Received: April 26, 2023. Revised: July 12, 2023. 60

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An Efficient Intrusion Detection System in Cloud Network Based on Deep Learning and Improved Marine Predators-Particle Swarm Optimization

Badari Narayan Vijayeendra Srinivasa Murthy[1]* **Siddappa Madappa[1]**

[1]*Department of Computer Science and Engineering, Sir Siddhartha Academy of Higher Education, Tumkur, India*
* Corresponding author's Email: badarivs@gmail.com

Abstract: The growth of data is an ongoing phenomenon, driven by the expansion of cloud networks, the rapid development of computer systems, and the proliferation of highly beneficial applications. To defend networks from various attacks, such as denial-of-service (DoS), SlowHTTPTest, Hulk, Slowloris, and LOIC-type DDoS attacks, the area of intrusion detection has grown significantly. Traditional intrusion detection systems are combined with a variety of security methods to provide security. However, such a system has limitations when it comes to effectively analyzing vast amounts of data. To detect the vast amount of different intrusion attacks and to select the optimum features, an improved marine predators algorithm (IMPA) is combined with a swarm optimization algorithm named particle swarm optimization (PSO) is proposed. The experiments are conducted using the open-source available dataset referred to as the CSE-CIC-IDS2018 dataset. Here, the classification is done using the Bi-LSTM classifier, an improved version of the LSTM classifier. The experimental results showed that the performance of the proposed IMPAPSO-Bi-LSTM model has achieved high accuracy of 99.40%, recall of 98.20% and f1-score of 98.62% compared to the existing techniques such as recurrent convolutional neural network (R-CNN) and federated intrusion detection system integrated with the blockchain (FIDChain).

Keywords: Cloud network, Intrusion detection systems, Feature selection, Marine predators' algorithm, Particle swarm optimization.

1. Introduction

Currently, organizations are assessing the feasibility of cloud computing as a potential option. A decade ago, cloud computing wasn't even considered an option. The concept of cloud computing was to supply a huge number of services from a single source [1]. Cloud computing necessitates the virtualization of storage resources and associated applications. A hypervisor inspector in a cloud system examines the shared hardware architecture and the general operation of the operating systems [2]. The cloud is becoming more popular, and numerous customers use logical isolation methods to share the same cloud resources [3]. An increasing number of companies, banks, and governments are adopting cloud computing services as they become more common. This modification also made these systems vulnerable to numerous hacker and intruder intrusions, necessitating the adoption of advanced security measures [4]. Due to a large amount of data or an absence of network intrusion detection abilities, attackers can enter business computer systems through Trojans, spyware, and even complex bugs, putting the firms' data security at risk [5].

Cloud computing is vulnerable to classic attacks such as IP spoofing, routing and remote access protocol (RAP) attacks, flooding, DoS, DDoS and so on. Intrusion manifestations include unexpected results when performing various client instructions, delayed framework execution, unexpected framework crashes, and changes in kernel data structures, as well as unusually poor system performance. Even though Intrusion prevention approaches such as client authentication, avoiding programming mistakes, and data protection have existed but computer intrusion prevention action is insufficient [6]. As a result, intrusion detection is

necessary as an additional defense to assure a computer system's security [7]. The intrusion detection system (IDS) is a critical component for maintaining cloud security and detecting dynamic intrusions in the cloud. An IDS primarily monitors the user's unlawful conduct in the cloud and decreases the security risk [8]. There aren't any papers that provide thorough evaluations of an unsupervised clustering-based ensemble IDSs classifier's capabilities with complete comparison evaluation utilizing existing IDS datasets like CIC-IDS2017, UNSW-NB15, as well as NSL-KDD [9]. In addition, the suggested system offers a reliable hybrid feature selection approach, known as HFS that effectively chooses pertinent attributes from among the imbalance and numerous instances of the different data sets to allow accurate, prompt attack classification [10].

In this research, an improved behaviour of MPA such as the predictors random walk is boosted by the high-tailed Weibull distribution is combined with the particle swarm optimization (PSO). The output of IMPAPSO is given to the Bi-LSTM for classification for effective network intrusion detection. IMPA is combined with a swarm optimization algorithm named PSO for effective network intrusion detection. But the MPA's main disadvantage was that, the number of iterations was divided between the main phases of the algorithm. Therefore, it required huge number of iterations, especially with nonlinear optimization problems. For solving this point, an improved version of the MPA was used based on distribution and particle swarm optimization. As a result, an IMPAPSO with Bi-LSTM method is introduced. The following is the current research's main contribution:

1) In this paper, an openly available dataset named CSE-CIC-IDS2018 is employed for the input features which provides 16,233,002 features accumulated over 10 days of network activity.
2) IMPAPSO is proposed as a feature selection algorithm to select the optimum features for precise classification.
3) An improved version of the LSTM classifier named as Bi-LSTM classifier is used for accurate classification. Then the common performance measures which include accuracy, recall, f1-score, etc are utilized to assess the proposed IMPAPSO approach's performance.

This research paper is structured as follows: the related studies of network intrusion detection are described in section 2. The proposed methodology is briefly explained in section 3 whereas, the experimental findings as well as the dataset study are presented in section 4. The conclusion of this research work is given in section 5.

2. Related work

Thilagam and Aruna [11] suggested a novel recurrent convolutional neural network (R-CNN) with the ant lion optimization method (ALO) for intrusion detection. The suggested R-CNN network was built by linking the LSTM and CNN layers. The suggested technique increases efficiency by using a prelu layer instead of a predefined relu layer and using ALO optimization to achieve a low error rate and a high classification rate. The suggested technique obtained greater classification accuracy by classifying whether the samples were normal or attacked accurately with a decreased error rate. However, not all of the 43 functions are available for IDS. Several characteristics were unconnected and redundant, resulting in a lengthy identification procedure and degraded efficiency.

Farhan and Jasim [12] used deep learning (DL) technology to develop a system for detecting network infiltration. To identify intrusion during data flow, the suggested technique used the LSTM method, which was utilized to construct the neural network that was applied to the CSE-CIC-IDS2018 actual data set. The dataset covers the most recent attacks and was organized according to the proportion of detection in the data. As a result, the suggested strategy obtained greater accuracy while the loss function was reduced in the training and testing stages. However, there was an imbalance in the CSE-CIC-IDS2018 dataset, as well as its vast size, which caused accuracy computing to fail.

Kim [13] demostrated a convolutional neural network (CNN)-based intrusion model to identify denial of service (DoS) attacks. The suggested solution employs CNN in conjunction with the CSE-CIC-IDS2018 dataset, which was the most recent IDS dataset and contains more complex DoS attacks. The CNN retrieves the image's unique characteristics while retaining spatial knowledge and decreases the quantity of the feature data by adding a pooling layer to the convolution layer. As a result of employing binary and multiclass classification methods, the suggested method demonstrated greater accuracy in spotting DoS attacks. However, it was discovered that the kernel size had no significant influence on both binary and multiclass classifications.

For identifying network attacks, Hagar and Gawali [14] proposed apache spark and deep learning

models. The suggested technique implements network intrusion detection systems (NIDS) using the CSE-CIC-IDS 2018 dataset. As a result, by employing the logistic regression multinomial technique, the suggested apache spark obtains the maximum accuracy in identifying network attacks. The attacks documented in these datasets, however, were not applied to the current era.

Liu [15] developed a unique difficult set sampling technique (DSSTE) approach that allows the classification model to improve its learning from unbalanced network data. For improved classification, the suggested technique separates the unbalanced training set into near and far-neighbor sets using the edited nearest neighbor (ENN) algorithm. As a consequence, the presented approach improved classification accuracy by enhancing the minority's learning under challenging samples. As a result, deep learning was most limited in its ability to comprehend the preprocessed aspects and does not employ its computerized feature extraction capabilities.

Fatemeh Ahmadi Zeidabadi [16] introduced a new optimization algorithm known as puzzle optimization algorithm. This algorithm is introduced to solve several optimization issues. The objective of the proposed POA was the mathematical simulation of the process that involves in solving a puzzle as an evolutionary optimizer. The proposed POA has no control parameters and does not require any parameter setting which is an advantage of this algorithm. The performance of this algorithm is tested on twenty-three objective functions and the results indicate that the high and acceptable ability of POA will resolve the optimization issues.

Purba Daru Kusuma [17] proposed a guided pelican algorithm (GPA) to solve and optimize the portfolio issues. The GPA is the improved version of pelican optimization algorithm (POA). First, GPA replaced the randomized target with the global best solution as a deterministic target in phase one. Second, GPA replaced the pelican's current location with the search space size in determining the local search space size in phase two. Third, GPA implemented multiple candidates in both phases rather than a single candidate as it was used in the original POA. The result shows that GPA outperforms all sparing algorithms in optimizing the portfolio problem.

Purba Daru Kusuma [18] proposed a stochastic komodo algorithm (KMA) which was a modified version of original KMA. This method has focused on conducting diversification and intensification without redundancy. The proposed algorithm was tested using five unimodal functions and five multimodal functions. The female dominant formation has achieved better results as per the observations. By appropriate tuning, this proposed algorithm is competitive enough with the original KMA and other well-known algorithms.

Eman Ashraf [19] presented an edge-cloud intrusion detection mechanism known as federated intrusion detection with the integration of blockchain (FIDChain) which used lightweight artificial neural networks to provide data privacy and prevent intrusion attacks in healthcare IoT devices. The performance of the presented approach was evaluated on CSE-CIC-IDS2018, Bot Net IoT, and KDD cup 99 datasets and achieved accurate results due to the ability of FIDChain to detect multiple botnets, less detection time, less computing, and processing capacity. However, the model parameters transmitted in each round may be used to recover the confidential data which leads to security risks.

The recent optimization techniques such as ALO, POA, GPA, and KMA are the state-of-the-art methods which deals with the issues that are related to optimization. The ALO algorithm [11] increased efficiency by using a ReLu layer instead of a predefined relu layer and achieve a low error rate and a high classification rate for intrusion detection. The POA [16] has no control parameters and does not require any parameter setting which is an advantage of this algorithm to solve and optimize the portfolio issues. GPA [17] outperforms all sparing algorithms in optimizing the portfolio problem by replacing the randomized target with the global best solution as a deterministic target. KMA [18] method has focused on conducting diversification and intensification without redundancy. FIDChain [19] has a limitation of data security and high communication overhead. However, the above discussed algorithms are not suitable for intrusion detection in cloud networks.

3. Proposed method

In this section, the overall process of the proposed methodology is explained briefly and also the algorithm of the proposed method is depicted in Fig. 1.

3.1 Data collection

The CSE-CIC-IDS2018 dataset is built from the huge network of simulated client targets with attack computers, which provides 16,233,002 instances accumulated over 10 days of network activity. These are the most recent datasets that contain information about network attacks, including novel forms of attacks. Generally, DL is utilized in many areas to tackle network attack detection difficulties. Slow

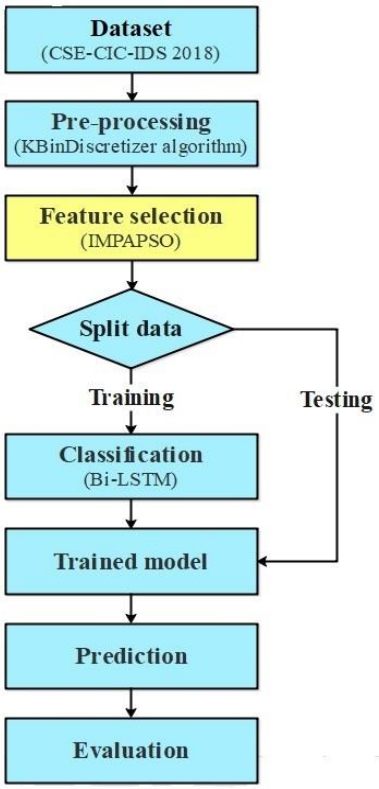

Figure. 1 Algorithm of proposed work

HTTP Test, hulk, slow loris, and LOIC-type DDoS attacks are included in the CSE-CIC-IDS 2018 dataset. A variety of these datasets, each with identical attributes, are pooled to form a single combined dataset that contained the various forms of DDoS attacks [16]. The collected features are given as input to the initial stage named pre-processing to enhance the properties of the features.

3.2 Pre-processing

Pre-processing involves data tuning and normalization because it uses features from the CSE-CIC-IDS2018 dataset to conduct the pre-processing. The records will be enumerated and regularised beforehand, and some of the point values in the original dataset are conceptual, constant, and of various other kinds. For processing input, these numbers must be changed to numerical type. Additionally, some numbers might not fall within the defined range because of various representations [17]. Initially, it is the strategy of only pre-processing discrete characteristics before discretization. Rather than employing a Min-Max scale for discrete features, the KBinDiscretizer method is used which converts 22 values to 80 characteristics by conducting sorting periodically. The KBinDiscretizer method similarly performs processing at specified cycles, changing 60 features from CSE-CIC-IDS 2018 data to 350 features [18]. After that, the preprocessed features are given as an input feature for conducting the feature selection process using various algorithms which are explained in the following section.

3.3 Feature selection

This section covers the phases of the specified FS model which are utilized to improve privacy and security. Determining the pertinent components that are given priority depends on their value, which is considered the major goal of the aforementioned phases. So, to achieve this major goal the binary variant of IMPAPSO is used. But the major drawback of MPA was that the number of iterations was split among the computation major stages, which discouraged academics from using it for their applications despite its simplicity in execution. Consequently, a significant number of repetitions was necessary, particularly for nonlinear optimization issues. An enhanced MPA based on distribution and particle swarm optimization was put forth to address this issue. To identify the intrusion in the network, this research work presents an examination of the recently created MPA with few adjustments and hybridization among the MPA and PSO. The suggested feature selection method (IMPAPSO) reduces the training data by only selecting features that are related to those in the Boolean class of the existing system. Then the selected features are divided, where nearly 80% of features are given as input for the classification process using the BI-LSTM classifier such as training. Where the remaining features are taken for the testing process which is used to determine the effectiveness of the trained model in terms of accurate prediction. Finally, the trained model is evaluated using common performance metrics such as detection accuracy, F1-score, precision, and recall.

3.3.1. MPA formulation

The primary source of inspiration for the MPA was ocean predators' digging strategy. The "survival of the fittest" tenet states that predators must discover the fastest way to maximize the rate of contact with their prey. The forging technique used by many animals is frequently a risky walking technique. The MPA optimization approach is a population-based optimization algorithm, comparable to numerous metaheuristic optimization methods. The initial dispersion of the first population over the investigation area was uniform.

$$X_i = X_{min} + rand\ (X_{max} - X_{min}) \quad (1)$$

From Eq. (1), X_{min} and X_{max} are the variables'

lowest and maximum values; $rand$ denotes a random vector with values ranging from 0 to 1.

3.3.2. Process of MPA

There are three main stages to the MPA exercise. There was a specific set of velocity ratios that defined each phase. The following are the three crucial stages.

Stage 1: This was considered in the simulation application's initial stages because, in this period, the predator was stationary at all while the prey moved with a brownian pattern. Thus, leading to a substantial velocity difference between each other. This stage was mathematically expressed in Eq. (2):

$$\overrightarrow{stepsize_i}=\vec{R}_B \otimes \left(\overrightarrow{Elite_i} - \vec{R}_B \otimes \overrightarrow{Prey_i}\right) i = 1,..,n \quad (2)$$

Where, $\overrightarrow{R_B}$ is a random-numbers vector, R is a random numbers vector among 0 and 1.

Stage 2: Exploration and exploitation were both incorporated into the second stage. There were two divisions created from the population. It was decided to conduct exploration in the initial segment and exploitation in the second. The prey is accountable for the exploitation stage. Contrarily, the predator is in control of the investigation. Eq. (3) reflects the rule that was used for the i population's initial portion.

$$\overrightarrow{stepsize_i}=\vec{R}_L \otimes \left(\overrightarrow{Elite_i} - \vec{R}_L \otimes \overrightarrow{Prey_i}\right) \quad (3)$$

Where, $i = 1, \ldots n/2$, $\vec{R}_L$ denotes the vector of random numbers and the prey motion is simulated by multiplying $\vec{R}_L$ and the Prey. $\vec{R}_L \otimes \overrightarrow{Prey_i}$ simulates the prey motion. Eq. (4) states the rule that applies to the population's second portion.

$$\overrightarrow{stepsize_i}=\vec{R}_B \otimes \left(\vec{R}_B \otimes \overrightarrow{Elite_i} - \overrightarrow{Prey_i}\right) \quad (4)$$

Where, $i = {}^{n}/_{2}, \ldots., n$, $\vec{R}_B \otimes \overrightarrow{Elite_i}$ indicates that the trajectory of the predator is a Brownian motion and the prey updates its position according to the movement of the predator

Stage 3: The exploitation stage is the final step which is formally expressed in Eq. (5):

$$\overrightarrow{stepsize_i}=\vec{R}_L \otimes \left(\vec{R}_L \otimes \overrightarrow{Elite_i} - \overrightarrow{Prey_i}\right) \quad (5)$$

Where, $i = 1, \ldots, n$, $\vec{R}_L$ is the Levy motion

3.3.3 Fish aggregating devices effect

Environmental concerns influence the behavior of marine predators and environmental problems include FADs. In the exploratory field, FADs might represent local optima and longer leaps during the optimization procedure might help to escape from falling in local optima. The FADs impact is mathematically expressed in Eq. (6):

$$\overrightarrow{Prey_i} = \begin{cases} \overrightarrow{Prey_i} + CF\left[\vec{X}_{min} + \vec{R} \otimes \left(\vec{X}_{max} - \vec{X}_{min}\right)\right] \otimes \vec{U} \\ if\ r \le FADs \\ \overrightarrow{Prey_i} + [FADs(1-r)+r]\left(\overrightarrow{Prey_{r1}} - \overrightarrow{Prey_{r2}}\right) \\ if\ r > FADs \end{cases} \quad (6)$$

Where, $\vec{U}$ is the binary vector which converts the array to 0 (array < 0.2) and 1 (array > 0.2), r is the random number among [0, 1], $\vec{X}_{min}$ and $\vec{X}_{max}$ are the vector containing the lowest and highest bounds, and r1 and r2 are the random values in the prey matrix.

3.3.4. Particle swarm optimizer (PSO)

PSO represents a widely used swarm optimization approach that was modeled after the behavior of swarms, including their communication methods when sharing information, pursuing prey, migrating, and gathering. The population, which uses experience to keep themselves up to speed, is represented by the particles. Such particles are randomly distributed via the exploration field with j^{th} dimensional locations (x_i) and velocities (v_i). They then update positions by the individual and worldwide best locations over the simulation procedure. Eqs. (7) and (8) serves as an algebraic representation of it,

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1} \quad (7)$$

$$v_{ij}^{t+1} = wv_{ij}^{t} + a_1 r_1 \left(x_{ij}^{p(t)} - x_{ij}^{t}\right) - a_2 r_2 \left(x_j^{g(t)} - x_{ij}^{t}\right) \quad (8)$$

Where, x_{ij} and v_{ij} are the location and velocity of the i^{th} particle in j^{th} dimension. The t represents ongoing iteration, w denotes inertia weight, a_1 and a_2 are the acceleration coefficients, $x_{ij}^{p(t)}$ is ith agent's best location, $x_j^{g(t)}$ is the global best agent, and r_1 and r_2 are the random numbers among [0, 1].

Here, the fitness function is assessed repeatedly then the global best agents are updated accordingly. Until reaching the stopping criteria, the above-stated process is continued.

Algorithm 1: Pseudocode of proposed IMPAPSO

1. Determine the MPAPSO parameters; N agents, T_{max} maximum number of iteration and optimization problem bounds; X_{max}, X_{min}
2. Compute the initial random solutions using Equation (1).
3. Set t = 1.
4. While (t > = T_{max})
5. Evaluate the fitness function ($fitness_i$).
6. Determine the Elite matrix (best solution obtained so far).
7. Use Equation (9) to calculate the Weibull distribution (W_D).
8. If (t<$\frac{T_{max}}{3}$)
9. Update the agent's positions using Equation (2)
10. If ($\frac{T_{max}}{3} < t < \frac{2\,T_{max}}{3}$)
11. For ($I = 1, \dots, N/2$).
12. Update the agent's positions using Equation (3)
13. End for
14. For ($i = N/2, \dots, N$).
15. Update the agent's positions using Equation (4)
16. End for
17. Else if (t > $\frac{2\,T_{max}}{3}$)
18. If $prob_i < rand_i$
19. Update the agent's positions using Equation (5)
20. Else
21. Update the agent's positions using Equations (7) and (8)
22. End if
23. End if
24. Using FADs effect and Equation (6) to update current agent.
25. T = t + 1
26. End while
27. Display the best solution

3.3.5. Proposed IMPAPSO structure

The positions of the agents are adjusted by the IMPAPSO algorithm which considers the essential parameters while detecting the intrusions. As a result, two tactics are used in this work to improve the behavior of MPA, where the predators' random walk is enhanced by a high tailed Weibull distribution including its exploitation phase. In MPA, using Brownian movement the agents' random walk is regulated to change beast behavior and is changed by the high-tailed Weibull distribution in the first adjustment, utilizing the following calculation, to increase the exploration stage.

$$W_D(U) = exp^{\left(\frac{u}{v}\right)^{\varsigma}} \tag{9}$$

Where, ς, v are shape and scale parameters and $W_D(U)$ is the high-tailed Weibull distribution.

To improve the basic MPA trial's exploitation process, the PSO controller is employed which enhances the algorithm's closure performance. According to a probability value, the agents in the third stage of the MPA could implement the MPA or PSO, as mentioned in Eq. (10).

$$x_i = \begin{cases} Operator\ of\ MPA & prob_i < rand_i \\ Operator\ of\ PSO & Otherwise \end{cases} \tag{10}$$

Where, $prob_i$ is the generated value based on a fitness function.

The MPA operators of Eq. (4) will be executed; If the value of $prob_i < rand_i$, else the PSO operators of Eq. (7) are executed.

3.3.6. Fitness function

This phase begins with turning x_i, into its Boolean value Bx_i. Where $i = 1,2, \dots, N$.

$$Bx_i = \begin{cases} 1\ if\ x_{ij} > 0.5 \\ 0 \quad otherwise \end{cases} \tag{11}$$

According to the output of Eq. (11), the number of features is decreased by eliminating the inappropriate features that related to zeros in x_i. After that, the fitness value is calculated by utilizing Eq. (12).

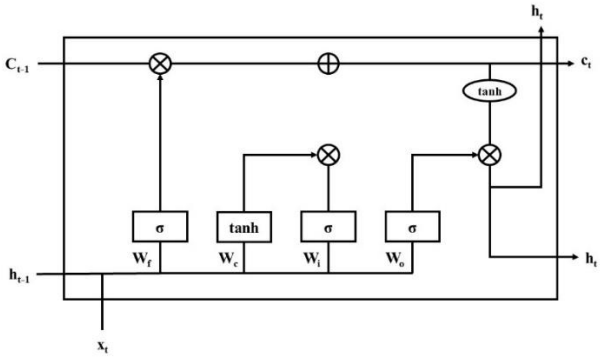

Figure. 2 LSTM structure

$$Fit_i = \lambda \times \gamma i + (1 - \lambda) \times \left(\frac{|BX_i|}{D}\right) \quad (12)$$

Where, $\lambda \in [0,1]$ - weights used to manage the balance among the related features ratio $\left(\frac{|BX_i|}{D}\right)$ and classification error γi. By utilizing training set, the γi is calculated according to the BI-LSTM classifier. Then, the best Fit with its corresponding agent x_b are determined.

3.4 Classification (BI-LSTM)

Based on cells and the forget gate, LSTM can retain significant knowledge over the long term [18]. Utilizing the self-feedback technique, an LSTM demonstrated a benefit in managing long-term reliance issues in the hidden layer, in which the arrhythmia signals needed data from the preceding steps as well. For keeping the data in the LSTM model, the memory cell primarily consists of three gates: an input gate, a forgetting gate, as well an output gate. This arrangement helped to address issues with long-term features. The structure of the LSTM model. Is depicted in Fig. 2.

The output is stated as h_t; memory cell value is stated as c_t; h_{t-1} is stated as the output of the LSTM cell. It has managed the subsequent stages that are described as follows:

- Bias term is signified as b_c; Memory cell is stated as $\widetilde{c_t}$; weight cell is stated as W_c which is explained in Eq. (13),

$$\widetilde{c_t} = tanh\,(W_c.[h_{t-1}, x_t] + b_c) \quad (13)$$

- The input gate is signified as i_t , W_i is signified as a weight matrix and the sigmoid function is stated as σ, b_i signifies the bias, the present input data manages the state value with the help of i_t, i_t is stated in Eq. (14),

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \quad (14)$$

- The f_t signifies forget gate value which assesses the memory depending on predicted data. Also, it is updated by managing f_t which is characterized in Eq. (15),

$$f_t = \sigma(W_i.[h_{t-1}, x_t] + b_f) \quad (15)$$

- The unit state value is signified as c_{t-1}, the memory cell is signified as c_t and that is expressed in Eq. (16),

$$c_t = f_t * c_{t-1} + i_t * \widetilde{c_t} \quad (16)$$

From Eq. (16), '*' is recognized as a dot product.

- o_t is referred to as the output gate which is calculated by the memory cell state and that was regulated by the output gate. o_t is stated in below Eq. (17),

$$o_t = \sigma(W_0.[h_{t-1}, x_t] + b_0) \quad (17)$$

- The output h_t attained from LSTM is designed by Eq. (18),

$$h_t = o_t * tanh\,(c_t) \quad (18)$$

The hidden unit function $\vec{h}$ of a hidden forward layer at every interval step t is calculated on the input current step x_t as well as prior hidden state h_{t-1}. The hidden unit function $\overleftarrow{h}$ of a hidden backward layer is calculated on x_t and future hidden state $\overleftarrow{h}_{t+1}$. The backward as well as forward context depictions created through $\overleftarrow{h}_t$ and $\vec{h}_t$, which are connected into a long vector.

To store information for a long period, the LSTM model reads, resets, and updates utilizing memory cells as well as control gates. In the LSTM, internal sharing system factors regulate the values for the weight matrix's output dimensions. Whereas the Bi-LSTM is a better iteration of the LSTM, capable of high-level abstraction with nonlinear change of intrusion data, analysis of two-way data, as well as precise computing [18]. The recurrent network layer coexists with the first layer in bidirectional LSTMs and is repeated. The input sequence is processed by the first layer, similarly, the second layer is given the input sequence's reversed copy [19]. With Bi-LSTMs, the recurrent network layer is duplicated and functions in tandem with the top layer. The first layer handles the input sequence, whereas the reversed duplicate of the input pattern is passed to the second layer [20]. Eqs. (19) and (20) describes the method of calculation:

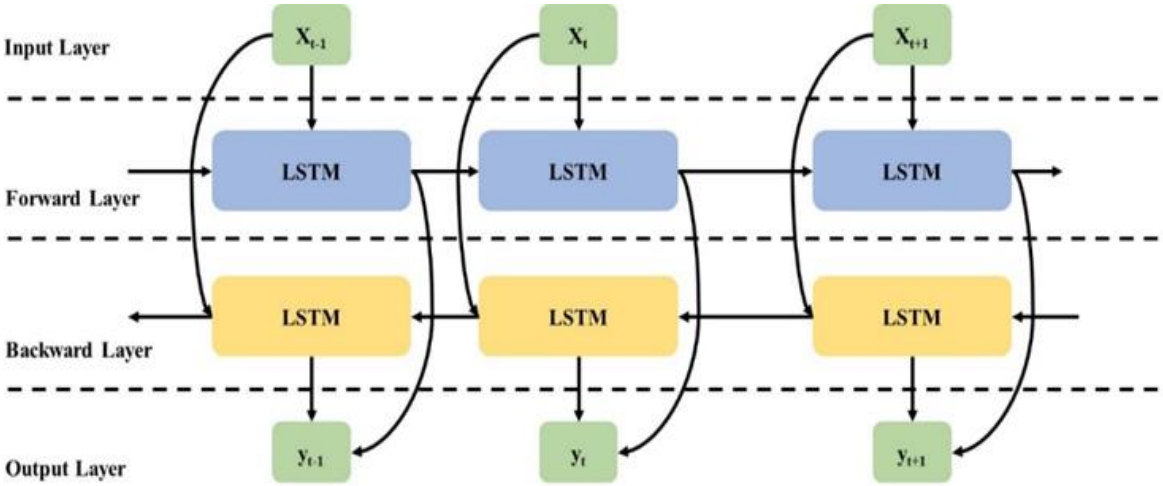

Figure. 3 Bi-LSTM architecture

Table 1. Mathematical representation of performance measures

Performance Metrix	Equation
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1- Score	$\frac{2\times P\times R}{P+R}$
Accuracy	$\frac{TP+TN}{TP+FN+TN+FP}$
Error Rate	$\frac{FP+FN}{TP+TN+FP+FN}$

$$\vec{h}_t = f\left(\vec{W}.\ x_t + \vec{W}.\vec{h}_{t-1} + \vec{b}\right) \quad (19)$$

$$y_t = g\left(U.\ \left[\overleftarrow{h};\vec{h}\right] + c\right) \quad (20)$$

Where,

$\vec{W}$ and $\overleftarrow{W}$ - hidden layer parameters of network,

x_t - input data,

$\overleftarrow{h_t}$ and $\vec{h}_t$ - the output results of the two LSTM layers at time t,

$\vec{b}$ and $\overleftarrow{b}$ - the offset value,

y_t - Bi-LSTM output. The bi-directional recurrent neural network is shown in Fig. 3.

4. Experimental results

This section provides the results and analysis of the present research. Here, the system is implemented using Intel core-i5 processor at 3.40 GHz on Ubuntu 16.4 OS with storage of 8 GB. Using various optimization algorithms such as ALO, grey wolf optimization (GWO) and whale optimization algorithm (WOA), puzzle optimization algorithm (POA), guided pelican optimization (GPA), and stochastic komodo algorithm (KMA) the performance of the proposed feature selection approach (IMPAPSO) is assessed. Likewise, the performance of the Bi-LSTM classifier is compared with different classifiers such as CNN, recurrent neural network (RNN) as well as LSTM respectively.

$TP - True\ Positive$
$TN - True\ Negative$
$FP - False\ Positive$
$FN - False\ Negative$

The mathematical representation of the performance measures for determining the performance of the proposed IMPAPSO algorithm and BI-LSTM classifier is given in Table 1. The result portion is classified into performance/quantitative analysis and comparative analysis which are briefly described in the following sections.

4.1 Quantitative analysis

In this section, the Bi-LSTM classifier's performance is evaluated with and without the feature selection process and compared with various classifiers such as CNN, RNN and LSTM. In Table 2, the Bi-LSTM classifier's performance is analysed without feature selection and compared with the above-mentioned classifiers.

The graphical depiction of the performance evaluation of Bi-LSTM and various classifiers without feature selection is depicted in Fig. 4 and Fig. 5. From Table 2, the Bi-LSTM classifier obtained a greater accuracy, precision, recall, f1-score, and an error rate of 95.64%, 93.98%, 94.99%, 94.48% and 4.36% respectively. Here, without including the feature selection process the Bi-LSTM classifier performed better when compared to CNN, RNN and LSTM. Following that, the Bi-LSTM classifier's

Table 2. Performance analysis of Bi-LSTM and various classifiers without feature selection

Classifiers	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Error rate (%)
CNN	88.76	87.55	90.00	88.758	11.24
RNN	85.95	85.00	82.90	83.936	14.05
LSTM	90.65	87.64	86.30	86.962	9.35
Bi-LSTM	95.64	93.98	94.99	94.48	4.36

Table 3. Performance analysis of Bi-LSTM model with feature selection

Classifier	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Error rate (%)
CNN	91.19	90.32	92.69	91.488	8.81
RNN	82.14	91.90	89.45	90.656	17.86
LSTM	94.94	90.04	92.60	91.303	5.06
BI-LSTM	99.19	99.47	98.63	99.046	0.81

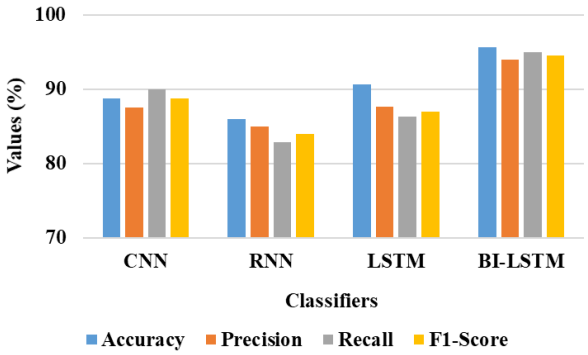

Figure. 4 Graphical representation of performance analysis

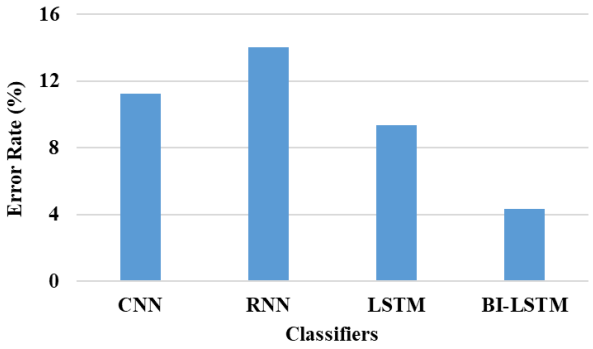

Figure. 5 Error Rate of Bi-LSTM and various classifiers

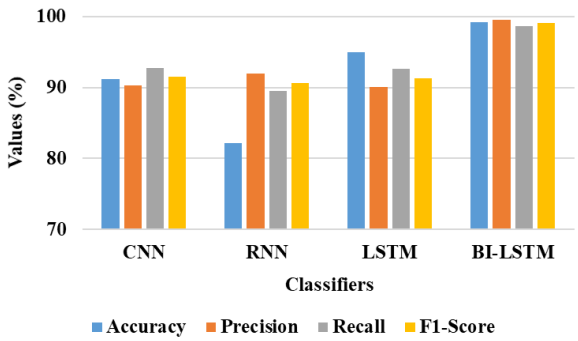

Figure. 6 Graphical depiction of performance evaluation

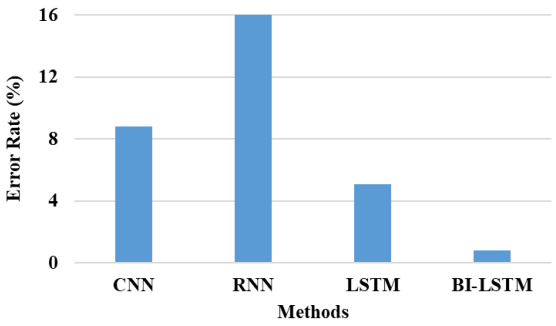

Figure. 7 Error Rate of Bi-LSTM and various classifiers

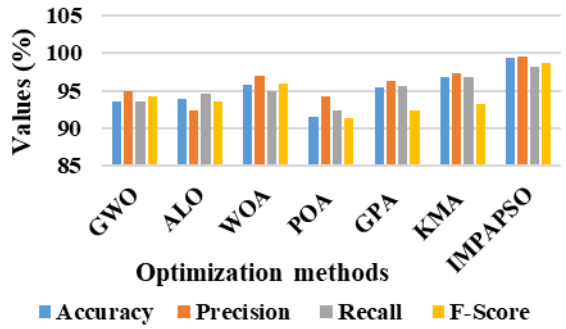

Figure. 8 Graphical representation of IMPAPSO with different optimization algorithm

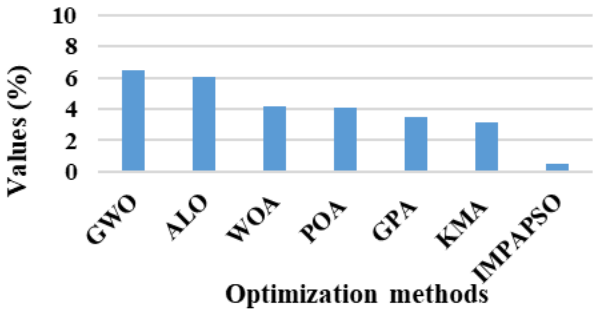

Figure. 9 Error rate of IMPAPSO with different optimization algorithms

Table 4. Performance analysis of IMPAPSO with different optimization algorithms

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Error rate (%)
GWO	93.53	94.88	93.57	94.21	6.47
ALO	93.90	92.45	94.59	93.50	6.10
WOA	95.79	96.90	94.89	95.88	4.21
POA	91.56	94.29	92.32	91.34	4.12
GPA	95.46	96.23	95.65	92.45	3.45
KMA	96.73	97.34	96.74	93.26	3.13
IMPAPSO	99.40	99.59	98.20	98.62	0.51

performance is assessed with feature selection and then compared with the above-mentioned classifiers which are given in Table 3.

The graphical depiction of the performance evaluation of Bi-LSTM and various classifiers with feature selection is depicted in Figs. 6 and 7. From Table 3, the Bi-LSTM classifier obtained a greater accuracy, precision, recall, f1-score, and an error rate of 99.19%, 99.47%, 98.63%, 99.046%, and 0.81% respectively. Here, including the feature selection process the Bi-LSTM classifier performed better when compared to CNN, RNN and LSTM.

Then, the proposed IMPAPSO algorithm is evaluated using various performance metrics and compared with optimization algorithms such as GWO, ALO, WOA, POA, GPA, and KMA. By investigating Table 4, the proposed IMPAPSO algorithm obtained superior performance in intrusion detection when compared to other optimization algorithms. Where, the obtained results of the proposed IMPAPSO method are as follows: Accuracy is 99.40%, Precision is 99.59%, recall is 98.20%, F1-Score is 98.62% and the error rate is 0.51%. The graphical representation of IMPAPSO algorithm with different optimization algorithms is depicted in Figs. 8 and 9.

4.2 Comparative evaluation

The comparative analysis of the IMPAPSO-Bi-LSTM with existing researches is provided in this section. To evaluate the performance of the IMPAPSO-Bi-LSTM model, existing optimization algorithms and techniques such as recurrent convolutional neural network (R-CNN) [11] and FIDChain [19] are used. The measure of result parameters of the proposed model with these techniques and algorithms is provided in Table 5.

The high time consumption issue of existing RCNN is solved by the proposed method. The high convergence speed makes IMPAPSO as an observed feature compared to other techniques. Moreover, the combination of Bi-LSTM with IMPAPSO results in accurate detection of network intrusion and also overcomes the computational overhead of optimization algorithms.

Table 5. Comparative evaluation of IMPAPSO-Bi-LSTM model with existing models

Models	Dataset	Accuracy (%)	Recall (%)	F1-Score (%)
R-CNN [11]	CSE-CIC-IDS2018	94.28	80.53	89.23
FIDChain [19]		85.88	85.81	58.70
IMPAPSO-Bi-LSTM (proposed)		99.40	98.20	98.62

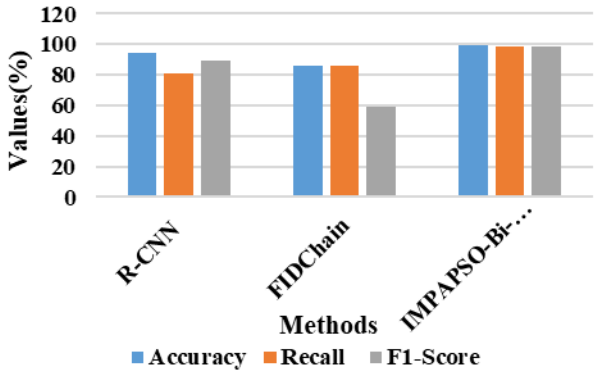

Figure. 10 Graphical representation of a comparison of IMPAPSO-Bi-LSTM with the existing model

The graphical representation of the comparison of IMPAPSO-Bi-LSTM with the existing model is depicted in Fig. 10. In the comparative Table 5, the proposed IMPAPSO-Bi-LSTM model obtained a greater accuracy, recall, and f1-score of 99.40%, 98.20% and 98.62% respectively. Finally, the proposed IMPAPSO-Bi-LSTM model showed superior and more promising performance than the existing models such as R-CNN [11], FIDChain [19].

5. Conclusion

Every day, an immense number of networks and systems are being developed, with many of these advanced systems capable of processing massive

amounts of data. Therefore, it is crucial to verify their level of security. To address this challenge, IMPAPSO is proposed in this paper whereas an openly available dataset named CSE-CIC-IDS2018 is employed for the input features. Secondly, the input features are given to the pre-processing stage to obtain the standardized features, where to perform pre-processing the KBinDiscretizer algorithm is used. Thirdly, to select the best features, the proposed IMPAPSO approach is used to perform precise classification. The selected optimum features are given as the input for Bi-LSTM classifier, an improved version of the LSTM classifier to perform classification. Then the Bi-LSTM classifier's performance is evaluated with and without the feature selection process and contrasted with traditional classifiers. From the analysis, the performance of the Bi-LSTM classifier is higher than the existing classifiers. Then the performance of the developed IMPAPSO-Bi-LSTM model is compared with the existing model that is developed for intrusion detection. In that final comparison, the developed IMPAPSO-Bi-LSTM model achieves greater performance than the existing models based on the accuracy (99.40%), recall (98.20%) and f1-score (98.62%). In the future, some other different optimization algorithms will be implemented based on Deep Learning to obtain more detection accuracy.

Nomenclature

Terms	Representation
X_{min} and X_{max}	variables' lowest and maximum values
$rand$	random vector
$\overrightarrow{R_B}$	random-numbers vector
t	ongoing iteration
w - inertia weight.	inertia weight
a_1 and a_2	acceleration coefficients
$x_{ij}^{p(t)}$	ith agent's best location
$x_j^{g(t)}$	global best agent.
r_1 and r_2	random numbers among [0, 1]
x_{ij}	location of the i^{th} particle in j^{th} dimension
v_{ij}	velocity of the i^{th} particle in j^{th} dimensio
$W_D(U)$	high-tailed Weibull distribution
ς, v	shape and scale parameters
b_c and W_c	Bias term and weight matrix
i_t	input gate
σ	sigmoid function
b_i	bias
f_t	forget gate value
c_{t-1}	unit state value
o_t	Output gate value

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

For this research work all authors' have equally contributed in Conceptualization, methodology, validation, resources, writing—original draft preparation, writing—review and editing.

References

[1] P. Singh and V. Ranga, "Attack and intrusion detection in cloud computing using an ensemble learning approach", *International Journal of Information Technology*, Vol. 13, No. 2, pp. 565-571, 2021.

[2] A. N. Jaber and S. U. Rehman, "FCM–SVM based intrusion detection system for cloud computing environment", *Cluster Computing*, Vol. 23, No. 4, pp. 3221-3231, 2020.

[3] S. Krishnaveni, S. Sivamohan, S. S. Sridhar, and S. Prabakaran, "Efficient feature selection and classification through ensemble method for network intrusion detection on cloud computing", *Cluster Computing*, Vol. 24, No. 3, pp. 1761-1779, 2021.

[4] A. Aldallal and F. Alisa, "Effective Intrusion Detection System to Secure Data in Cloud Using Machine Learning", *Symmetry*, Vol. 13, No. 12, p. 2306, 2021.

[5] Z. Wang, D. Jiang, L. Huo, and W. Yang, "An efficient network intrusion detection approach based on deep learning", *Wireless Networks*, 2021.

[6] D. Srilatha and G. K. Shyam, "Cloud-based intrusion detection using kernel fuzzy clustering and optimal type-2 fuzzy neural network", *Cluster Computing*, Vol. 24, No. 3, pp. 2657-2672, 2021.

[7] M. Yan, Y. Chen, X. Hu, D. Cheng, Y. Chen, and J. Du, "Intrusion detection based on improved density peak clustering for imbalanced data on sensor-cloud systems", *Journal of Systems Architecture*, Vol. 118, p. 102212, 2021.

[8] V. Prabhakaran and A. Kulandasamy, "Hybrid semantic deep learning architecture and optimal advanced encryption standard key management scheme for secure cloud storage and intrusion detection", *Neural Computing and Applications*,

Vol. 33, No. 21, pp. 14459-14479, 2021.
[9] C. Zhang, Y. Chen, Y. Meng, F. Ruan, R. Chen, Y. Li, and Y. Yang, "A novel framework design of network intrusion detection based on machine learning techniques", *Security and Communication Networks*, Vol. 2021, p. 6610675, 2021.
[10] E. Jaw and X. Wang, "Feature selection and ensemble-based intrusion detection system: an efficient and comprehensive approach", *Symmetry*, Vol. 13, No. 10, p. 1764, 2021.
[11] T. Thilagam and R. Aruna, "Intrusion detection for network based cloud computing by custom RC-NN and optimization", *ICT Express*, Vol. 7, No. 4, pp. 512-520, 2021.
[12] B. I. Farhan and A. D. Jasim, "Performance analysis of intrusion detection for deep learning model based on CSE-CIC-IDS2018 dataset", *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 26, No. 2, pp. 1165-1172, 2022.
[13] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, "CNN-based network intrusion detection against denial-of-service attacks", *Electronics*, Vol. 9, No. 6, p. 916, 2020.
[14] A. A. Hagar and B. W. Gawali, "Apache Spark and Deep Learning Models for High-Performance Network Intrusion Detection Using CSE-CIC-IDS2018", *Computational Intelligence and Neuroscience*, Vol. 2022, p. 3131153, 2022.
[15] L. Liu, P. Wang, J. Lin, and L. Liu, "Intrusion Detection of Imbalanced Network Traffic Based on Machine Learning and Deep Learning", *IEEE Access*, Vol. 9, pp. 7550-7563, 2021.
[16] F. A. Zeidabadi and M. Dehghani, "POA: Puzzle optimization algorithm", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 1, pp. 273-281, 2022, doi: 10.22266/ijies2022.0228.25.
[17] P. D. Kusuma and A. L. Prasasti, "Guided Pelican Algorithm", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 6, pp. 179-190, 2022, doi: 10.22266/ijies2022.1231.18.
[18] P. D. Kusuma and M. Kallista, "Stochastic Komodo Algorithm", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 4, pp.156-166, 2022, doi: 10.22266/ijies2022.0831.15.
[19] E. Ashraf, N. F. Areed, H. Salem, E. H. Abdelhay, and A. Farouk, "Fidchain: Federated intrusion detection system for blockchain-enabled IoT healthcare applications", *Healthcare*, Vol. 10, No. 6, p. 1110, 2022.
[20] S. Ullah, M. A. Khan, J. Ahmad, S. S. Jamal, Z. E. Huma, M. T. Hassan, N. Pitropakis, Arshad, and W. J. Buchanan, "HDL-IDS: A Hybrid Deep Learning Architecture for Intrusion Detection in the Internet of Vehicles", *Sensors*, Vol. 22, No. 4, p. 1340, 2022.
[21] P. R. M. E. Kanna and P. M. E. Santhi, "Unified deep learning approach for efficient intrusion detection system using integrated spatial–temporal features", *Knowledge-Based Systems*, Vol. 226, p. 107132, 2021.
[22] J. Yoo, B. Min, S. Kim, D. Shin, and D. Shin, "Study on Network Intrusion Detection Method Using Discrete Pre-Processing Method and Convolution Neural Network", *IEEE Access*, Vol. 9, pp. 142348-142361, 2021.
[23] A. Li and S. Yi, "Intelligent Intrusion Detection Method of Industrial Internet of Things Based on CNN-BiLSTM", *Security and Communication Networks*, Vol. 2022, p. 5448647, 2022.
[24] K. Saurabh, S. Sood, P. A. Kumar, U. Singh, R. Vyas, O. P. Vyas, and R. Khondoker, "LBDMIDS: LSTM Based Deep Learning Model for Intrusion Detection Systems for IoT Networks", In: *Proc. of 2022 IEEE World AI IoT Congress (AIIoT)*, Seattle, WA, USA, pp. 753-759, 2022.