IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 11, 2022, accepted June 10, 2022, date of publication June 15, 2022, date of current version June 23, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3183213

Hybrid Optimization Enabled Robust CNN-LSTM Technique for Network Intrusion Detection

BHUSHAN DEORE AND SURENDRA BHOSALE
Department of Electrical Engineering, Veermata Jijabai Technological Institute, Mumbai 400019, India
Corresponding author: Bhushan Deore (bsdeore_p18@ee.vjti.ac.in)

ABSTRACT Nowadays, computer networks and the Internet are unprotected from many security threats. Introducing adaptive and flexible security-related techniques is challenging because of the new types of frequently occurring attacks. An intrusion detection system (IDS) is a security device similar to other measures, including firewalls, antivirus software, and access control models devised to strengthen communication and information security. Network intrusion detection system (NIDS) plays a vital function in defending computer networks and systems. However, several issues concerning the sustainability and feasibility of existing techniques are faced with recent networks. These concerns are directly related to the rising levels of necessary human interactions and reducing the level of detection accuracy. Several approaches are designed to detect and manage various security threats in a network. This study uses Chimp Chicken Swarm Optimization-based Deep Long Short-Term Memory (ChCSO-driven Deep LSTM) for the intrusion detection process. A CNN feature extraction process is necessary for effective intrusion detection. Here, the Deep LSTM is applied for detecting network intrusion, and the Deep LSTM is trained using a designed optimization technique to enhance the detection performance.

INDEX TERMS Intrusion detection, deep long short-term memory, chimp optimization algorithm, chicken swarm optimization algorithm, convolutional neural network features.

I. INTRODUCTION

The extensive amount of data generated by intrusion detection networks has more complexities and poses challenges to network security. This research area has received significant attention with the rapid progression of network technologies, such as cloud computing [1], fifth-generation (5G) [2], and Internet of Things (IoT) [3]. Network security has gradually increased with the increasing utilization of computer networks across various applications and fields. Various organizations employ conventional security tools, including anti-spam models, antiviruses, and firewalls, to protect against network attacks. However, traditional security models fail to identify novel and sophisticated attacks [4]. Despite cyber-attacks with large scale and high concealment features, conventional intrusion detection approaches have more restrictions when detecting precision and accuracy rates. Hence, it is more important to devise an effective and precise intrusion detection approach to enhance the detection performance [5]. In today's world, the internet is being widely used by everyone. Various types of cyber-attacks are destructive to information security. Thus, network security has become increasingly important in this research area. Various measures are available to protect network security, such as encryption, intrusion detection, authentication, and firewalls. The intrusion detection model can detect the illegal characteristics of such attacks. The intrusion detection process is a vital task in network safety compared to other protective measures because it can identify attacks from network traffic. Usually, there are four common attacks: denial of service (DoS), remote to local (R2L), user to root (U2R), and probe.

The primary intention of intrusion detection is to categorize network traffic into five main types: Normal, DoS, probe, R2L, and U2R. Maintainers consider various measures to protect the network, and traffic is categorized as an abnormal attack [6]. In general, an intrusion detection model is employed in a network system. However, the intrusion detection model is more important during the increase in network traffic. Intrusion detection techniques are divided into two types: anomaly-based and misuse-based approaches. The Signature attacks are compared in a misuse-based model, which effectively detects unfamiliar attacks. At the same time, anomaly-based detection techniques can detect

The associate editor coordinating the review of this manuscript and approving it for publication was Xiaojie Su.

zero-day or undefined attacks. In addition, the pay-load-driven intrusion detection method affects scalability owing to the increase in network traffic and high-speed networks [7]. Intrusion detection is a process of detecting and marking intrusions in a network, and it has two primary operations [8]. At first, existing network data is analyzed, and the information features of the attack data are recorded. They are matched with data present in a network or host, and this process is termed misuse detection. The second function is the formulation of connections among normal behaviour trajectory features in the sample and network data. The differences in behaviour features are observed as intrusion behaviour. This detection process is called anomaly detection [9]. Network intrusion detection systems (NIDS) have been developed as defence systems for observing network behaviours and detecting intrusive actions in modern times. Moreover, a NIDS is an effective defence tool protecting against sophisticated and threat attacks [10].

Network traffic is detected, and abnormal traffic is blocked in the network intrusion detection model. The attack type is seen, and representative data of the attack category are constantly enhanced. Thus, the system's defence is improved. Incidentally, network anomaly identification is attributed to binary class and multi-classification issues. Currently, machine learning, data mining, and neural networks (NN) are commonly utilized by researchers in network anomaly recognition, and these schemes achieve better performance. Conventional machine learning and data mining approaches are primarily based on data selection and feature extraction processes [9]. The machine learning technique effectively enhances the accuracy of the intrusion detection process and has become a promising research area in wireless network security [5]. These approaches intend to learn appropriate features from a large sample of unlabelled data features employed for a restricted number of labelled data features in the supervised learning process [11]. Unlabelled and labelled data are obtained from various distributions, although they should be related to each other [12]. To date, various researchers have used deep learning methods for intrusion detection. Standard deep learning techniques, such as stacked auto-encoders (SAEs) [4], restricted Boltzmann machines (RBMs) [5], supervised learning with convolutional neural networks (CNNs) [6], and deep belief networks (DBNs) [13] are utilized for effective intrusion detection. However, the CNN model can decrease the number of parameters by using various shared weights and sparse connectivity schemes. Furthermore, these methods are devised for supervised learning models and require many labelled network data as input [4].

The main contribution of this study is the introduction of a network intrusion detection model using the developed ChCSO-based Deep LSTM. This intrusion detection approach comprises four phases: preprocessing, dimension transformation, feature extraction, and intrusion detection. First, the input data are obtained from a dataset and preprocessed to remove redundant data. Data normalization is also performed in the preprocessing phase, which arranges the input data. Subsequently, mutual information [14], [15] is employed to complete the dimension transformation process, where the dimension of the data is changed by choosing various features. Consequently, the CNN feature [16] is extracted from the dimension transformation output. Finally, the intrusion detection process is performed using Deep LSTM [17], classifying the data as intruders or genuine users. Furthermore, Deep LSTM is trained using the devised ChCSO algorithm to improve the detection performance. Accordingly, the ChCSO model is designed by integrating the CSO [18] with ChOA technique [19].

The most important contribution is specified as follows:

- **The developed ChCSO enabled Deep LSTM for intrusion detection**, and an effective intrusion detection model is introduced based on the developed ChCSO-based Deep LSTM. Deep LSTM is applied for detecting intrusion, and the deep learning model is trained using the developed ChCSO technique to improve the detection performance. The devised ChCSO model is developed by incorporating the CSO and ChOA algorithms. Moreover, the Deep LSTM model classifies the data as an intruder or genuine user.

The remainder of this paper is organized as follows: Section 2 discusses the literature review on the conventional intrusion detection model. Section 3 introduces the ChCSO-based Deep LSTM for the intrusion detection process. The devised scheme results and discussion are presented in section 4, and section 5 concludes the paper.

II. MOTIVATION

Network intrusion identification is the most effective security process. However, this process has several problems for the effective and efficient identification of anomalies. Moreover, labelling traffic datasets and the imbalanced class distribution of network traffic is considered a significant challenge. These challenges and problems faced by the existing intrusion detection approach are supposed to stimulate the development of novel intrusion detection approaches.

A. LITERATURE REVIEW

The literature on conventional intrusion detection techniques and their advantages and limitations is explained in this section. Khan *et al.* [4] modelled a two-stage deep learning technique for network intrusion detection to enhance detection accuracy. Here, stacked AE and softmax classifiers were considered for network intrusion detection. Moreover, this approach comprises two decision segments: the initial phase is employed to classify network traffic based on the probability score value and the detection of attack types and normal state. The training time of this approach was reduced, even though the computational complexity did not decrease. Yang *et al.* [5] designed a deep learning scheme to reduce the computational complexity of detecting intrusions. In this approach, a deep belief network (DBN) is included with a multi-Restricted Boltzmann machine (RBM) and

backpropagation (BP) network for the effectual intrusion detection process. Moreover, the backpropagation model was employed to tune the weights of the multi-RBM model; eventually, a support vector machine (SVM) was utilized for the training process. This approach significantly increases the precision rate but does not decrease the training time. Wu *et al.* [6] introduced a CNN to identify network intrusion to enhance performance. First, the input data were preprocessed, and the input data format was changed to an image format. Subsequently, a CNN model is applied to perform the training process. This approach effectively solves balanced database issues but does not improve detection accuracy. The computational complexity and storage space were reduced in this model; however, power consumption was not reduced. To decrease the power consumption, Wang *et al.* [9] designed an improved DBN for intrusion detection. In this approach, a kernel-based Extreme Learning Machine (ELM) was applied for the training process. In addition, the enhanced grey wolf optimizer (EGWO) method was employed to optimize the kernel parameters. The processing time of this technique decreased even though dimension reduction was not performed.

Shone *et al.* [20] presented a deep learning model for intrusion detection. A nonsymmetric deep auto-encoder (NDAE) was devised for the unsupervised feature learning process. The processing time of this approach was highly reduced, although this technique was not evaluated in real-world backbone network traffic to improve the performance. Toldinas *et al.* [21] devised a multistage deep learning image-recognition model for intrusion identification to reduce the training duration. Here, the network features were transformed into four-channel images. Furthermore, the converted images were used for the detection process. The computational complexity of this approach was significantly decreased, although the class imbalance issues were not solved. Gustavo De Carvalho Bertoli *et al.* [22] presented an end-to-end model for network intrusion recognition to solve the class imbalance issues. A machine learning technique was designed for the training process. To decrease the dimension of the data, Andresini *et al.* [23] introduced an AE-based deep learning scheme to identify network intrusion. The two AE's were trained during the training. Besides, the triplet network is prepared for the learning process of the feature vector representation. The detection performance was significantly improved; Though, this model was not implemented in artificial intelligence to simplify the process.

B. CHALLENGES

The challenges experienced by present intrusion detection approaches are explained below,

- The major challenge of the intrusion detection process is to achieve high attack detection and a lower false alarm rate by analyzing and observing the events in a network or computer system for detecting probable occurrences.

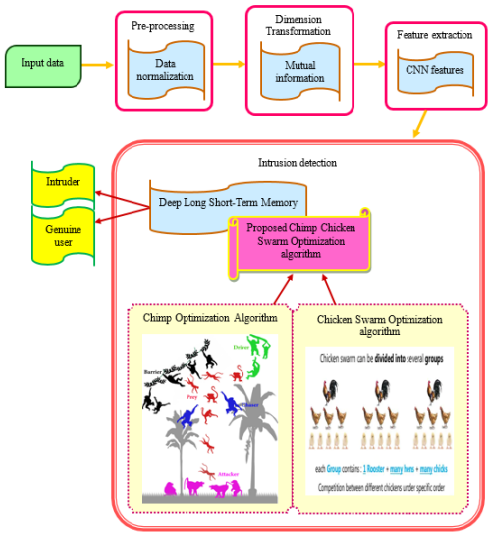

FIGURE 1. Block diagram of an intrusion detection system using proposed ChCSO-based deep LSTM.

- A two-stage deep learning method was devised in [4] for the network intrusion detection process to enhance detection performance and reduce detection time. This method does not combine other techniques, namely multi-task and reinforcement-learning, to improve detection performance.
- A deep learning technique was developed [5], but the detection accuracy did not improve with a smaller sample size to enhance the detection performance.
- The CNN method was introduced in [6] for the intrusion detection process to control zero-day attacks, even though this technique did not enhance the detection accuracy or decrease the detection period.
- The deep learning approach was developed in [20] for the intrusion detection process; however, this model failed to manage zero-day attacks. This approach can be further extended by including real-world network traffic.

III. DEVELOPED CHIMP CHICKEN SWARM OPTIMIZATION-BASED DEEP LSTM FOR NETWORK INTRUSION DETECTION

Network security has received considerable attention owing to the increasing safety concerns. This section describes the developed network intrusion detection process using a ChCSO-based Deep LSTM model. The input data are acquired from a dataset and preprocessed, where data normalization is performed. Subsequently, a dimension transformation is performed based on mutual information [14], [15]. Later, the CNN feature [16] is extracted for further detection in the feature extraction process. Finally, intrusion detection is performed using Deep LSTM [17], and it is trained using a designed ChCSO algorithm to obtain better detection performance. A block diagram of the ChCSO-based Deep LSTM for intrusion detection is shown in Figure 1.

A. INPUT DATA

Let us consider the dataset H along with q amount of intrusion data, which is specified by,

$$H = \{K_1, K_2, \ldots, K_e, \ldots, K_q\} \tag{1}$$

where K denotes the total amount of intrusion data, H symbolizes the dataset, and K_e is the data located at e^{th} index. Intrusion data K_e of $m \times n$ dimension is subjected to a further pre-processing process.

B. PRE-PROCESSING

The input data K_e is used for a preprocessing process in which data normalization is performed for intrusion detection. Normalization is an effective process that arranges data in a database. This process generates tables and produces relationships among the created tables, depending on the rules. Moreover, the data normalization process effectively reduced the redundancy present in the input data. The data-normalized output is denoted as T_e with the size of $m \times n$

C. DIMENSION TRANSFORMATION

Once data normalization is performed, dimension transformation is performed using mutual information [14], [15].Here, the preprocessed data T_e is used to perform the data transformation process to extract CNN features.The normalized data from the preprocessed output are in a vector format. However, the input should be in image format for extracting CNN features because the vector format of the normalized data is converted to a matrix format. If the number of features is n, then $1 \times n$ is converted into an $k \times k$ matrix for the feature extraction process. The k number of the images is chosen using the mutual information model. The processing time of the mutual information model is shorter; Thus, it is used for the devised network intrusion detection model. This model is referred to as a better indicator of relevance, and it estimates the relationship between the class labels and features that are simultaneously sampled. Mutual information helps estimate the information that one variable poses with regard to another. In addition, information theory states that mutual information between two variables is zero if and only if two variables are statistically independent. The mutual information $M(W, X)$ among variables indicating the database W and class labels X is illustrated as,

$$M(W; X) = \sum_{x \in X} \sum_{w \in W} v(w, x) \log \frac{v(w, x)}{v(w)v(x)} \tag{2}$$

where $v(w, x)$ specifies the joint probability distribution function of W and X, $v(w)$ and $v(x)$ are marginal distributions of W and X. The dimension transformation output is represented as R_e, and is used for further feature extraction.

D. FEATURE EXTRACTION

The dimension transformation output R_e is used as the input for extracting CNN features.The CNN model [16] includes five layers: input, convolutional, pooling, Fully Connected (FC), and output layers. The Extraction of CNN feature is shown in Figure 2.

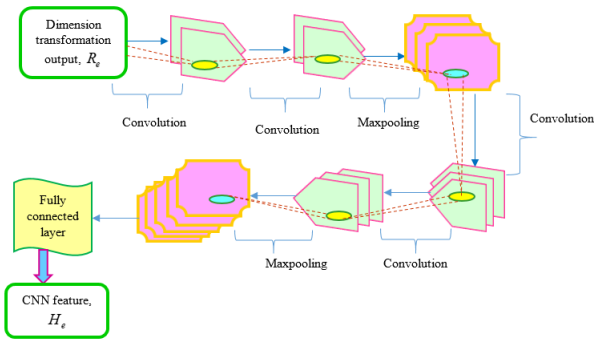

FIGURE 2. Extraction of CNN feature.

The major task of the convolution layer is to extract features from the dimension transformation output. Moreover, it comprises several layers of convolutional kernels, and all the layers correspond to the weight and deviation coefficients. The weight coefficient is considered as c_l,the input of the convolutional layer l is N_{l-1}, and the deviation quality is S_l, while convolution kernel l is in process. The convolution procedure is given by,

$$N_{l-1} = g\left(c_l \otimes N_{l-1} + s_l\right) \tag{3}$$

where N_l denotes the output of the convolution kernel l, $\otimes$, symbolizes the convolution operation, and $g(n)$ specifies the activation function.

The convolution kernel frequently bends the input data to extract the characteristic data. In addition, the ReLU is employed as the activation function of the convolutional layer. The ReLU activation function is more straight forward than the sigmoid, tanh, and activation functions. The ReLU layer increased the speed of the training process and efficiently prevented gradient disappearance. The ReLU is represented by,

$$\mathrm{Re}LU(N_l) = \begin{cases} N_l; & N_l > 0 \\ 0; & N_l \leq 0 \end{cases} \tag{4}$$

N_l Value represents the output of the convolution kernel l.In addition, ReLU (rectified linear unit) is one of the most popular functions used as hidden layer activation function in deep neural networks. The major operation of the pooling layer is to understand the invariances and decrease the difficulty of the CNN by eradicating redundant information using a downsampling process. The pooling process is completed in two ways: maximum and average pooling. The maximum value is selected as the pooling outcome in max pooling, whereas the average value is considered as the pooling outcome in the average pooling process. The max-pooling process is expressed as,

$$B_w = \max\left(O_w^0, O_w^1, O_w^2, \ldots, O_w^f\right) \tag{5}$$

where B_w indicates the output of the pooling area w, max refers to the max-pooling function, and O_w^f specifies an element f of the pooling area w. Additionally, FC layers are used as classifiers. Their major purpose is to weigh the features of the convolutional and pooling layers mapped to the hidden layer, and re-map to the sample indication space. The equivalent dropout function is arranged to randomly discard the neurons in the FC layer to avoid overfitting issues.

The extracted feature is in matrix format; Then, the n number of features and extracted CNN features are converted to vector format for further intrusion detection. The output of the feature extraction process is denoted as V_e, and it is subjected to Deep LSTM for network intrusion detection.

E. DEVELOPED INTRUSION DETECTION MODEL USING CHIMP CHICKEN SWARM OPTIMIZATION-BASED DEEP LSTM

The output of the feature extraction process, V_e, is taken as the input for Deep LSTM to perform the network intrusion detection process. Moreover, the Deep LSTM [17] is trained by the devised ChCSO approach, which is newly designed by combining the CSO algorithm [18] and the ChOA algorithm [19].

1) DEEP LSTM STRUCTURE

The Deep LSTM model efficiently enhances the training performance with less processing time and is therefore employed for network intrusion detection. The composite structure of the inner relative state cells and memory cells provides an effective detection performance. Generally, the detection outcome is mainly dependent on cell states; Therefore, the working function of Deep LSTM is mainly based on the memory cell. The output node G_d receives the input P from the input layer of the network and the previous hidden state Q_{e-1}. The extracted features V_e and Q_{e-1} are given to the tanh function,

$$G_d = \tanh(P.B_{GP} + Q_{e-1}B_{GQ} + I_{in}) \tag{6}$$

where B_{GP} indicates the weight matrix between the input layer and an input node of the memory cell, Q_{e-1} is the input of the hidden state at period stamp $(e-1)$, B_{GQ} implies the weight matrix among hidden states at numerous periods, and I_{in} specifies bias to the input node. The input gate ϖ_e is equivalent to the input node; Thus, it receives a similar input as an identical input node. The input gate utilizes a sigmoidal activation function, called the input gate, as it evades the input flow passing from other nodes to the current node. The input gate is expressed as,

$$\varpi_e = \gamma\left(PB_{GP} + Q_{e-1}B_{GQ} + I_{ig}\right) \tag{7}$$

where ϖ_e indicates the input gate at period e, γ symbolizes the sigmoidal activation function, and I_{ig} represents the bias to the input gate. Furthermore, the internal state Z is a node that includes the self-loop recurrent edge of the activation function along with the unit weight, which is given by,

$$Z = \varpi_e \Theta G_d + Z_{e-1} \tag{8}$$

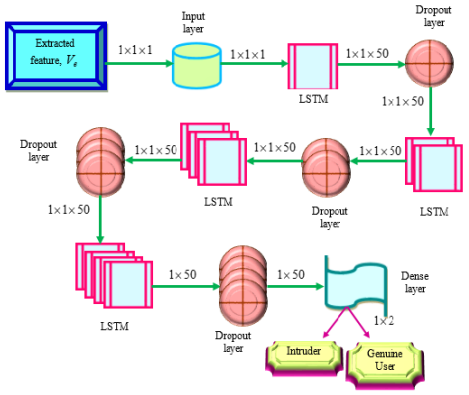

FIGURE 3. Structural diagram of deep LSTM.

where Z_e indicates the internal state at duration e and Z_{e-1} is an internal state at period stamp $e-1$. The forget gate A is employed to reinitiate the interior state of the memory cell, which is specified by,

$$A_e = \gamma\left(P.B_{AP} + Q_{e-1}B_{PQ} + I_{frg}\right) \tag{9}$$

where A_e refers to the forget state at a time e, Θ symbolizes the pointwise linear operator, B_{AP} denotes the weight matrix between the forget gate and input layer, B_{PQ} indicates the weight matrix between the forget gate and hidden state, and I_{frg} refers to the bias of the forget gate. In addition, the output gate τ_e is illustrated as,

$$\tau_e = \gamma\left(P.B_{\tau P} + Q_{e-1}B_{\tau Q} + I_{og}\right) \tag{10}$$

where $B_{\tau P}$ symbolizes the weight matrix between the output gate and input layers, $B_{\tau Q}$ specifies the weight matrix between the hidden states and output gate, and I_{og} refers to the bias of the output gate. The outcome obtained from the memory cell is expressed as,

$$Q_e = \tanh\left(Z_e\right)\Theta\tau_e \tag{11}$$

where $Z_e = G_d\Theta\varpi_d + Z_{e-1}\Theta A_e$ and the output of the deep LSTM are represented as S_e, where the data are detected as genuine users or intruders. The output of the Deep LSTM is specified as J_e, and Figure 3 shows the architecture of the Deep LSTM.

2) DEVELOPED CHIMP CHICKEN SWARM OPTIMIZATION APPROACH FOR TRAINING PROCEDURE OF DEEP LSTM

This section describes the training of the Deep LSTM model based on the devised ChCSO algorithm. The ChCSO approach is introduced here by incorporating CSO [18] and ChOA [19]. The ChOA is usually designed based on the hunting behaviour of chimps. An attacker carries out prey hunting, and residual chimps in a group, including barriers, drivers, and chasers, contribute to the hunting procedure. The convergence speed is very high, which is the main benefit of the ChOA; However, its computational complexity is increased. On the other hand, the CSO approach is a bio-inspired optimization method, which imitates the hierarchy of chicken

swarms and food searching behaviour. Every chicken has a potential solution for optimization problems. The hierarchical order is the most significant in the social lives of chickens, and chicken swarms are classified as one rooster, several hens, and chicks. The chicken identifies the rooster, chick, and hen, depending on the fitness measure of the chicken for all groups. Meanwhile, chickens with the best and worst fitness values are considered chicks, roosters, residuals and hens. The CSO model effectively increased the convergence rate and search accuracy. Therefore, the CSO approach is integrated with the ChOA to obtain better convergence with less computational complexity.

The algorithmic process of the devised ChCSO technique is described as follows,

a: INITIALIZATION

Here, initialization of the t solution with the b amount of solution is performed, which is illustrated by,

$$L = \{L_1, L_2, \ldots, L_t, \ldots, L_b\}; 1 \leq t \leq b \tag{12}$$

where b specifies the entire number of solutions and L_t refers to the t^{th} solution. Here, $L \in B_{GP}, B_{GQ}, B_{AP}, B_{PQ}, B_{\tau P}, B_{\tau Q}, I_{in}, I_{ig}, I_{frg}, I_{og}$.

b: FITNESS FUNCTION COMPUTATION

The optimal solution is selected using a fitness measure; a fitness value with a lower value is the optimal solution for network intrusion detection. The fitness value is estimated using the following expression,

$$\rho = \frac{1}{e}\sum_{\xi=1}^{e}(J_e^* - J_e)^2 \tag{13}$$

where e specifies the total number of samples, ρ represents the fitness measure, J_e is the output from Deep LSTM, and J_e^* indicates the target output.

c: DRIVING AND CHASING PREY

Prey hunting is performed during the exploitation and exploration phases. The expression is derived using the distance for prey driving and chasing, which is specified as,

$$N = |kL_{prey}(a) - xL_{chimp}(a)| \tag{14}$$

where a indicates the iteration count, x and k specify the coefficient value, $L_{chimp}(a)$ indicates the position location of the chimp, and $L_{prey}(a)$ denotes the current location of the prey. The prey chase is expressed as follows,

$$L_{chimp}(a+1) = L_{prey}(a) - n.N \tag{15}$$

where n refers to coefficient vectors, and N implies distance. Moreover, the coefficient vector is expressed as,

$$n = 2mi_1 - m \tag{16}$$

where m linearly decreases from 2.5 to 0, and i_1 represents a random integer. The coefficient vector is illustrated by,

$$k = 2i_2 \tag{17}$$

where, i_2 denotes the random number. Substitute equation (14) in (15),

$$L_{chimp}(a+1) = L_{prey}(a) - n.|kL_{prey}(a) - xL_{chimp}(a)| \tag{18}$$

Let us consider $L_{prey}(a) > L_{chimp}(a)$, the above equation becomes,

$$L_{chimp}(a+1) = L_{prey}(a) - n.kL_{prey}(a) + n.xL_{chimp}(a) \tag{19}$$

$$L_{chimp}(a+1) = L_{prey}(a)(1-nk) + n.xL_{chimp}(a) \tag{20}$$

To obtain better convergence, the movement of the chick expression is integrated in the ChOA, thereby,

$$L_i(a+1) = L_i(a) * (1 + Randu(0, \omega^2)) \tag{21}$$

$$L_i(a) = \frac{L_i(a+1)}{1 + Randu(0, \omega^2)} \tag{22}$$

Let us assume that $L_i(a) = L_{chimp}(a)$, Substituting equation (22) into equation (20),

$$L_{chimp}(a+1) = L_{prey}(a)(1-n.k) + n.x\frac{L_{chimp}(a+1)}{1 + Randu(0, \omega^2)} \tag{23}$$

$$L_{prey}(a)(1-n.k) = L_{chimp}(a+1) - \frac{n.x.L_{chimp}(a+1)}{1 + Randu(0, \omega^2)} \tag{24}$$

$$L_{chimp}(a+1)\left(1 - \frac{n.x.L_{chimp}(a+1}{1 + Randu(0, \omega^2)}\right) = {}_{prey}(a)(1-n.k) \tag{25}$$

$$L_{chimp}(a+1)\left(\frac{1 + Randu(0, \omega^2) - n.x}{1 + Randu(0, \omega^2)}\right) = {}_{prey}(a)(1-n.k) \tag{26}$$

$$L_{chimp}(a+1) = \frac{1 + Randu(0, \omega^2)}{1 + Randu(0, \omega^2) - n.x} \times L_{prey}(a)(1-n.k) \tag{27}$$

where $randu(0, \omega^2)$ indicates a Gaussian distribution with a standard deviation ω^2 and a mean of 0. The standard deviation is formulated as follows,

$$\omega^2 = \begin{cases} 1, & \text{if } O_i \leq O_x \\ \exp\left(\frac{O_x - O_i}{|O_i| + \xi}\right), & \text{if otherwise,} \end{cases} \quad x \in [1, Z],\ x \neq i \tag{28}$$

d: ATTACKING PROCESS

The attack process comprised two stages: exploring the position of the prey and surrounding the prey for the attack. An attacker chimp typically performs the attack process. Moreover, other chimps in a group, such as barriers, drivers, and chaser chimps, contribute during an attack. Therefore, the four optimal solutions are specified as,

$$L(a+1) = \frac{L_1 + L_2 + L_3 + L_4}{4} \tag{29}$$

where, L_1, L_2, L_3 and L_4 are the location of chimps. Every best solution is specified as,

$$L_1 = L_{attacker} - n_1(N_{attacker}) \tag{30}$$

where L_1 represents the attacker, n_1 signifies the coefficient vector of the attacker, and $N_{attacker}$ represents the attacker's prey.

$$L_2 = L_{barrier} - n_2(N_{barrier}) \tag{31}$$

where L_2 symbolizes the barrier, n_2 denotes the coefficient vector of the barrier, and $N_{barrier}$ represents the barrier prey.

$$L_3 = L_{chaser} - n_3(N_{chaser}) \tag{32}$$

where L_3 signifies the barrier, n_3 implies the coefficient vector of the barrier, and N_{chaser} denotes the chaser prey.

$$L_4 = L_{driver} - n_3(N_{driver}) \tag{33}$$

where L_4 signifies the driver, n_4 is the coefficient vector of the driver, and N_{driver} symbolizes the driver prey. In addition, the distance of every solution is given by,

$$N_{attacker} = |k_1 L_{attacker} - x_1 L| \tag{34}$$
$$N_{barrier} = |k_2 L_{barrier} - x_2 L| \tag{35}$$
$$N_{chaser} = |k_3 L_{chaser} - x_3 L| \tag{36}$$
$$N_{driver} = |k_4 L_{driver} - x_4 L| \tag{37}$$

where k_1, k_2, k_3 and k_4 as well as x_1, x_2, x_3, and x_4 refer to the coefficient vectors of the attacker, barrier, chaser, and driver, respectively.

e: PREY ATTACKING

The attacker chimp completes the attack, while the prey stops the movement at the terminal stage of an attacker. The value of must be reduced to design the attacking stage scientifically. Moreover, it allows chimps to update their position depending on the position of the driver, chaser, barrier, attacker, and prey.

f: SEARCHING FOR PREY

The exploration stage of the ChOA is performed using the positions of the barrier, attacker, driver, and the chaser. Here, the chimps separate to chase the prey and integrate them to attack the prey.

g: SOCIAL INCENTIVE

Obtaining social meet and appropriate motivation in the last segment cause chimps to release hunting tasks. There is a possibility of a preference between the normal update model or chaotic method for restarting the position of the chimp depending on the chasing and driving process of the prey, which is specified as,

$$L_{chimp}(a+1) = \begin{cases} \dfrac{1 + Randu(0, \omega^2)}{1 + Randu(0, \omega^2) - n.x} \\ \quad \times L_{prey}(a)(1 - n.k), & \text{if } k < 0.5 \\ \text{Chaotic value}, & \text{if } k > 0.5 \end{cases} \tag{38}$$

where, k specifies random integer among [0,1].Here, a chaotic value may contain sequences of the evolving variable that exactly repeat themselves, resulting in regular intervals beginning at any point in that sequence.

h: RE-EVALUATE FITNESS MEASURE

The fitness function achieves the optimal solution, where a lower fitness measure is taken as the best solution, and the fitness value is computed based on Equation (13).

i: TERMINATE

The above process is repeated until the best solution is achieved. The pseudocode of the introduced ChCSO scheme is presented in algorithm. 1

Algorithm 1: Pseudocode of Devised ChCSO Algorithm

```
Input  : Population L
Output: Best solution
Start
Initialize population of chimp and other algorithmic
  parameters
Evaluate the position of each chimp
Arbitrarily diverse the chimps into independent groups
Until termination condition is obtained
Compute the fitness measure based on equation (13)
L_attacker =optimal search agent
L_chaser =second optimum search agent
L_barrier =third best search agent
L_driver =fourth optimal search agent
while a < max Iteration do
    foreach Chimp do
        Discover chimp group
        Apply group model for updating parameters
    foreach search chimp do
        if k < 0.5 then
            if |n| < 1 then
                Update the position based on equation
                  (27)
            else if |n| > 1 then
                Choose arbitrary search agent
        else if k > 0.5 then
            Update the position using equation (38)
    Update algorithmic parameters
    a = a + 1
return optimal solution
```

Thus, the Deep LSTM model effectively classifies the data as intruders or genuine users with a minimal process duration. The training process is improved by using the ChCSO approach.

IV. RESULTS AND DISCUSSION

The results and discussion of the ChCSO-enabled Deep LSTM approach for intrusion detection are presented in this section. The experimental setup, dataset description,

performance metrics, comparative techniques, comparative analysis, and discussion are presented in this section.

A. DATASET EXPLANATION

The introduced intrusion detection technique is executed using two datasets: the NSL-KDD [24] and BoT-IoT [25] databases.

1) NSL-KDD DATABASE

These data are usually employed in data mining tool competitions and third international knowledge discovery. The competition process is used to design a network intrusion detector, which separates ''good'' normal connections, and ''bad'' links, termed intrusion or attacks. Furthermore, these data comprise standard data and are imitative of the military networks.

2) BoT-IoT DATASET

This dataset is produced by the network structure in the cyber range lab for intrusion detection. These data are generated by the integration of normal and botnet traffic. The source files are provided in various formats: csv, pcap, and argus files. These files are divided based on an attack subgroup and a group for labelling. Furthermore, pcap files have more than 72.000.000 records with proportions of 69.3 GB, while the extracted flow traffic in CSV is 16.7 GB in size.

B. PERFORMANCE METRICS

The performance of the ChCSO-based Deep LSTM model is computed based on three metrics: accuracy, specificity, and sensitivity.

1) ACCURACY

This metric is employed to estimate the appropriately detected intrusion, which is computed by,

$$\phi = \frac{TP + TN}{TP + TN + FP + FN} \tag{39}$$

2) SENSITIVITY

Sensitivity is used to detect the precisely detected ratio of positive, and it is given by,

$$\nu = \frac{TP}{TP + FN} \tag{40}$$

3) SPECIFICITY

Specificity is estimated to detect the accurately identified ratio of negatives and is calculated as,

$$\varepsilon = \frac{TN}{TN + FP} \tag{41}$$

where ϕ specifies accuracy, ν implies sensitivity, ε denotes specificity, TP implies true positives, FP refers to false positives, TN symbolizes true negatives, and FN indicates false negatives.

C. COMPARATIVE METHODS

Existing intrusion detection techniques, namely deep-stacked auto-encoder (DSAE) [4], CNN [6],DBN [20], Dolphin atom search optimization-based Deep Recurrent neural network (DASO-based Deep RNN), and adaptive DASO-based Deep RNN are considered for estimating the performance of the devised intrusion detection model.

D. COMPARATIVE ANALYSIS

This section presents a comparative analysis of the devised ChCSO-based Deep LSTM method based on two databases, namely, NSL-KDD and the BoT-IoT database.

1) COMPARATIVE ANALYSIS USING NSL-KDD DATASET

A comparative analysis of the devised ChCSO-driven Deep LSTM with and without attacks by altering the training data is presented in this section.

a: ANALYSIS WITHOUT ATTACK

Figure 4 presents an analysis of the introduced ChCSO-based Deep LSTM using the NSL-KDD dataset without attacks for the performance metrics. The analysis of the developed ChCSO-based Deep LSTM using accuracy by changing the percentage of training data is shown in Figure 4 (i). The accuracy value of the introduced ChCSO-based Deep LSTM is 0.9568, whereas those of existing approaches, such as DBN, CNN, DSAE, DASO-based deep RNN, and adaptive DASO-based Deep RNN, are 0.8479, 0.8245, 0.8094, 0.9180, and 0.9367, respectively, for 80% of the training data. Figure 4 (ii) shows the analysis of the developed ChCSO-based Deep LSTM scheme using the sensitivity by changing the training data. The sensitivity of DBN is 0.9364, CNN is 0.9281, DSAE is 0.89, DASO-based Deep RNN is 0.9788, adaptive DASO-based deep RNN is 0.9851, and the developed ChCSO-based Deep LSTM is 0.9872 for 80% of the training data. The analysis of the designed ChCSO-based Deep LSTM technique based on specificity by varying the percentage of training data is shown in Figure 4 (iii). When the percentage of training data is 80, the specificity attained by the DBN, CNN, DSAE, DASO-based Deep RNN, adaptive DASO-based Deep RNN, and developed ChCSO-based Deep LSTM i 0.7394, 0.8969, 0.9174, 0.9611, 0.9754, and 0.9814, respectively.

b: ANALYSIS WITH ATTACK

A comparative analysis of the devised ChCSO-driven Deep LSTM using NSL-KDD data with an attack in terms of performance metrics is shown in Figure 5. Figure 5 (i) shows an analysis of the developed ChCSO-based Deep LSTM scheme for accuracy by altering the training data value. The accuracy obtained by DBN is 0.7840, CNN is 0.8076, DSAE is 0.8503, DASO-based Deep RNN is 0.8791, adaptive DASO-based Deep RNN is 0.8929, and the developed ChCSO-based Deep LSTM is 0.9210 for 80% of the training data. An analysis of the introduced ChCSO-based Deep LSTM technique based

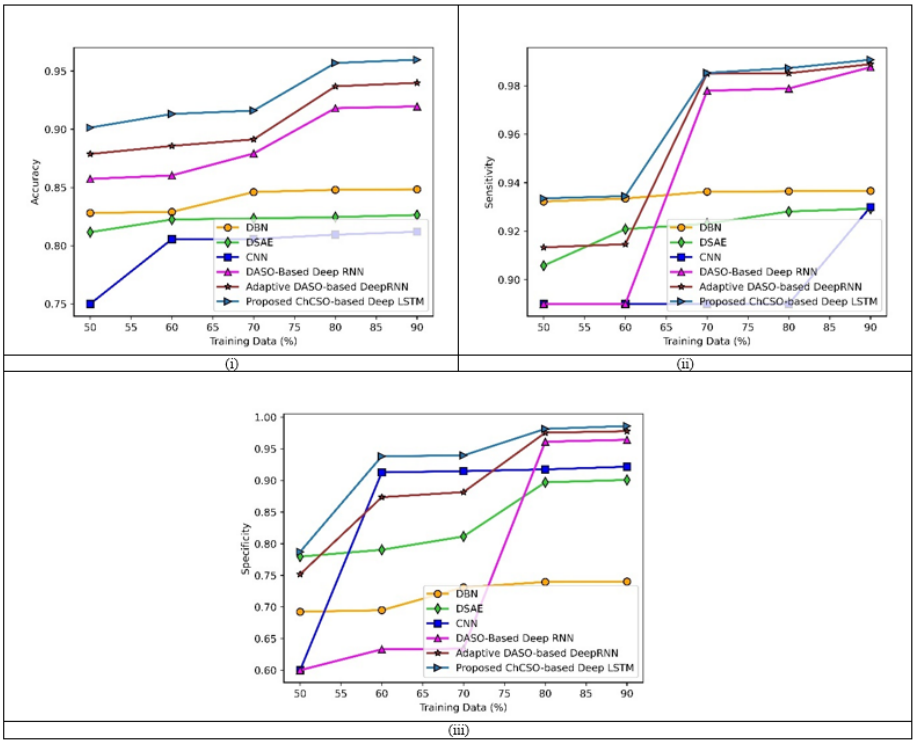

FIGURE 4. Comparative analysis of ChCSO based deep LSTM using NSL-KDD dataset without attacks (i) Accuracy, (ii) Sensitivity, (iii) Specificity.

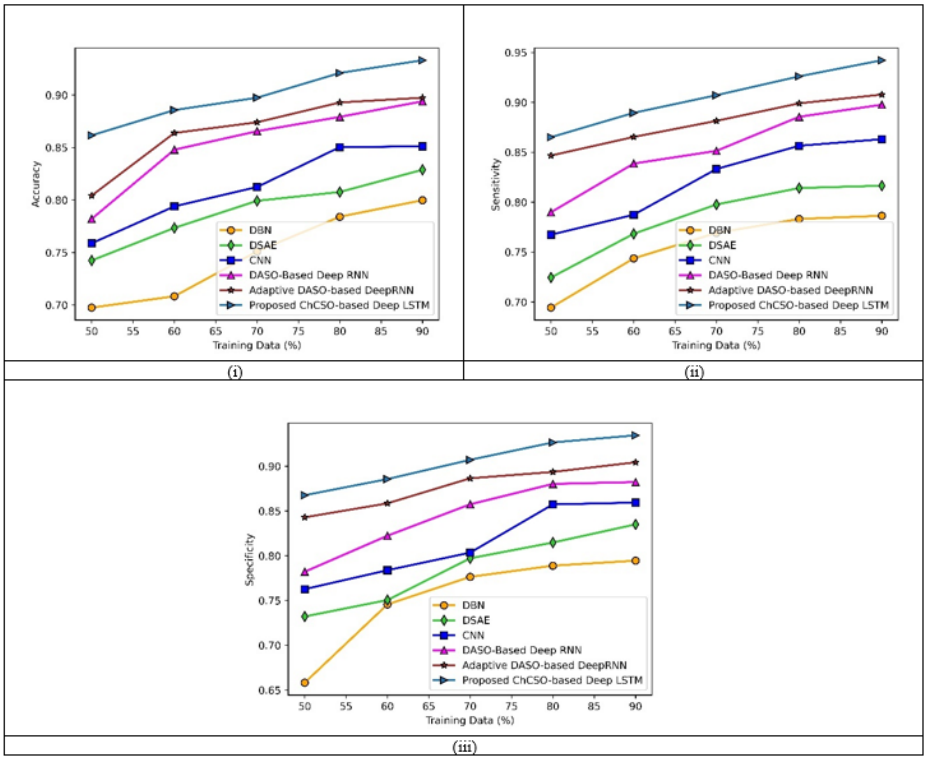

FIGURE 5. Comparative analysis of ChCSO based deep LSTM using NSL-KDD dataset with attacks (i) Accuracy, (ii) Sensitivity, (iii) Specificity.

on sensitivity by varying the percentage of training data is illustrated in Figure 5 (ii). When the percentage of training data is 80, the sensitivities attained by the DBN, CNN, DSAE, DASO-based Deep RNN, adaptive DASO-based deep RNN, and developed ChCSO-based Deep LSTM are 0.7831, 0.8141, 0.8564, 0.8855, 0.8990, and 0.9260, respectively.

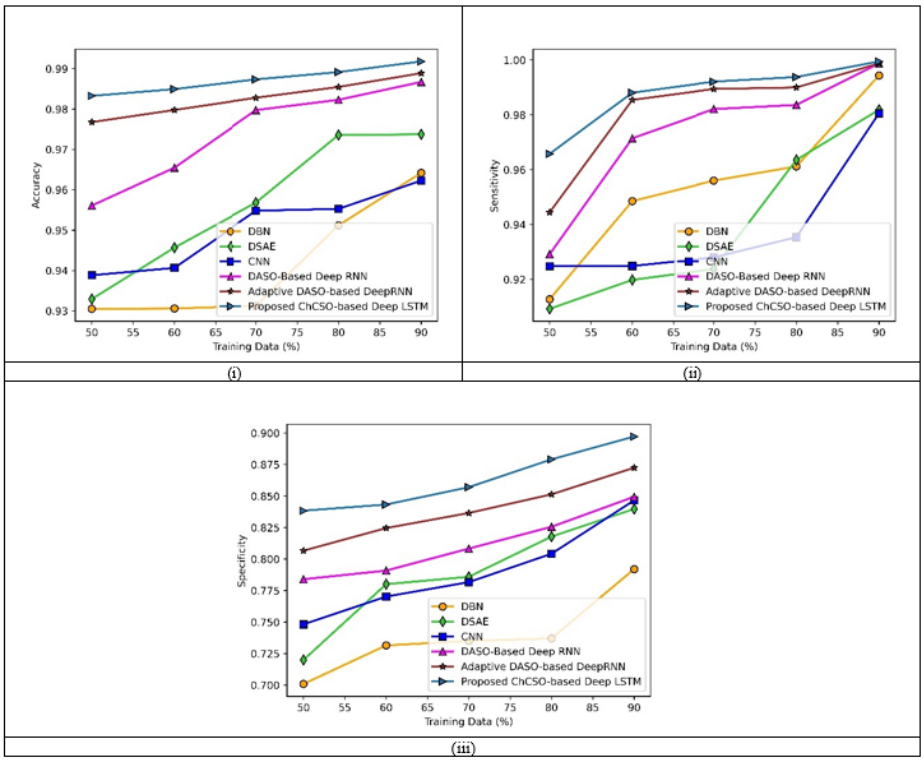

FIGURE 6. Comparative analysis of ChCSO based deep LSTM using BoT-IoT dataset without attacks (i) Accuracy, (ii) Sensitivity, (iii) Specificity.

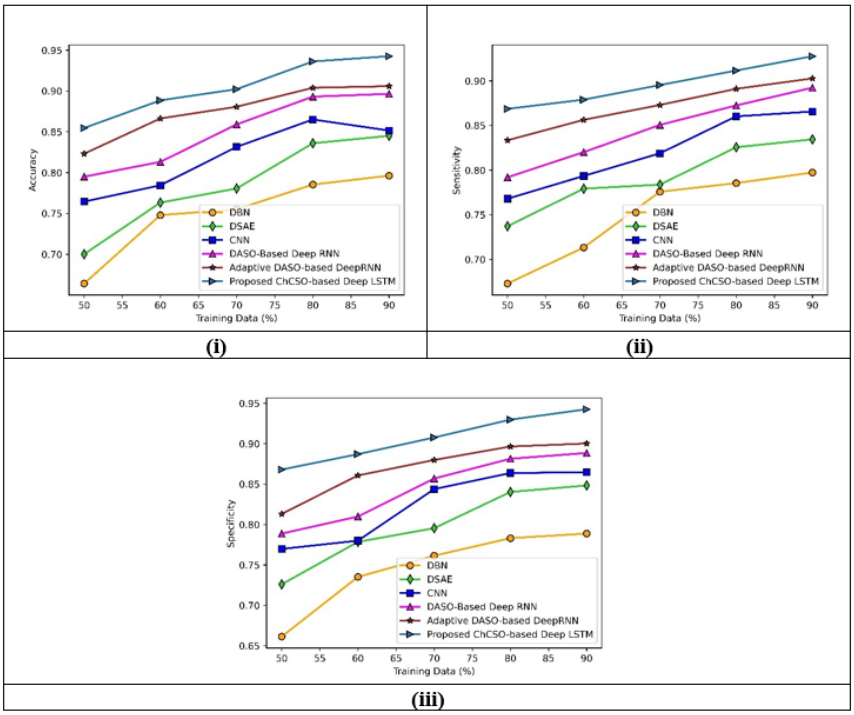

FIGURE 7. Comparative analysis of ChCSO based deep LSTM using BoT-IoT dataset with attacks (i) Accuracy, (ii) Sensitivity, (iii) Specificity.

The analysis of the developed ChCSO-based Deep LSTM using specificity by shifting the percentage of training data is shown in Figure 5 (iii). The specificity of the developed ChCSO-based Deep LSTM is 0.9568, whereas those of the existing approaches, such as DBN, CNN, DSAE, DASO-based deep RNN, and adaptive DASO-based Deep RNN are 0.7889, 0.8147, 0.8573, 0.8802, 0.8935, and 0.9265, respectively, for 80% of the training data.

TABLE 1. Comparative description.

Datasets	Based on	Metrics/ Methods	DBN	DSAE	CNN	DASO-based Deep RNN	Adaptive DASO-based Deep RNN	Developed ChCSO-based Deep LSTM
NSL-KDD dataset	Without attack	Accuracy	0.8483	0.8263	0.8119	0.9195	0.9397	0.9596
		Sensitivity	0.9366	0.9292	0.93	0.9876	0.9888	0.9907
		Specificity	0.7399	0.9008	0.9217	0.9642	0.9777	0.9860
	With attack	Accuracy	0.7998	0.8288	0.8511	0.8940	0.8974	0.9331
		Sensitivity	0.7863	0.8165	0.8630	0.8976	0.9078	0.9421
		Specificity	0.7945	0.8350	0.8593	0.8822	0.9043	0.9344
BoT-IoT dataset	Without attack	Accuracy	0.9641	0.9737	0.9622	0.9867	0.9888	0.9917
		Sensitivity	0.9943	0.9818	0.9805	0.9987	0.9987	0.9994
		Specificity	0.7920	0.8397	0.8465	0.8494	0.8724	0.8972
	With attack	Accuracy	0.7962	0.8450	0.8515	0.8964	0.9058	0.9425
		Sensitivity	0.7973	0.8344	0.8656	0.8923	0.9027	0.9276
		Specificity	0.7888	0.8483	0.8647	0.8884	0.9002	0.9425

2) COMPARATIVE ANALYSIS IN TERMS OF BoT-IoT DATA

A comparative analysis of the introduced ChCSO-based Deep LSTM with and without attacks by altering the training data is presented in this section.

a: ANALYSIS WITHOUT ATTACK

Figure 6 shows a comparative analysis of the introduced ChCSO-based Deep LSTM using BoT-IoT data without attacks for the performance metrics. The analysis of the developed ChCSO-based Deep LSTM using accuracy by changing the percentage of training data is shown in Figure 6 (i). The accuracy value attained by the devised ChCSO-driven Deep LSTM is 0.9891, whereas those of existing approaches, such as DBN, CNN, DSAE, DASO-based Deep RNN, and adaptive DASO-based Deep RNN are 0.9512, 0.9735, 0.9552, 0.9822, and 0.9854, respectively, for 80% of the training data. Figure 6 (ii) shows the analysis of the developed ChCSO-based Deep LSTM scheme using the sensitivity by changing the training data percentage. The sensitivity of DBN is 0.9612, CNN is 0.9635, DSAE is 0.9354, DASO-based Deep RNN is 0.9836, adaptive DASO-based Deep RNN is 0.99, and the developed ChCSO-based Deep LSTM is 0.9938 for 80% of the training data. An analysis of the devised ChCSO-based Deep LSTM technique based on specificity by varying the percentage of training data is illustrated in Figure 6 (iii) When the percentage of training data is 80, the specificities attained by the DBN, CNN, DSAE, DASO-based Deep RNN, adaptive DASO-based deep RNN, and developed ChCSO-based Deep LSTM is 0.7370, 0.8178, 0.8041, 0.8255, 0.8513, and 0.8791, respectively.

b: ANALYSIS WITH ATTACK

The analysis of ChCSO-based Deep LSTM depends on the BoT-IoT data with the attack in terms of performance metrics, as shown in Figure 7. Figure 7 (i) shows an analysis of the developed ChCSO-based Deep LSTM scheme in terms of accuracy by altering the training data percentage. The accuracy obtained by DBN is 0.7853, CNN is 0.8359, DSAE is 0.8650, DASO-based Deep RNN is 0.8930, adaptive DASO-based Deep RNN is 0.9038, and the developed ChCSO-based Deep LSTM is 0.9361 for 80% of the training data. An analysis of the devised ChCSO-based Deep LSTM technique based on sensitivity by varying the percentage of training data is illustrated in Figure 7 (ii). When the percentage of training data is 80, the sensitivities attained by the DBN, CNN, DSAE, DASO-based Deep RNN, adaptive DASO-based deep RNN, and developed ChCSO-based Deep LSTM are 0.7854, 0.8256, 0.8602, 0.8724, 0.8910, and 0.9115, respectively. The analysis of the developed ChCSO-based Deep LSTM using specificity by changing the percentage of training data is shown in Figure 7 (iii). The specificity of the ChCSO-based Deep LSTM is 0.9297, whereas those of the existing approaches, such as DBN, CNN, DSAE, DASO-based deep RNN, and adaptive DASO-based Deep RNN are 0.7830, 0.8402, 0.8636, 0.8813, and 0.8964, respectively, for 80% of the training data.

E. COMPARATIVE DESCRIPTION

A comparative description of the introduced ChCSO-based Deep LSTM approach based on the NSL-KDD and BoT-IoT datasets with respect to performance metrics is presented in Table 1. The accuracy obtained by the developed ChCSO-based Deep LSTM is 0.9596, whereas those of existing approaches, such as DBN, CNN, DSAE, DASO-based Deep RNN, and adaptive DASO-based Deep RNN are 0.8483, 0.8263, 0.8119, 0.9195, and 0.9397, respectively, for 90% of the training data. The accuracy rate is significantly increased because of the development of an effective optimization algorithm. The sensitivity value obtained by DBN is 0.9366, CNN is 0.9292, DSAE is 0.93, DASO-based Deep RNN is 0.9876, adaptive DASO-based deep RNN is 0.9888, and the developed ChCSO-based Deep LSTM is 0.9907, while the training data is 90%. The sensitivity value increased in the developed technique by considering the deep learning model. When the percentage of training data is 90, the specificities of the DBN, CNN, DSAE, DASO-based Deep RNN, adaptive DASO-based Deep RNN, and developed ChCSO-based Deep LSTM are 0.7399, 0.9008, 0.9217, 0.9642, 0.9777, and 0.9860, respectively. In this method, the extraction of the CNN features effectively increased the specificity.

V. CONCLUSION

An effective network intrusion detection method using the developed ChCSO-based Deep LSTM is presented in this paper. In this designed network detection model, input data are acquired from the BoT-IoT and NSL-KDD databases and normalized to order the input data. Mutual information is applied to the dimension transformation process. Feature extraction is essential for effective detection. In this method, the CNN feature is extracted from dimension-transformed data. Finally, intrusion detection is performed using Deep LSTM and is trained using the designed ChCSO approach. The developed ChCSO technique is introduced by integrating ChOA and CSO algorithms. The Deep LSTM model classifies the data as genuine users and intruders. Furthermore, the performance of the introduced intrusion-detection scheme is estimated based on three metrics. Thus, the presented ChCSO algorithm achieved better performance, with an accuracy of 0.9917, specificity of 0.9994, and sensitivity of 0.9860. In addition, the designed network intrusion detection model can be further extended by considering another deep learning approach, along with an effective deep learning technique.

REFERENCES

[1] J. Shen, T. Zhou, D. He, Y. Zhang, X. Sun, and Y. Xiang, "Block design-based key agreement for group data sharing in cloud computing," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 6, pp. 996–1010, Nov. 2019.

[2] S. F. Jilani, Q. H. Abbasi, and A. Alomainy, "Inkjet-printed millimetre-wave PET-based flexible antenna for 5G wireless applications," in *IEEE MTT-S Int. Microw. Symp. Dig.*, Aug. 2018, pp. 1–3.

[3] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 2019.

[4] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, "TSDL: A two-stage deep learning model for efficient network intrusion detection," *IEEE Access*, vol. 7, pp. 30373–30385, 2019.

[5] H. Yang, G. Qin, and L. Ye, "Combined wireless network intrusion detection model based on deep learning," *IEEE Access*, vol. 7, pp. 82624–82632, 2019.

[6] K. Wu, Z. Chen, and W. Li, "A novel intrusion detection model for a massive network using convolutional neural networks," *IEEE Access*, vol. 6, pp. 50850–50859, 2018.

[7] S. Zavrak and M. Iskefiyeli, "Anomaly-based intrusion detection from network flow features using variational autoencoder," *IEEE Access*, vol. 8, pp. 108346–108358, 2020.

[8] M. Kumar and A. K. Singh, "Distributed intrusion detection system using blockchain and cloud computing infrastructure," in *Proc. 4th Int. Conf. Trends Electron. Informat. (ICOEI)*, Jun. 2020, pp. 248–252.

[9] Z. Wang, Y. Zeng, Y. Liu, and D. Li, "Deep belief network integrating improved kernel-based extreme learning machine for network intrusion detection," *IEEE Access*, vol. 9, pp. 16062–16091, 2021.

[10] A. Yousef, G. Kvascev, S. Gajin, and Z. Jovanovic, "Flow-based anomaly intrusion detection system using two neural network stages," *Comput. Sci. Inf. Syst.*, vol. 11, no. 2, pp. 601–622, 2014.

[11] J. S. Anita and J. S. Abinaya, "Impact of supervised classifier on speech emotion recognition," *Multimedia Res.*, vol. 2, no. 1, pp. 9–16, 2019.

[12] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *Expert Syst. Appl.*, vol. 36, no. 10, pp. 11994–12000, Dec. 2009.

[13] A. Hojage, "Race detection using mutated salp swarm optimization algorithm based DBN from face shape features," *Multimedia Res.*, vol. 4, no. 2, pp. 7–14, Apr. 2021.

[14] B. Selvakumar and K. Muneeswaran, "Firefly algorithm based feature selection for network intrusion detection," *Comput. Secur.*, vol. 81, pp. 148–155, Mar. 2019.

[15] E. G. Learned-Miller, "Entropy and mutual information," Dept. Comput. Sci., Univ. Massachusetts, Amherst, MA, USA, Tech. Rep., Sep. 2013.

[16] F. Tu, S. Yin, P. Ouyang, S. Tang, L. Liu, and S. Wei, "Deep convolutional neural network architecture with reconfigurable computation patterns," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 8, pp. 33–2220, Apr. 2017.

[17] W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, and X. Xie, "Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 30, no. 1, pp. 1–7, Mar. 2016.

[18] X. Meng, Y. Liu, X. Gao, and H. Zhang, "A new bio-inspired algorithm: Chicken swarm optimization," in *Proc. Int. Conf. Swarm Intell.*, Oct. 2014, pp. 86–94.

[19] M. Khishe and M. R. Mosavi, "Chimp optimization algorithm," *Expert Syst. Appl.*, vol. 149, Jul. 2020, Art. no. 113338.

[20] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.

[21] J. Toldinas, A. Venčkauskas, R. Damaševičius, Š. Grigaliūnas, N. Morkevičius, and E. Baranauskas, "A novel approach for network intrusion detection using multistage deep learning image recognition," *Electronics*, vol. 10, no. 15, p. 1854, Aug. 2021.

[22] G. De Carvalho Bertoli, L. A. P. Junior, O. Saotome, A. L. Dos Santos, F. A. N. Verri, C. A. C. Marcondes, S. Barbieri, M. S. Rodrigues, and J. M. P. De Oliveira, "An end-to-end framework for machine learning-based network intrusion detection system," *IEEE Access*, vol. 9, pp. 106790–106805, 2021.

[23] G. Andresini, A. Appice, and D. Malerba, "Autoencoder-based deep metric learning for network intrusion detection," *Inf. Sci.*, vol. 569, pp. 706–727, Aug. 2021.

[24] *NSL-KDD Dataset Taken*. Accessed: Dec. 2021. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html

[25] *BoT-IoT Dataset Taken*. Accessed: Dec. 2021. [Online]. Available: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/bot_iot.php

BHUSHAN DEORE received the B.E. degree in electronics and telecommunications engineering from North Maharashtra University, Jalgaon, India, in 2010, and the M.E. degree in electronics engineering from the University of Mumbai, India, in 2013. He is currently pursuing the Ph.D. degree in electrical engineering with the Veermata Jijabai Technological Institute, Mumbai, India. He is currently an Assistant Professor with the Department of Electronics and Telecommunication Engineering, Ramrao Adik Institute of Technology, Navi Mumbai. His research interests include intrusion detection, machine learning, and cybersecurity.

SURENDRA BHOSALE received the bachelor's degree in electrical engineering from Shivaji University, Kolhapur, India, in 1987, and the master's degree in electrical engineering from the University of Mumbai, India, in 2001, and the Ph.D. degree in electrical engineering from the Veermata Jijabai Technological Institute, Mumbai University, India, in 2016. He has more than 34 years of experience in teaching. Currently, he is the Head of the Department and an Associate Professor in electrical engineering with the Veermata Jijabai Technological Institute, Mumbai. His research interets include wireless communications and routing algorithms, applications of machine learning, and deep learning algorithms.