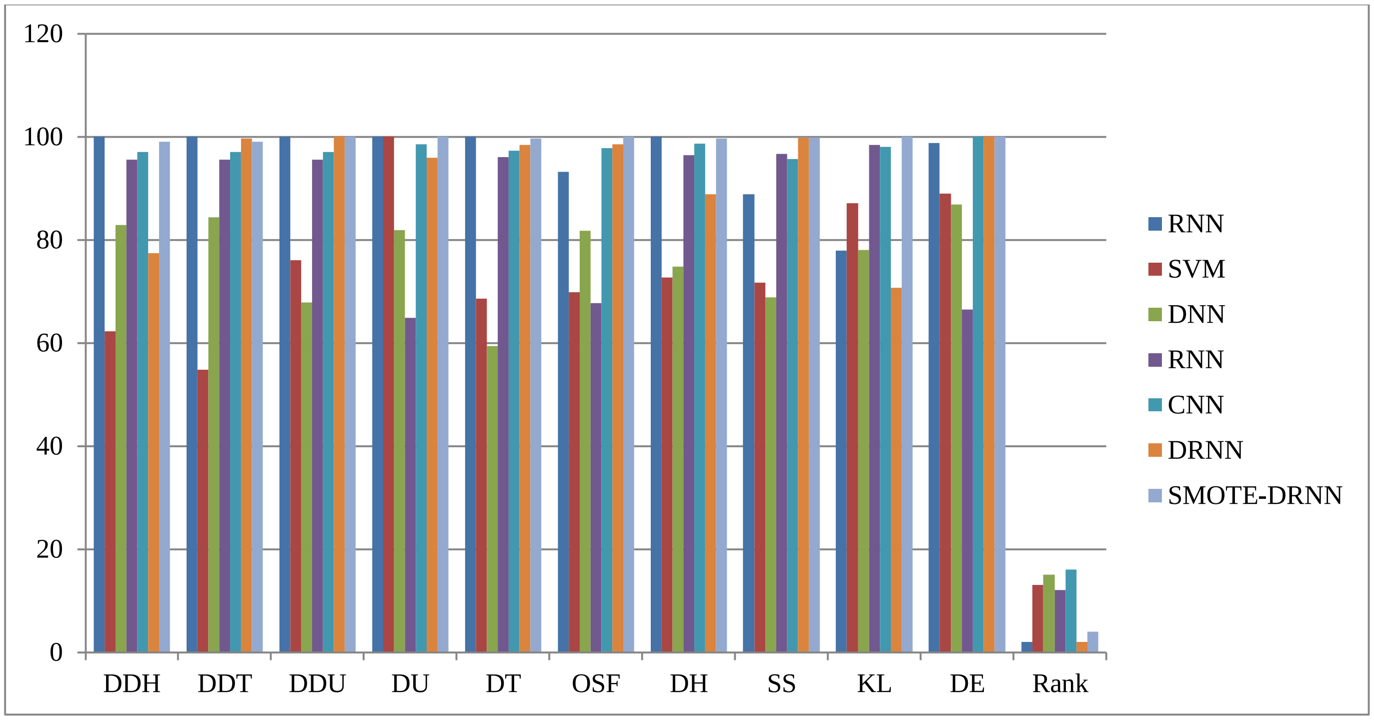

Fig. 7 ML and DL are recall

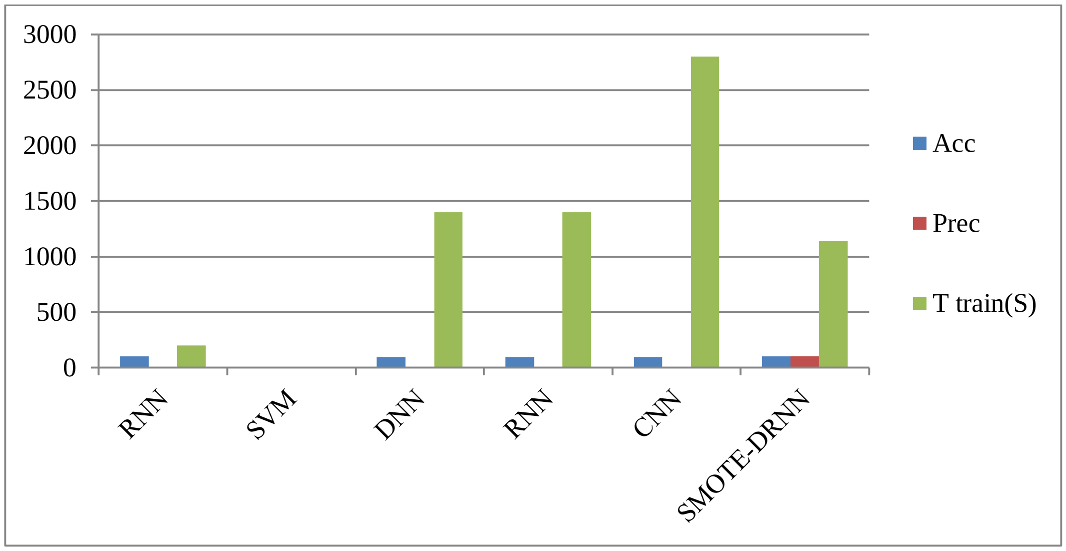

Fig. 8 Overall effectiveness of the ML &DL models for classification

5. Conclusion

A novel ensembled-deep-learning model for detecting IoT-BOTNET attacks is presented in this paper. The proposed model included six major phases: "Pre-processing, Data Augmentation, Feature Extraction, Feature Selection, and Botnet attack mitigation". Initially, the collected raw data was pre-processed via Data Cleaning and Data Normalization (min-max normalization). Then, the pre-processed data was augmented using the "Synthetic Minority Oversampling Technique (SMOTE)" approach to focus on the problem of class imbalance. Then, features like Central tendency (Generalized mean, Winsorized mean, Median, standard deviation, variance), Measure of Dispersion (Skewness, Variance, IQR), and "Information Gain" were extracted from the augmented data. Using the extracted features, the best features were selected by the Hybrid optimization model-Clan Updated Grasshopper Optimization Algorithm (CUGOA). (The proposed CUGOA model was the combination of the Elephant Herding Optimization (EHO) and Grasshopper Optimization Algorithm (GOA), respectively. Then, the Botnet Attack detection was accomplished using the new ensembled-deep-learning model that enclosed the DCNN, Attention-based Bi-LSTM, and optimized RNN. The DCNN and Attention-based Bi-LSTM were tuned using the selected optimal features. The outcome from DCNN and Attention-based Bi-LSTM was fed as input to the optimized RNN model. The final detected outcome regarding the presence/ absence of a botnet attack was acquired from the optimized RNN model, whose bias function was fine-tuned using the new Hybrid optimization model. Once the attacker was found to be present in the network, it was mitigated using the new Botnet Traffic Filter (BTF). Thus, the network became highly reliable. The proposed model was evaluated over the existing models regarding "accuracy, sensitivity, specificity, and precision" as well.

References

[1] Andrea Zanella et al., "Internet of Things for Smart Cities," *IEEE Internet of Things Journal*, vol. 1, no, 1, pp. 23-32, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[2] Cisco Annual Internet Report (2018–2023) White Paper, Cisco, pp. 1-35, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[3] Gernot Vormayr, Tanja Zseby, and Joachim Fabini, "Botnet Communication Patterns," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2768-2796, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[4] Manos Antonakakis et al., "Understanding the Mirai Botnet," *Proceedings of the 26th USENIX Security Symposium*, Vancouver, BC, Canada, pp. 1093-1110, 2017. [Google Scholar] [Publisher Link]

[5] Constantinos Kolias et al., "DDoS in the IoT: Mirai and Other Botnets," *Computer*, vol. 50, no. 7, pp. 80-84, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[6] Robert M. Lee, Michael J. Assante, and Tim Conway, "Analysis of the Cyber Attack on the Ukrainian Power Grid," *Electricity Information Sharing Analysis Center*, pp. 1-23, 2016. [Google Scholar] [Publisher Link]

[7] Brittany D. Davis, Janelle C. Mason, and Mohd Anwar, "Vulnerability Studies and Security Postures of IoT Devices: A Smart Home Case Study," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10102-10110, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[8] Wei Zhou et al., "The Effect of IoT New Features on Security and Privacy: New Threats, Existing Solutions, and Challenges Yet to be Solved," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1606-1616, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[9] Maria Stoyanova et al., "A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches, and Open Issues," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1191-1221, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[10] Ioannis Stellios et al., "A Survey of IoT-Enabled Cyberattacks: Assessing Attack Paths to Critical Infrastructures and Services," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3453-3495, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[11] Tie Qiu et al., "How Can Heterogeneous Internet of Things Build Our Future: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2011-2017, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[12] Dave McMillen, Wei Gao, and Charles DeBeck, "A New Botnet Attack Just Mozied into Town," *Security Intelligence*, 2020. [Google Scholar] [Publisher Link]

[13] Saleh Soltan, Prateek Mittal, and H. Vincent Poor, "BlackIoT: IoT Botnet of High Wattage Devices can Disrupt the Power Grid," *Proceedings of the 27th USENIX Security Symposium*, Baltimore, MD, USA, pp. 15-32, 2021. [Google Scholar] [Publisher Link]

[14] Saleh Soltan, Prateek Mittal, and H. Vincent Poor, "Protecting the Grid against IoT Botnets of High-Wattage Devices," *Arxiv*, pp. 1-15, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[15] Harjinder Singh Lallie et al., "Cyber Security in the Age of Covid-19: A Timeline and Analysis of Cyber-Crime and Cyber-Attacks during the Pandemic," *Computers & Security*, vol. 105, pp. 1-20, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[16] Nickolaos Koroniotis et al., "Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset," *Future Generation Computer Systems*, vol. 100, pp. 779-796, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[17] Meng Zhang et al., “Deep Learning for Short-Term Voltage Stability Assessment of Power Systems,” *IEEE Access*, vol. 9, pp. 29711-29718, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[18] Mahdi Ajdani, and Hamidreza Ghaffary, “Introduced a New Method for Enhancement of Intrusion Detection with Random Forest and PSO Algorithm,” *Security and Privacy*, vol. 4, no. 2, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[19] Mokhtar Mohammadi et al., “A Comprehensive Survey and Taxonomy of the SVM-Based Intrusion Detection Systems,” *Journal of Network and Computer Applications*, vol. 178, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[20] Mangayarkarasi Ramaiah et al., “An Intrusion Detection System Using Optimized Deep Neural Network Architecture,” *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 4, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[21] Karthik Kumar Vaigandla, Radha Krishna Karne, and Allanki Sanyasi Rao, “A Study on IoT Technologies, Standards and Protocols,” *IBMRD's Journal of Management & Research*, vol. 10, no. 2, pp. 7-14, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[22] Karthik Kumar Vaigandla et al., “Communication Technologies and Challenges on 6G Networks for the Internet: Internet of Things (IoT) Based Analysis,” *2022 2nd International Conference on Innovative Practices in Technology and Management*, Gautam Buddha Nagar, India, pp. 27-31, 2022. [CrossRef] [Google Scholar] [Publisher Link]