CYBERNETICS AND SYSTEMS: AN INTERNATIONAL JOURNAL
https://doi.org/10.1080/01969722.2022.2122008

# Virtual Machine Migration-Based Intrusion Detection System in Cloud Environment Using Deep Recurrent Neural Network

B. V. Srinivas[a], Indrajit Mandal[b], and Seetharam Keshavarao[c]

[a]Department of Information Science and Engineering, Atria Institute of Technology, Bengaluru, Karnataka; [b]Department of Computer Science & Engineering, MINA Institute of Engineering and Technology, Myryalguda, Telangana; [c]R.L Jallappa Institute of Technology, Dodaballapur

**ABSTRACT**
Cloud system attracts users with the desired features, and in the meanwhile, cloud system may experience various security issues. An effective intrusion detection system is offered by the proposed Sail Fish Dolphin Optimization-based Deep Recurrent Neural Network (SFDO-based Deep RNN), which is utilized to identify anomalies in the cloud architecture. The developed SFDO is formed by integrating Sail Fish Optimizer (SFO) and Dolphin Echolocation (DE) algorithm. Virtual Machine (VM) migration and cloud data management are accomplished using ChicWhale algorithm. Some of the attribute features are collected from the cloud model such that these features are grouped using Fuzzy C-Means (FCM) clustering. The feature fusion process is carried out by a Deep RNN classifier that has been trained using the specified SFDO technique in order to achieve the intrusion detection mechanism. The approach with the lowest error value is thought to be the best approach for intruder detection, according to the fitness function. Using the BoT-IoT dataset, the proposed method's accuracy, detection rate (DR), and false-positive rate (FPR) were assessed. The results showed that it outperformed the previous methods with values of 0.9614, 0.9648, and 0.0429, respectively.

**KEYWORDS**
Cloud computing; deep learning; FCM clustering; intrusion detection; VM migration

## Introduction

CCT has been designed and shown a significant role in recent decades in the production and human life (Sabahi 2011; Zhang et al. 2020). CCT is designed by conventional computer technologies, like parallel computing (Chen et al. 2009), grid computing (Lim et al. 2007), network storage, virtualization with the network technologies and utility computing (Soldatos, Serrano, and Hauswirth 2012; Zhang et al. 2020). The revolution of CC offered enhanced flexibility, functionality to the developers, cost-

**CONTACT** B. V. Srinivas srini13986@gmail.com Department of Information Science and Engineering, Atria Institute of Technology, Anandnagar, Hebbal 560024, Karnataka, Bengaluru

effectiveness, individuals and the business to the world, and availability AlKadi et al. (2019). CC is the pool of large resources when compared with the conventional network. However, the people obtained the features from the number of resources of the network based on their requirements (Zhang et al. 2020). The CC system is highly vulnerable to security threats dsue to the growing development. The dynamic nature of the cloud framework makes examining the VM's activity more complex (Abdalla & Ahmed 2020). Hence, neither network based nor host-based intrusion detection mechanism is required to meet in necessity in the virtual environment (Farshchi et al. 2018; Agrawal, Wiktorski, and Rong 2016; Pandeeswari and Kumar 2016).

Before using the CC model, organization should be aware of security issues, like data segregation, regulatory compliance, investigative support, location of data, recovery, privileged user access, and long-term liability (Hui 2018). The cloud system offers security by monitoring the audits, user actions, network traffic, logs, and system configurations (Ganeshkumar and Pandeeswari 2016). Anomaly detection is the major problem that researchers focus on in various application domains and areas. With the dynamics and ever-growing complexity of the CC system, detecting anomalies (Agrawal et al. 2017; Karuppusamy et al. 2022) is highly significant in increasing the system's dependability (Fu, Liu, and Pannu 2012). IDS are crucial elements in a secure network environment because they enable the early detection of malicious behavior and attacks. Using the Map-Reduce framework (Ali et al. 2022), the data gathered from various sources is combined, processed, and compared while analyzing event correlations that could point to malicious or intrusive activity (Mohsin and Abdalla 2020; Sabahi 2011). An intrusion detection method using deep belief network (DBN) (Alzaqebah et al. 2022; Mekhmoukh and Mokrani 2015) improve feature extraction performance.

Reliability of the cloud platform can be enhanced, focusing on the universality and the efficiency of VM detection function and the features of VM in cloud platform such that anomaly detection in VM is required to design in cloud Hui (2018). Anomaly detection in the network traffic is efficiently used to combat the unknown and the known attack types. Various categories of detection methods used in cloud platforms are VM introspection, hypervisor introspection, and misuse detection. Different researchers introduced various detection methods for analyzing and identifying the unusual and abnormal behaviors in different application domains of the cloud platform (Garg et al. 2018). Most of the research works focuses on applying machine learning approaches, data mining (Dhage and Meshram 2012) and soft computing (Mahmood et al. 2012) to design the anomaly detection system. However, these systems (Vieira et al. 2010) still lack accuracy, and it has high computational complexity (Zarrabi and Zarrabi 2012; Ganeshkumar and Pandeeswari 2016). Ye et al. (2016) developed the software-defined network model for capturing the inter VM traffic and detecting the unknown abnormal behaviors in the CC system. An intrusion detection model based on the cloudlet mesh was designed by Chen et al. (2016) to eliminate the intruders and ensure the privacy model in the cloud.

Following is a description of the research's key impact:

- *Developed SFDO-based Deep RNN*: SFDO-based Deep RNN, a deep learning classifier, is used to model an intrusion detection method that uses the cloud framework to detect anomalies.
- *Proposed SFDO*: SFO and DE algorithms are combined to create it for Deep RNN training.

## 2. Motivation

This section discusses various methods for detecting intrusions, along with the advantages and disadvantages of each methods, which motivates the development of the proposed SFDO-based Deep RNN methodology for detecting anomalies in the cloud.

### 2.1. Literature survey

Some of the existing methods used for detecting the anomalies in the cloud are described here. AlKadi et al. (2019) introduced an MLO model to fit network information and the LOF function to find the irregular patterns in the network traffic using the GMM. The false alarm rate and detection rate performance are increased. However, it failed to deploy this model in the real cloud system. In order to identify abnormalities in the cloud, Hui, Y. Hui (2018) introduced an incremental VM workload clustering methodology with globality awareness and locality preservation. It increased the performance and effectively reduced the average running time. However, the evaluation was not accomplished with the accuracy metric. A FCM + ANN was introduced by Pandeeswari and Kumar (2016) to find anomalies in the hypervisor layer. It was more efficient for detecting anomalies by achieving less false alarm rate. Due to the limited number of instances, computing better performance was insufficient. Ganeshkumar *et al.* (2016) introduced an ANFIS scheme to detect anomalies in cloud systems. This method was effective and efficient for finding anomalies in the cloud. This method was not effective for the limited availability of training cases.

Alabdulatif et al. (2017) introduced a lightweight homomorphic encryption model for ensuring privacy and data security. The communication and computational overhead were reduced. It did not enhance the performance. Calheiros et al. (2017) introduced an isolation-based anomaly detection scheme in cloud. This method was efficient for detecting malicious in the large-scale data centers of the cloud. However, it failed to analyze the applicability of this model for customer segmentation. Zhang et al. (2020) introduced an objective-based feature selection scheme to detect anomalies in the cloud framework. This technique decreased the number of characteristics, while improving accuracy and lowering the cost of detection. The effectiveness of this model, however, was not examined. In 2012, Fu *et al.* presented a hybrid self-evolving anomaly detection system with a support vector machine (SVM). It increased detection accuracy but failed to integrate reactive and proactive management schemes to obtain better dependability. Alharbi et al. (2021) proposed the LGBA-NN to select feature subsets and hyperparameters to detect botnet attacks efficiently. The LGBA-NN outperforms other recent algorithms, according to the results. In this regard, the IoT space is overly complex when compared to conventional computing environments. Ali et al. (2022) proposed an ESCNN to analyze the intrusion and threat activities in the IoT. With a minimal false alarm rate, our technique increased attack detection accuracy. High reliability, fast computation, and reduced computation complexity, however, were not achieved by this strategy. Alzaqebah et al. (2022) proposed a tweaked version of the grey wolf optimization algorithm for intrusion detection. This method improves the IDS's ability to identify both regular and unusual network traffic. This method, however, has a slow convergence rate, poor local searching ability, and low solving accuracy.

## 3. Proposed Sail Fish Dolphin Optimization-Based Deep Recurrent Neural Network for Intrusion Detection in Cloud

The proposed SFDO-based Deep RNN for identifying attacks in the cloud framework is described in this section. To begin with, the migration and intrusion detection processes in the cloud are accomplished using a Deep RNN classifier. Here, the data management and VM migration process are carried out using the ChicWhale optimization algorithm. Then, the attributes named as features are collected from the cloud model to increase detection performance in the cloud framework. The FCM is used to group the features after the attributes have been gathered, making the detection process easier. The Deep RNN classifier is then trained using the suggested SFDO method, and the intrusion detection operation is then carried out.

The proposed approach's schematic view is shown in Figure 1.

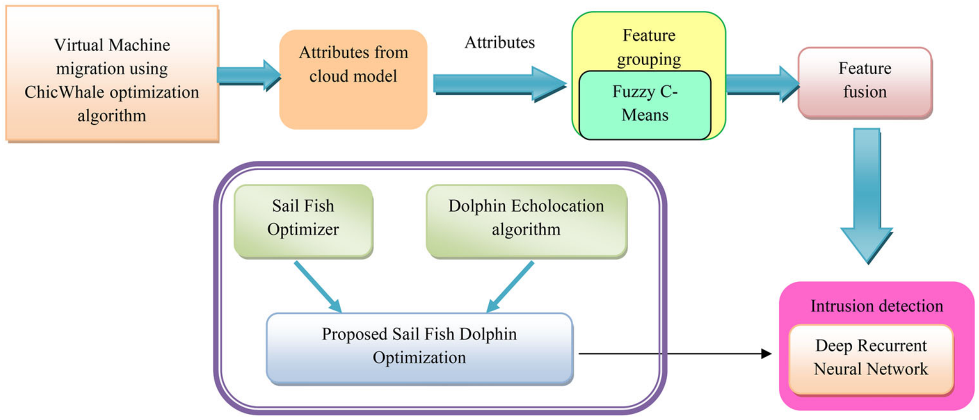

Virtual Machine migration using ChicWhale optimization algorithm
Attributes from cloud model
Attributes
Feature grouping
Fuzzy C-Means
Feature fusion
Sail Fish Optimizer
Dolphin Echolocation algorithm
Proposed Sail Fish Dolphin Optimization
Intrusion detection
Deep Recurrent Neural Network

**Figure 1.** Schematic view of the proposed Sail Fish Dolphin Optimization-based Deep RNN.

## *3.1. Cloud data Management and Migration*

The definition of the symbols used in the paper is provided in Appendix. Let us consider the cloud model contains $A$ number of PMs and $B$ number of VMs such that the PMs and VMs are expressed as, $R = \{R_1, R_2, \ldots, R_A\}$, and $Q = \{Q_1, Q_2, \ldots, Q_B\}$, respectively. The user request the task to execute in VM in the round robin fashion and the total number of tasks assigned to VM is given as$T = \{T_1, T_2, \ldots, T_C\}$. The cloud uses various factors, like memory, bandwidth and CPU, to process the scheduled activities. The cloud uses the resources offered by VM to execute the tasks requested by the user by computing the load using factors like memory, the total number of tasks, network bandwidth and CPU. The load factor is computed based on the utilization of resources with respect to time. However, the resource utilization is computed using the constraints, such as processing components, CPU utilization, bandwidth used, memory, and million instructions per second (MIPS). The number of migrations performed by VM is used to compute the migration cost of VM. The power used for the migration process is termed as energy and hence the energy directly depends on the consumption of power (AlKadi et al. 2019).

The ChicWhale optimization method is used to carry out the migration procedure. The ChicWhale optimization algorithm is developed by the integration of CSO (Meng et al. 2014) and WOA (Mirjalili and Lewis. 2016). The migration process is done by considering the fitness measure that uses parameters such as resource utilization, load, migration energy, and energy. The following is an explanation of the ChicWhale optimization algorithm's algorithmic steps:

i) *Initialization*: The population size and the parameters required in the optimization are specified in the initial step.

ii) *Fitness computation:* The fitness is measured based on constraints, like resource utilization, load, migration energy and energy.

iii) *Update equation:* It contains three groups of swarms, such as roosters, hens and chicks. Here, the position of hen is updated

$$D_{ab}(\alpha+1) = \frac{E}{E-1+w_1 * rand + w_2 * rand} \begin{bmatrix} w_1 * rand D_{c1,b}(\alpha) + w_2 * rand\, D_{c2,b}(\alpha) \\ -\dfrac{D^*(\alpha)(1-EI)}{E}(1-w_1 * rand - w_2 * rand) \end{bmatrix}, \tag{1}$$

where, *rand* specifies a random number that ranges in the interval of $[0,1]$, $c_1$ indicates the index of the rooster, $c_2$ specifies the index of chicken, $D_{ab}(\alpha+1)$ represents the new update location of the rooster at the iteration $\alpha$ and $E$ signifies coefficient vector.

iv) *Termination*: To establish the weights for VM migration, the aforementioned procedures are performed as many times as necessary.

## 3.2. Attribute Collection from the Cloud Model

After the VM migration process using the ChicWhale algorithm has been successfully completed, the attribute collection phase starts. Some of the attributes collected during the extraction process of network flow are specified as follows: $ID_{row}$, $ST_{rcd}$, $fg$, $fg-num$, $TR_{proto}$, $num-TR$, $IP_{src}$, $port_{src}$, $IP_{dest}$, $port_{dest}$, $pks$, $tot-bytes$, $Trans-state$, $num-Trans$, $Last_{rcd}$, $Ar_{seq}$, $Tot_{rcd}$ $Agg_{avg}$, $Agg_{dev}$, $Agg_{total}$, $Agg_{\min}$, $Agg_{\max}$, $pkt_{std}$, $pkt_{dts}$, $byte_{std}$, $byte_{dts}$, $pkt_{total}$, $SD-pkt_{per\sec}$, $DS-pkt_{per\sec}$, $CL$, $ctgy$, and $sub-ctgy$ (Koroniotis *et al.* 2019).

In addition to the attributes listed above, some more features are generated from the cloud model that is specified as follows: $Tot-bytes_{SIP}$, $Tot-bytes_{DIP}$, $Tot-pkts_{SIP}$, $Tot-pkts_{DIP}$, $Tot-proto$, $Tot-port_{dest}$, $Avg_{SIP}$, $Avg_{DIP}$, $num-inbnd_{SIP}$, $num-inbnd_{DIP}$, $Avg-port_{src}$, $Avg-port_{dest}$, $num-ptsgpd_{DIP}$, and $num-ptsgpd_{SIP}$.

## 3.3. Feature Fusion

Once the features and the attributes are selected from the cloud model, then the process of feature fusion can be performed by grouping the features using the FCM model. The process of combining and optimizing different features is called feature fusion. It removes the redundant information and keeps more effective features with discriminant information. This process makes the detection mechanism increase the rate of intrusion detection.

### 3.3.1. Feature Grouping by FCM

This section describes the process of feature grouping using the FCM model Liu, Wu, and Sun (2022). Let us consider $f$ the selected features such that $F = \{f_j\};\ j = \{1, 2, 3, ..., n\}$, where $n$ indicates the total number of features. The objective function that is minimized by FCM is expressed as follows:

$$\beta_i = \sum_{j=1}^{n} \sum_{l=1}^{m} \lambda_{jl}^{m} g^2 (f_l, S_j), \tag{2}$$

where $\lambda_{jl}$ represents the membership function matrix, $g$ specifies Euclidean distance among $f_l$ and $S_j$, and $k$ specifies the degree of fuzziness, and its value must be greater than 1 $(k > 1)$. The heart of FCM is the membership function $\lambda_{jl}$, which is given as follows:

$$\lambda_{jl} = \frac{1}{\sum_{v=1}^{z} \left( \frac{g_{jl}}{g_{vl}} \right)^{2/(k-1)}}. \tag{3}$$

The cluster center is given as follows:

$$S_j = \frac{\sum_{j=1}^{n} \lambda_{jl}^{k} f_l}{\sum_{l=1}^{m} \lambda_{jl}^{k}}. \tag{4}$$

Due to the simplicity of the distance metric, the Euclidean distance is used.

### 3.3.2. Feature Fusion of Every Group

After the VM migration process using the ChicWhale algorithm has been successfully completed, the attribute gathering step starts. Let us assume $N$ the number of features in the group, and thereby the feature fusion is represented as follows:

$$F^{new} = \sum_{i=1}^{|N|} \frac{1}{\mu_i} \beta_i \tag{5}$$

where, $\mu_i$ represents the constant, $F^{new}$ is the fused feature, and $N$ specifies the total number of features in the group.

### Intrusion detection by Deep Recurrent Neural Network

Deep RNN is used to carry out the intrusion detection process following the feature fusion. Here, the SFO Shadravan, Naji, and Bardsiri (2019) and DE algorithm Borkar and Mahajan are combined to form the SFDO algorithm, which is used to train Deep RNN.

### *Structure of Deep RNN*

In order to record network data using the directed connection of the hidden layers, Deep RNN [31] is an artificial neural network with a number of hidden layers. Let us consider the input state as $X_d$ with the time stamp $d$, $Z_d$ as the hidden layer state at time $d$, and $q$ as the nonlinear functions, like ReLU and tanh. The input of the processing unit is recorded in internal states using the activation function. The input state that processes the input for performing the detection process is the fused feature$F_{new}$. The output result at time $d$ is specified as $\delta_d$.

The hidden layer at the time $d$ is given as follows:

$$Z_d = q\left(X_d P^{(ZX)} + X_{d-1} P^{(ZZ)}\right). \tag{6}$$

Finally, the output derived from the classifier is represented as follows:

$$\delta_d = q\left(Z_d P^{(\delta Z)}\right). \tag{7}$$

The training of Deep RNN uses the SFDO algorithm to find the best location of parameter optimization until the convergence condition is satisfied.

### *Training Procedure of SFDO Algorithm*

The developed SFDO technique is used to perform the deep learning classifier's training procedure. The proposed SFDO algorithm is devised by integrating SFO Shadravan, Naji, and Bardsiri (2019) and DE algorithm Borkar and Mahajan (2017). The steps of the proposed SFDO algorithm are explained as follows:

i) ***Initialization:*** Initialization is the first step in the optimization of sailfish population in the $p$ dimensional space. In the search area, sailfish are viewed as potential solutions.. The sailfish can search in the dimensional space using the variable position vectors. The $h^{th}$ member sailfish in $y^{th}$ searching at $r^{th}$ iteration is given as, $J_{hy} \in W(h = 1, 2, \ldots, u)$. Here, $u$ indicates the total number of sailfish. Moreover, the parameters $L$ is considered as 4, and value of $\sigma$ is set as 0.001, respectively.

ii) ***Compute fitness measure:*** Using the optimization algorithm, the fitness measure is computed to determine the best intrusion detection process solution. The function used to compute the fitness is denoted as follows:

$$\mathrm{M} = \frac{1}{\kappa}\sum_{\ell=1}^{\kappa}[\delta_\ell - \mathrm{H}_\ell], \tag{8}$$

where M denotes fitness, $\kappa$ denotes total number of samples, and $\mathrm{H}_\ell$ represents target output.

iii) ***Compute$\varpi$ :*** The coefficient $\varpi$ at $r^{th}$ iteration is represented as follows:

$$\varpi_r = 2 \times rand(0,1) \times V - V, \tag{9}$$

where $rand(0,1)$ represents the random number that ranges in the interval of $[0,1]$, and $V$ describes prey density, which shows how many prey there are during each cycle. The parameter $V$ is more significant to update the location of sailfish, as the prey gets reduced while performing hunting strategy of sailfish such that $V$ is computed as follows:

$$V = 1 - \left(\frac{\rho_J}{\rho_J - \rho_{sar}}\right). \tag{10}$$

Here, $\rho_J$ represents number of sailfish at each cycle, and $\rho_{sar}$ indicates the number of sardines at each cycle.

iv) *Update the location of sailfish:* The sailfish attacks prey school and promotes success in the hunting strategy. Based on the hunters' location, the sailfish updates its position using the herding behavior. The attack alternation model of sailfish is defined using group hunting. It is required to search a wide area in search space to compute the best solution. The sailfish attack in all directions within the shrinking circle, and this behavior is modeled as follows:

$$J_{hy}^{r+1} = J^* - \eta\left(rand(0,1)\left(\frac{J^* + J_{injured}}{2}\right) - J_{hy}^r\right), \tag{11}$$

where, $J_{hy}^r$ represents the location of sailfish at the $r^{th}$ iteration, $J_{injured}$ represents injured sardine, and $rand(0,1)$ represents the random number that ranges in the interval of $[0,1]$. The standard equation of DE algorithm is given as follows:

$$J_{hy}^{r+1} = J_{hy}^r + s_{hy}^{r+1}, \tag{12}$$

$$J_{hy}^{r+1} = J_{hy}^r + s_{hy}^r + \gamma_{1t}(M_t - J_{hy}^r) + \gamma_{2t}(K - J_{hy}^r), \tag{13}$$

$$J_{hy}^{r+1} = J_{hy}^r(1 - \gamma_{1t} - \gamma_{2t}) + s_{hy}^r + \gamma_{1t}M_t + \gamma_{2t}K, \tag{14}$$

$$J_{hy}^r = \frac{J_{hy}^{r+1} - s_{hy}^r - \gamma_{1t}M_t - \gamma_{2t}K}{(1 - \gamma_{1t} - \gamma_{2t})}. \tag{15}$$

By substituting Equation (15) into Equation (11) is represented as follows:

$$J_{hy}^{r+1} = J^* - \eta\left(rand(0,1)\left(\frac{J^* + J_{injured}}{2}\right) - \left(\frac{J_{hy}^{r+1} - s_{hy}^r - \gamma_{1t}M_t - \gamma_{2t}K}{(1 - \gamma_{1t} - \gamma_{2t})}\right)\right), \tag{16}$$

$$J_{hy}^{r+1} = \frac{1-\gamma_{1t}-\gamma_{2t}}{1-\gamma_{1t}-\gamma_{2t}-\eta} \left\{ J^* - \eta \left( rand(0,1) \left( \frac{J^* + J_{injured}}{2} \right) + \left( \frac{+s_{hy}^r + \gamma_{1t} M_t + \gamma_{2t} K}{(1-\gamma_{1t}-\gamma_{2t})} \right) \right) \right\}, \tag{17}$$

where

v) *Computex :* At each cycle, each sardine updates its position based on the sailfish's location and attack strength. At each cycle the attack power used by the sailfish is given as,

$$x = \chi \times (1 - (2 \times itr \times \sigma)), \tag{18}$$

where $x$ denotes attack power, and $\chi$ and $\sigma$ specifies coefficients, respectively.

vi) *Compute G and Y* : During the hunting process, the sailfish reduce the attack power over time. The parameter $G$ and $Y$ are used by sardines to update the position, and the parameters are given as,

$$G = \rho_{sar} \times x, \tag{19}$$

$$Y = p_r \times x, \tag{20}$$

where $p_r$ denotes the number of variables at $r^{th}$ iteration, and $\rho_{sar}$ denotes number of sardines.

vii) *Update location of sardine*: At iteration $r$, the position of the sardine is computed as follows:

$$H^{r+1} = rand\left(J_{hy}^r - H^r + x\right), \tag{21}$$

where $H^r$ denotes the location of sardine at the previous iteration and $x$ denotes attack power.

viii) *Replace sailfish with injured sardine*: When the injured sardine emerges to be more fit than the sailfish, others can swiftly capture it if they update their location using the equation below,

$$J_{hy}^r = H^r; if\, f(H) < f(J), \tag{22}$$

where $J_{hy}^r$ denotes the sailfish position, and $H$ denotes the sardine location

ix) *Termination:* Up until the maximum numbers of iterations are used or the best solution is found, the above mentioned stages are repeated.

Algorithm 1 portrays the pseudo code of the developed SFDO-based Deep RNN.

```
Sl. No    Pseudo code of the proposed SFDO-based Deep RNN
1         Input : $J$ , $L$, and $\sigma$
2         Output: $J_{hy}^{r+1}$
3         Establish the population
4         Set $A = 4$ and $\sigma = 0.001$
5         Compute M
6         Find the best sardines and sailfish.
7         As long as the termination condition is not met
8         for each $J$
9                   Compute $\varpi$ using Eq. (9)
10                  Using Eq, update the sailfish's location (20)
11        end for
12              Compute $x$using Eq. (21)
13        if ($x < 0.5$)
14              Compute $G$using Eq. (22)
15              Computer $Y$using Eq. (23)
16              Select the set of sardine based on $G$ and $Y$
17              Sardine's location has been updated using Eq. (24)
18        else
19              Eq. (24) should be used to update every sardine's location
20        end if
21           Compute M for all sardine
22        if there exists better solution
23              Use Eq. (25) to swap out injured sardines for sailfish
24              Remove hunted sardine
25              Best sardine and sailfish updates
26        end if
27        end while
28        Return best sailfish
```

## Results and Discussion

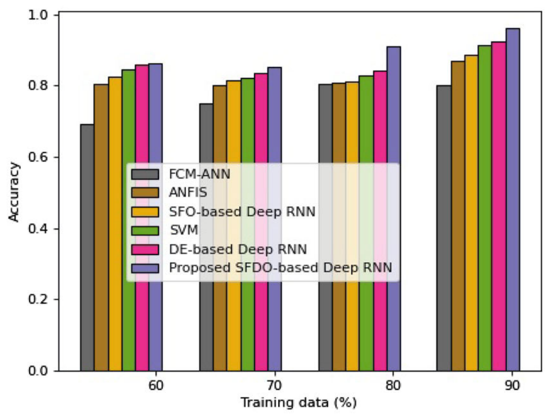
This article discusses the outcomes produced by the suggested SFDO-based Deep RNN when the group size was changed. Using the dataset listed in https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/bot_iot.php (Dataset: BoT-IoT dataset 2020) and the JAVA tool, the proposed model is implemented.

### *Comparative Methods*

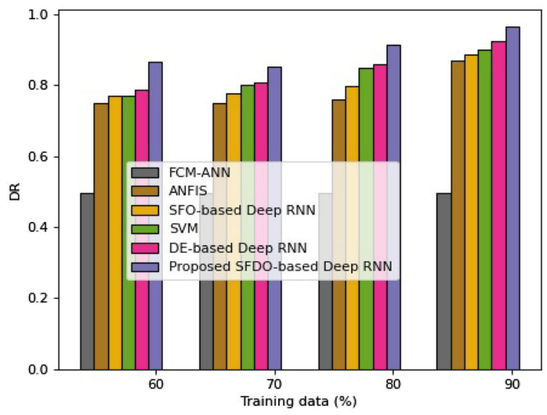
The performance of the proposed model is analyzed using the conventional methods, like FCM-ANN (Pandeeswari and Kumar 2016), ANFIS model (Ganeshkumar and Pandeeswari 2016), SVM (Fu, Liu, and Pannu 2012), SFO-based Deep RNN, and DE-based Deep RNN.

### *Comparative Analysis*

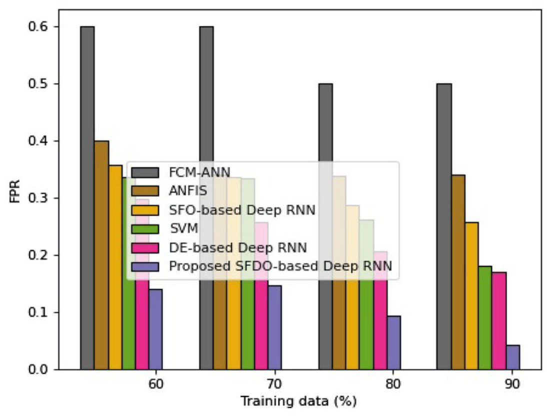
This section explains the comparative analysis of the suggested model when the number of groups is changed to 3, 4, and 5.



(a)



(b)



(c)

**Figure 2.** Analysis with group size = 3, (a) accuracy, (b) DR, and (c) FPR.

## *Analysis with Group Size = 3*

Figure 2 displays the analysis of the proposed model using a three-person group size. Figure 2a shows accuracy analysis results. When 60% of the training data are used, the accuracy of FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and proposed SFDO-based Deep RNN is 0.6927, 0.8048, 0.8462, 0.826, 0.857, and 0.8634, respectively. When the training data are taken as 70%, the accuracy of the FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and proposed SFDO-based Deep RNN is 0.7500, 0.8029, 0.8214, 0.815, 0.836, and 0.8542, respectively. When the training data are 80% complete, the accuracy of the FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and the proposed SFDO-based Deep RNN is 0.8034, 0.8093, 0.8276, 0.8107, 0.840, and 0.9107, respectively.

In Figure 2b, the analysis of DR is displayed. When training data are taken into account, DR values for FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and proposed SFDO-based Deep RNN are respectively 0.4976, 0.7503, 0.7692, 0.7689, 0.786, and 0.8663. When training data are taken into consideration, the DR of the FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and proposed SFDO-based Deep RNN are, in order, 0.4957, 0.7500, 0.80, 0.775, 0.807, and 0.8539. The DR of the FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and proposed SFDO-based Deep RNN are, respectively, 0.4960, 0.7586, 0.8498, 0.797, 0.857, and 0.9146 when training data are set to 80%. The outcomes of the FPR analysis are displayed in Figure 2c. When trained on 60% of the data, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and the proposed SFDO-based Deep RNN have FPRs of 0.6, 0.3996, 0.3368, 0.357, 0.298, and 0.1403, respectively. When training data are taken into consideration, the FPR of the FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and proposed SFDO-based Deep RNN are, respectively, 0.6, 0.3395, 0.3334, 0.336, 0.257, and 0.1466. When training data are taken into account, the FPR of the FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and proposed SFDO-based Deep RNN are, respectively, 0.5, 0.3391, 0.2611, 0.206, and 0.0941.

## *Analysis with Group Size = 4*

Figure 3 displays the analysis of the proposed model using a four-person group size. Figure 3a displays an analysis of accuracy. When training data are taken into account, the accuracy values for FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and proposed SFDO-based Deep RNN are, respectively, 0.7857, 0.8045, 0.8090, 0.806, 0.877, and 0.9331. The DR analysis is shown in Figure 3b. When training data are considered, the DR of FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and the proposed SFDO-based Deep RNN are, respectively, 0.4974, 0.7502, 0.7857, 0.769, 0.808, and 0.8435. The FPR analysis is illustrated in Figure 3c. When training data are assumed to be 90%, the FPR of the FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and proposed SFDO-based Deep RNN are, in order, 0.3405, 0.3333, 0.1823, 0.216, 0.108, and 0.0543.

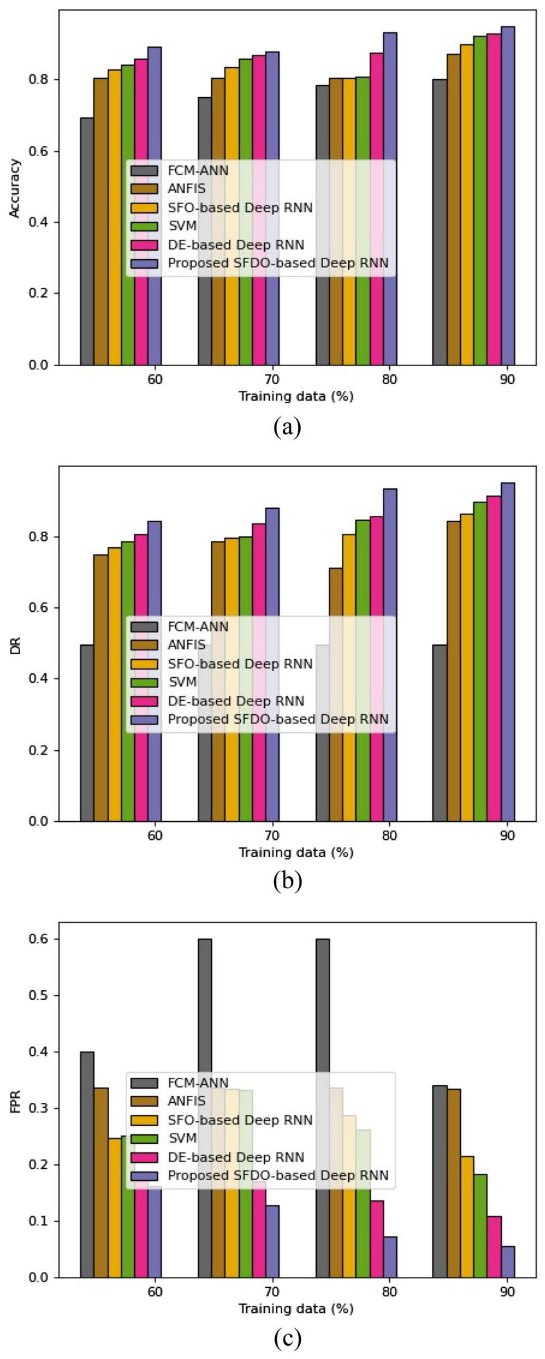
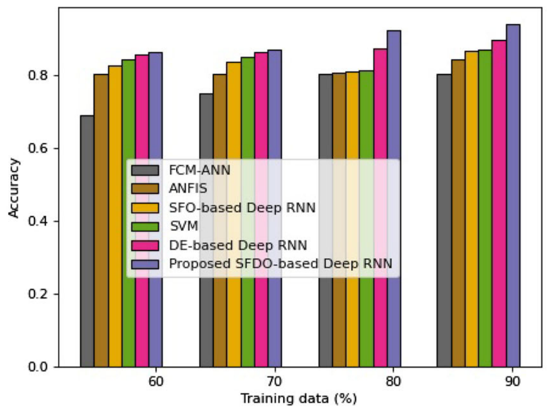


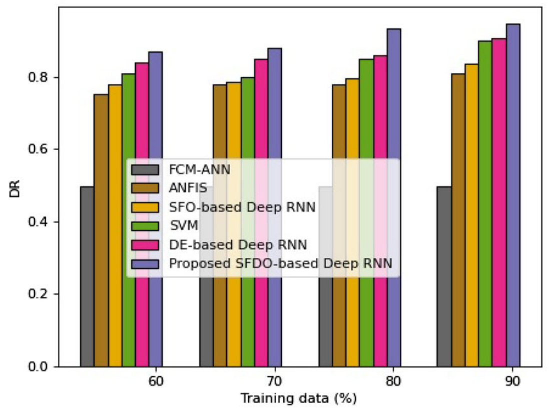
**Figure 3.** Analysis with group size = 4, (a) accuracy, (b) DR, and (c) FPR.

## *Analysis with Group Size = 5*

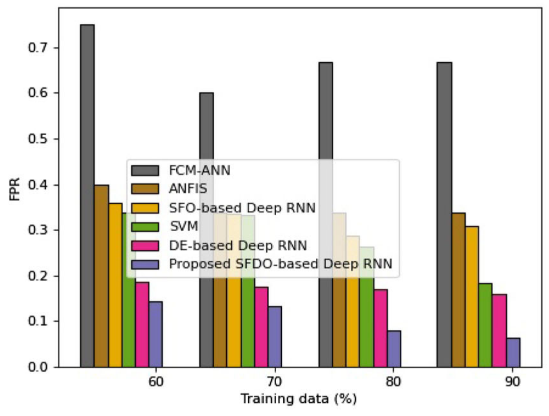
The analysis of the proposed model, which takes into account a group size of five, which is shown in Figure 4. In Figure 4a, accuracy analysis is depicted. When 60% of the training data are used, the accuracy of



(a)



(b)



(c)

**Figure 4.** Analysis with group size = 5, (a) accuracy, (b) DR, and (c) FPR.

FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and proposed SFDO-based Deep RNN is 0.6926, 0.8047, 0.8462, 0.826, 0.859, and 0.8635, respectively. When the training data are taken as 70%,

Table 1. Comparative discussion.

| Metrics/Methods | | FCM-ANN | ANFIS | SVM | SFO-based Deep RNN | DE-based Deep RNN | Proposed SFDO-based Deep RNN |
|---|---|---|---|---|---|---|---|
| Group size = 3 | *Accuracy* | 0.8029 | 0.8713 | 0.9130 | 0.886 | 0.926 | 0.9614 |
| | *DR* | 0.4954 | 0.8696 | 0.9003 | 0.8865 | 0.925 | 0.9648 |
| | *FPR* | 0.5 | 0.3399 | 0.1813 | 0.2578 | 0.1689 | 0.0429 |
| Group size = 4 | *Accuracy* | 0.8013 | 0.8706 | 0.9231 | 0.897 | 0.928 | 0.9492 |
| | *DR* | 0.4950 | 0.8462 | 0.8997 | 0.865 | 0.915 | 0.9522 |
| | *FPR* | 0.3405 | 0.3333 | 0.1823 | 0.215 | 0.107 | 0.0543 |
| Group size = 5 | *Accuracy* | 0.8044 | 0.8462 | 0.8704 | 0.868 | 0.897 | 0.9412 |
| | *DR* | 0.4975 | 0.8077 | 0.8995 | 0.836 | 0.9076 | 0.9459 |
| | *FPR* | 0.6667 | 0.3369 | 0.1826 | 0.3087 | 0.1578 | 0.0645 |

the accuracy of the FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and proposed SFDO-based Deep RNN is 0.7501, 0.8038, 0.8519, 0.836, 0.865, and 0.8731, respectively. When trained with 80% of the available data, FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and the proposed SFDO-based Deep RNN all have accuracy values of 0.8042, 0.8085, 0.8148, 0.8107, 0.875, and 0.9267.

The DR analysis is displayed in Figure 4b. When training data are taken into account, the DR of FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and the proposed SFDO-based Deep RNN are 0.4973, 0.7503, 0.8077, 0.7799, 0.8378, and 0.8684. When training data are taken into consideration, the DR of the FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and the proposed SFDO-based Deep RNN are, in order, 0.4968, 0.7778, 0.8001, 0.786, 0.847, and 0.8777. When training data are assumed to be 80%, the DR of the FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and the proposed SFDO-based Deep RNN are, in order, 0.4971, 0.7778, 0.8492, 0.796, 0.857, and 0.9314.

In Figure 4, the FPR analysis is displayed (c). When training data are taken into consideration, the FPR of FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and the proposed SFDO-based Deep RNN are, respectively, 0.7500, 0.3996, 0.3372, 0.357, 0.186, and 0.1422. When training data are taken into account, the FPR of the proposed SFDO-based Deep RNN is 0.60, 0.3379, 0.3332, 0.335, 0.175, and 0.1324. When training data are taken into account, the FPR of the proposed SFDO-based Deep RNN, ANFIS, SVM, and FCM-ANN are 0.6667, 0.3376, 0.2621, 0.287, 0.168, and 0.0789, respectively.

## *Comparative Discussion*

A comparison of the suggested approach is shown in Table 1. Based on the most effective outcomes of the comparative methods, the analysis presented here. The accuracy values for the FCM-ANN, ANFIS, SVM,

**Table 2.** Analysis with other state-of-art methods.

| Metrics/Methods | HIDS (Alharbi et al. 2021) | Deep learning (Alharbi et al. 2021) | Deep NN + GA (Alharbi et al. 2021) | Include the proposed method |
|---|---|---|---|---|
| *Accuracy* | 0.8564 | 0.9240 | 0.9613 | 0.9614 |

SFO-based Deep RNN, DE-based Deep RNN, and the proposed SFDO-based Deep RNN are 0.8029, 0.8713, 0.9130, 0.886, 0.926, and 0.9614, respectively, for 90% of the training data. The proposed SFDO-based Deep RNN, ANFIS, SVM, SFO-based Deep RNN, and DE-based Deep RNN have DRs of 0.4954, 0.8696, 0.9003, 0.8865, 0.925, and 0.9648 for 90% of the training data, respectively. When the group size is assumed to be three, the FPR for 90% of the training data for FCM-ANN, ANFIS, SVM, SFO-based Deep RNN, DE-based Deep RNN, and the proposed SFDO-based Deep RNN are, respectively, 0.5, 0.3399, 0.1813, 0.2578, 0.1689, and 0.0429. This method works better when there are three participants.

### *Analysis with Other State-of-Art Methods*

Table 2 shows comparisons with other cutting-edge techniques. The table shows that the accuracy of the proposed method is 0.9614, while the accuracy of HIDS, deep learning, and deep NN + GA are, respectively, 0.8564, 0.9240, and 0.9613.

## Conclusion

The proposed SFDO-based Deep RNN in this study models an optimal and effective intrusion detection strategy to find abnormalities in the cloud architecture. First, the ChicWhale algorithm is used to advance cloud data management and VM migration. The cloud model's attribute features are gathered, and the gathered attributes are used to carry out a feature grouping process using FCM. After the feature grouping process is finished, a Deep RNN classifier is used to fuze the features to create an intrusion detection process. As a result, the Deep RNN classifier is trained using the SFDO algorithm, a proposed optimization algorithm. The features of both algorithms are turned on to boost performance and detection accuracy in the cloud model. The proposed SFDO-based Deep RNN performed better in terms of accuracy, DR, and FPR, with scores of 0.9614, 0.9648, and 0.0429, respectively. The drawbacks of the proposed method are computation is slow due to recurrent nature and difficult to process longer sequences. Also, the creation of a different optimization technique to efficiently train the classifier would be the research's future aspect.

## Nomenclature

| Abbreviations | Definitions |
|---|---|
| SFDO-based Deep RNN | Sail Fish Dolphin optimization-based deep recurrent neural network |
| SFO | Sail fish optimizer |
| DE | Dolphin echolocation |
| VM | virtual machine |
| FCM | Fuzzy C-means clustering |
| CCT | cloud computing technology |
| MLO | mixture localization-based outliers |
| LOF | local outlier factor |
| GMM | Gaussian Mixture Model |
| FCM + ANN | Fuzzy c-means model and artificial neural network |
| ANFIS | Adaptive Neuro-Fuzzy inference system |
| SVM | support vector machine |
| MIPS | million instructions per second |
| CSO | chicken Swarm optimization |
| WOA | Whale optimization algorithm |
| FPR | false-positive rate |
| DR | detection rate |
| ESCNN | evolutionary sparse convolution neural network |
| LGBA-NN | Local–Global Best Bat Algorithm for Neural Networks |
| IDS | intrusion detection system |
| HIDS | host-based intrusion detection system |
| Deep NN + GA | Deep Neural Network + Genetic Algorithm |

## Data Availability Statement

The data that support the findings of this study are openly available in BoT-IoT dataset at https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/bot_iot.php.

## References

Abdalla, M, and A. M. Ahmed. 2020. Optimization driven mapreduce framework for indexing and retrieval of big data. *Ksii Transactions on Internet and Information SYSTEMS* 14:1886–908.

Agrawal, B., T. Wiktorski, and C. Rong. 2016. Adaptive anomaly detection in cloud using robust and scalable principal component analysis. In IEEE 15th International Symposium on Parallel and Distributed Computing (ISPDC), 100–6.

Agrawal, B., T. Wiktorski, and C. Rong. 2017. Adaptive real-time anomaly detection in cloud infrastructures. *Concurrency and Computation: Practice and Experience* 29 (24): e4193. doi:10.1002/cpe.4193.

Alabdulatif, A., H. Kumarage, I. Khalil, and X. Yi. 2017. Privacy-preserving anomaly detection in cloud with lightweight homomorphic encryption. *Journal of Computer and System Sciences* 90:28–45. doi:10.1016/j.jcss.2017.03.001.

Alharbi, A., W. Alosaimi, H. Alyami, H. T. Rauf, and R. Damaševičius. 2021. Botnet attack detection using local global best bat algorithm for industrial internet of things. *Electronics* 10 (11):1341. doi:10.3390/electronics10111341.

Ali, M. H., M. M. Jaber, S. K. Abd, A. Rehman, M. J. Awan, R. Damaševičius, and S. A. Bahaj. 2022. Threat analysis and distributed denial of service (DDoS) attack recognition in the Internet of Things (IoT). *Electronics* 11 (3):494., doi:10.3390/electronics11030494.

AlKadi, O., N. Moustafa, B. Turnbull, and K.-K. R. Choo. 2019. Mixture localization-based outliers models for securing data migration in cloud centers. *IEEE Access*. 7:114607–18. doi:10.1109/ACCESS.2019.2935142.

Alzaqebah, A., I. Aljarah, O. Al-Kadi, and R. Damaševičius. 2022. A modified grey wolf optimization algorithm for an intrusion detection system. *Mathematics* 10 (6):999. doi: 10.3390/math10060999.

Borkar, G. M, and A. R. Mahajan. 2017. A secure and trust based on-demand multipath routing scheme for self-organized mobile ad-hoc networks. *Wireless Networks* 23 (8): 2455–72. doi:10.1007/s11276-016-1287-y.

Calheiros, R. N., K. Ramamohanarao, R. Buyya, C. Leckie, and S. Versteeg. 2017. On the effectiveness of isolation-based anomaly detection in cloud data centers. *Concurrency and Computation: Practice and Experience* 29 (18):e4169. doi:10.1002/cpe.4169.

Chen, G., G. Sun, Y. Xu, and B. Long. 2009. Integrated research of parallel computing: Status and future. *Science Bulletin* 54 (11):1845–53. doi:10.1007/s11434-009-0261-9.

Chen, M., Y. Qian, J. Chen, K. Hwang, S. Mao, and L. Hu. 2016. Privacy protection and intrusion avoidance for cloudlet-based medical data sharing. *IEEE Transactions on Cloud Computing* 8 (4):1274–1283.

Dataset: BoT-IoT dataset. 2020. https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cyber-security/ADFA-NB15-Datasets/bot_iot.php

Dhage, S. N, and B. B. Meshram. 2012. Intrusion detection system in cloud computing environment. *International Journal of Cloud Computing* 1 (2/3):261–82. doi:10.1504/IJCC.2012.046711.

Farshchi, M., J. G. Schneider, I. Weber, and J. Grundy. 2018. Metric selection and anomaly detection for cloud operations using log and metric correlation analysis. *Journal of Systems and Software* 137:531–49. doi:10.1016/j.jss.2017.03.012.

Fu, S., J. Liu, and H. Pannu. 2012. A hybrid anomaly detection framework in cloud computing using one-class and two-class support vector machines. In *International Conference on Advanced Data Mining and Applications*, Springer, Berlin, Heidelberg, 726–38.

Ganeshkumar, P, and N. Pandeeswari. 2016. Adaptive neuro-fuzzy-based anomaly detection system in cloud. *International Journal of Fuzzy Systems* 18 (3):367–78. doi:10.1007/s40815-015-0080-x.

Garg, S., K. Kaur, N. Kumar, S. Batra, and M. S. Obaidat. 2018. HyClass: Hybrid classification model for anomaly detection in cloud environment. In IEEE International Conference on Communications (ICC), 1–7.

Hui, Y. 2018. A virtual machine anomaly detection system for cloud computing infrastructure. *The Journal of Supercomputing* 74 (11):6126–34. doi:10.1007/s11227-018-2518-z.

Karuppusamy, L., J. Ravi, M. Dabbu, and S. Lakshmanan. 2022. Chronological salp swarm algorithm based deep belief network for intrusion detection in cloud using fuzzy entropy. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields* 35 (1):1–19. doi:10.1002/jnm.2948.

Koroniotis, N., N. Moustafa, E. Sitnikova, and B. Turnbull. 2019. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems* 100:779–96. doi:10.1016/j.future.2019.05.041.

Lim, D., Y. S. Ong, Y. Jin, B. Sendhoff, and B. S. Lee. 2007. Efficient hierarchical parallel genetic algorithms using grid computing. *Future Generation Computer Systems* 23 (4): 658–70. doi:10.1016/j.future.2006.10.008.

Liu, D., Z. Wu, and S. Sun. 2022. Study on Subway passenger flow prediction based on deep recurrent neural network. *Multimedia Tools and Applications* 81 (14):18979–92. doi:10.1007/s11042-020-09088-x.

Mahmood, Z., C. Agrawal, S. S. Hasan, and S. Zenab. 2012. Intrusion detection in cloud computing environment using neural network. *International Journal of Research in Computer Engineering and Electronics* 1 (1):1–4.

Mekhmoukh, A, and K. Mokrani. 2015. Improved Fuzzy C-Means based Particle Swarm Optimization (PSO) initialization and outlier rejection with level set methods for MR brain image segmentation. *Computer Methods and Programs in Biomedicine* 122 (2): 266–81. doi:10.1016/j.cmpb.2015.08.001.

Meng, X., Y. Liu, X. Gao, and H. Zhang. 2014. A new bio-inspired algorithm: chicken swarm optimization. In *International conference in swarm intelligence*, Springer, Cham, 86–94.

Mirjalili, S, and A. Lewis. 2016. The whale optimization algorithm. *Advances in Engineering Software* 95:51–67. doi:10.1016/j.advengsoft.2016.01.008.

Mohsin, M., H. Li, and H. B. Abdalla. 2020. Optimization driven Adam-Cuckoo search-based deep belief network classifier for data classification. *IEEE Access*. 8:105542–60. doi: 10.1109/ACCESS.2020.2999865.

Pandeeswari, N, and G. Kumar. 2016. Anomaly detection system in cloud environment using fuzzy clustering based ANN. *Mobile Networks and Applications* 21 (3):494–505. doi:10.1007/s11036-015-0644-x.

Sabahi, F. 2011. Cloud computing security threats and responses. In IEEE 3rd International Conference on Communication Software and Networks, Xi'an, China, 245–249.

Shadravan, S., H. R. Naji, and V. K. Bardsiri. 2019. The Sailfish Optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. *Engineering Applications of Artificial Intelligence* 80:20–34. doi:10.1016/j.engappai.2019.01.001.

Soldatos, J., M. Serrano, and M. Hauswirth. 2012. Convergence of utility computing with the internet-of-things. In IEEE Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Palermo, Italy, 874–849.

Vieira, K., A. Schulter, C. Westphall, and C. Westphall. 2010. Intrusion detection for grid and cloud computing. *IT Professional* 12( (4):38–43. doi:10.1109/MITP.2009.89.

Ye, X., X. Chen, H. Wang, X. Zeng, G. Shao, X, and Yin, C. X. 2016. An anomalous behavior detection model in cloud computing. *Tsinghua Science and Technology* 21 (3):322–32. doi:10.1109/TST.2016.7488743.

Zarrabi, A, and A. Zarrabi. 2012. Internet intrusion detection system service in a cloud. *International Journal of Computer Science Issues (IJCSI)* 9 (5):308.

Zhang, Z., J. Wen, J. Zhang, X. Cai, and L. Xie. 2020. A many objective-based feature selection model for anomaly detection in cloud environment. *IEEE Access*. 8:60218–31. doi: 10.1109/ACCESS.2020.2981373.

## Appendix

| Attributes | Description |
|---|---|
| $ID_{row}$ | Row identifier |
| $ST_{rcd}$ | Start time of record |
| $fg$ | Flow state flags observed in the transaction |
| $fg-num$ | Numerical representation of the feature flags |
| $TR_{proto}$ | Texture representation of the transaction protocols find in the network flow |
| $num-TR$ | Feature protocol that are represented in the numerical form |
| $IP_{src}$ | IP address of source |
| $port_{src}$ | Port number of source |
| $IP_{dest}$ | IP address of source |
| $port_{dest}$ | Port number of destination |
| $pks$ | Total number of packets present in the transactions |
| $tot-bytes$ | Total number of bytes in the transaction. |
| $Trans-state$ | State of transaction |
| $num-Trans$ | Feature state that are represented in numerical form |
| $Last_{rcd}$ | Last time of record |
| $Ar_{seq}$ | Sequence number of Argus |
| $Tot_{rcd}$ | Total duration of record |
| $Agg_{avg}$ | Average duration of the aggregated records |
| $Agg_{dev}$ | Standard deviation of the aggregated records |
| $Agg_{total}$ | Total duration of the aggregated records |
| $Agg_{\min}$ | Minimum duration of the aggregated records |
| $Agg_{\max}$ | Maximum duration of the aggregated records, |
| $pkt_{std}$ | Number of packets that is transferred from source to destination |
| $pkt_{dts}$ | Number of packets transmitted from the destination to source |
| $byte_{std}$ | Number of bytes between source and destination |
| $byte_{dts}$ | Total number of bytes between destination and source |
| $pkt_{total}$ | Total count of packets transmitted in the transaction per second |
| $SD-pkt_{per\sec}$ | Total count of packets transmitted from source to destination per second |
| $DS-pkt_{per\sec}$ | The total count of data packets transmitted from destination to source per second |
| $Tot-bytes_{SIP}$ | Total count of bytes available per source IP |
| $Tot-bytes_{DIP}$ | Total count of bytes available per destination IP |
| $Tot-pkts_{SIP}$ | Total count of packets available per source IP |
| $Tot-pkts_{DIP}$ | Total number of packets available per destination IP |
| $Tot-proto$ | Total count of packets in each protocol |
| $Tot-port_{dest}$ | Total count of packets available in each port of destination |
| $Avg_{SIP}$ | Average rate of the protocol for each source IP that is measured using the number of data packets transmitted at each instance of time |
| $Avg_{DIP}$ | Average rate of the protocol for each destination IP |
| $num-inbnd_{SIP}$ | Number of inbound connections for each source IP |
| $num-inbnd_{DIP}$ | Number of inbound connections for each destination IP |
| $Avg-port_{src}$ | Average rate per protocol with respect to source port number |
| $Avg-port_{dest}$ | Average rate per protocol with respect to destination port number |
| $num-ptsgpd_{DIP}$ | Total count of packets connected by the state of flows and protocols at each destination IP |
| $num-ptsgpd_{SIP}$ | Total count of packets connected by the state of flows and protocols at each IP of source |