Water Moth Search Algorithm-based Deep Training for Intrusion Detection in IoT

Rekha P. M.[1,*] and Nagamani H. Shahapure[2], Punitha M.[2] and Sudha P. R.[2]

[1]*Department of Information Science and Engineering, JSS Academy of Technical Education, Dr. Vishnuvardhana Road, Bengalru-560060, Karnataka, India*
[2]*Department of Information Science, JSS Academy of Technical Education, Bangalore, Karnataka, India*
E-mail: rekhapm12@gmail.com
[*]*Corresponding Author*

Received 09 January 2021; Accepted 15 July 2021;
Publication 21 September 2021

Abstract

The economic growth and information technology leads to the development of Internet of Things (IoT) industry and has become the emerging field of research. Several intrusion detection techniques are introduced but the detection of intrusion and malicious activities poses a challenging task. This paper devises a novel method, namely the Water Moth Search algorithm (WMSA) algorithm, for training Deep Recurrent Neural Network (Deep RNN) to detect malicious network activities. The WMSA algorithm is newly devised by combining Water Wave optimization (WWO) and the Moth Search Optimization (MSO). The pre-processing is employed for the removal of redundant data. Then, the feature selection is devised using the Wrapper approach, then using the selected features; the Deep RNN classifier effectively detects the intrusion using the selected features. The proposed WMSA-based Deep RNN showed improved results with maximal accuracy, specificity, and sensitivity of 0.96, 0.973 and 0.960.

Journal of Web Engineering, Vol. 20_6, 1781–1812.
doi: 10.13052/jwe1540-9589.2064

Keywords: Internet of things, intrusion detection, water wave optimization, deep recurrent neural network, moth search optimization.

Nomenclature

Abbreviation	Definition
IoT	Internet of Things
WWO	Water wave optimization
MSO	Moth Search Optimization
WMSA	Water Moth Search algorithm
IDS	Intrusion detection system
NIDS	Network-based intrusion detection system
AI	Artificial Intelligence
RNN	Recurrent Neural Network
N-BaIoT	Network-Based Detection of IoT Botnet Attacks
BRIoT	Behavior rule specification based IoT
COLIDE	Collaborative intrusion detection framework
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
PCA	Principal component analysis
VM	Virtual Machines
NIC	Network Interface Cards
WAN	Wide area network
LAN	Local area network
IP	Internet Protocol
DDoS	Distributed denial of service
ANN	Artificial Neural Network
GA+DBN	Genetic Algorithm and Deep Belief Network

1 Introduction

IoT is one of the third industrial revolution [1, 9], which is interconnected through the Internet of the computing devices embedded in objects that enables for sending and receiving data [1, 10]. IoT is utilized in various fields, like domestic, healthcare, energy distribution, finances, tourism, transportation, and the smart-cities [1, 11]. The rapid growth of IoT technology is utilized for household appliances that improve the life quality [2, 12]. In addition, the IoT is distributed nature, and openness, which leads to the attacks [13–15]. Additionally, several IoT nodes store, process, and collect information privately, but they are apparent to malicious

attackers [16]. Hence, maintaining privacy of the IoT is the priority of successfully deploying the IoT environment [2, 17]. The application of IoT involves people's livelihood, industry, military, and commerce, where the authentication is more essential. If information security issue exists, the losses and damage are very dangerous. The provision of authentication and integrity is still challenging. Hence, the privacy preserving in the IoT is very essential [8, 13].

Intrusion detection is one of the significant steps that ensure IoT security networks. Intrusion detection having various security mechanisms to manage the security intrusions, which are detected using four layers of IoT architecture. The Network Layer serves as the backbone to connect various IoT devices, like Network Intrusion Detection Systems (NIDS) [2, 18]. As the indispensable technology in the authentication of the network, intrusion detection monitors abnormal behavioral dynamically the model to check whether the events are susceptible to attack [3, 14, 19]. In addition, the intrusion detection approaches are categorized into two types. Generally, anomaly better accesses the systems automatically and detects the incoming intrusions. It recognizes the attacks, but still, false alarms may arise. An IDS is the best technique to identify the attackers [7, 20] and detects the policy violations or the malicious activities of system actives or the network traffics [7, 15, 21]. Intrusion Detection is a third-part software or stand-by device that inquires several changes; hence, it is appropriate for inherited systems or resource constraints to protect network security.

NIDS has been scrutinized for achieving the conventional devices securely since the 1980s [11, 24]. Conventional NIDS are lacking for the IoT systems because of limited power, heterogeneity, constrained resources, and connectivity [1, 22, 23]. Hence, the IDSs of IoT systems is relevant. The popular strategies deployed between the IoT systems are the NIDSs or IDSs for the connected smart Things. Several IDS approaches are performed using cluster analysis [2, 25], statistical analysis [2, 25], deep training [2, 27], and artificial neural network [2, 26]. Among the aforementioned techniques, deep training outperformed other techniques [2, 28]. Artificial Intelligence (AI) becomes very popular for intelligent detection to solve the issues. AI-enabled schemes discover the deep knowledge automatically from historical data, which makes wise judgments for predicting network intrusions [3, 29, 30]. Automata-enabled intrusion detection method uses Labeled Transition Systems' extension to introduce the uniform description to detect IoT networks intrusions effectively. This model describes heterogeneous networks with graphs and terms, and other IDS algorithms detect intrusions

by distinguishing abstracted action flows to solve the problems mentioned above [7].

The research aims to design the intrusion detection method in IoT using the proposed WMSA-based Deep RNN. Initially, the input is subjected to the pre-processing module for eliminating the redundant data. Consequently, the Wrapper approach is introduced for feature selection, where the feature subset selection algorithm searches good subsets to evaluate the feature subsets. Finally, the Deep RNN is trained using the developed WMSA, which is integrated with WWO and the MSO.

The major contributions of the paper are:

- **Proposed WMSA:** The proposed WMSA optimization algorithm is developed by integrating the WWO and the MSA algorithm for the optimal tuning of weights and biases of the Deep RNN classifier.
- **Proposed WMSA-based Deep RNN for intrusion detection in IoT:** The proposed WMSA-based Deep RNN is adapted for detecting intrusions in IoT. Here, the Deep RNN is trained by the proposed WMSA.

The remaining sections of the paper are arranged as follows: Section 2 elaborates the the conventional techniques with the challenges faced. Then, the system model of IoT is described in Section 3, and the proposed method for intrusion detection using WMSA-based Deep RNN is portrayed in Section 4. Finally, the outcomes of the proposed strategy is provided in Section 5, and Section 6 present the conclusion.

2 Literature Survey

The existing techniques of intrusion detection in IoT and its limitations are deliberated below: Nadia Chaabouni et al. [1] devised IDS for IoT security. This framework employed the systems with their mobility, heterogeneity, and IoT-specific challenges. The method provides an optimal success rate in privacy and security. Still, more efforts are needed to detect zero-day attacks and IDSs were developed to update the considered attacks. Ying Zhang et al. [2] developed IDS. In addition, the optimization technique helps to achieve the high detection rate. The method needs more time for the training process. Jiaqi Li et al. [3] presented Artificial Intelligence (AI)-enabled IDS for IoT networks. This approach flexibly captured the flow of the network with a global view and detected the attack intelligently. Initially, the Bat algorithm with binary differential mutation and the swarm division was introduced for selecting the appropriate features. After that, the random forest was employed

for changing the sample weights based on weighted voting for classifying the flows. The method failed to consider real networks for traffic classification to improve the system's performance. Yair Meidan et al. [4] developed an approach for IoT, termed N-BaIoT, for extracting the behavior snapshots of the network. Besides, autoencoders were used to detect the traffic of the IoT devices. This methodology needs more time to detect the intrusion.

Vishal Sharma et al. [5] presented a behavior rule specification-enabled misbehavior detection method, termed BRIoT, to detect intrusion in cyber-security systems. This approach was utilized for verifying behavior rules correctly but the method failed to consider IoT- embedded CPSs due to high run time and memory,. Junaid Arshad et al. [6] modeled collaborative intrusion detection framework (COLIDE) in IoT. This framework combines host and network-enabled detection for effectively achieving intrusion detection based on 6LoWPAN. The method did not find the detection accuracy better. Yulong Fu et al. [7] developed IDS for the IoT networks. This is the extension of the Labelled Transition system. The method failed to include other methods for describing and evaluating the contents of translating packets. Lianbing Deng et al. [8] a presented transfer training algorithm for detecting the mobile network intrusion detection. This framework uses the combination of Fuzzy C-means clustering and Principal component analysis (PCA) for detecting the intrusion. The method improves detection efficiency with a lower false-positive rate, but constructing a good combination classifier was difficult.

2.1 Challenges

The challenges confronted by the conventional strategies are deliberated below:

- In [2], the deep training based IDS was developed to detect intrusion in IoT. However, this method failed to optimize the deep network parameters.
- In intrusion detection, the low detection efficiency exists because of obtained high false-positive rate. This aspect is explained by the lack of good studies on intrusion events, challenging [31].
- Intrusion detection at edge router failed to consider devices' behavior that leads to maximal communication overheads among nodes [6].
- The absence of the assessment methodologies, appropriate metrics, and the general framework used to evaluate alternative IDS techniques is challenging [31].

3 IoT System Configuration Model

The system configuration of IoT to detect the intrusion and is shown in Figure 1. IoT networks contain typical network elements, including a workstation, routers, IoT devices, and laptops. The network platforms initially include the attacking and normal Virtual Machines (VMs) and tap and the firewall. In addition, the two Network Interface Cards (NICs) and the packet filtering firewall are configured in the environment. In this case, one NIS is configured in WAN, whereas another in LAN. The main purpose of using the firewall ensures the dataset labeling process validity and the network access are managed by monitoring the outgoing and the incoming network packets. VMs are utilized to communicate through the Internet, pass the traffic based

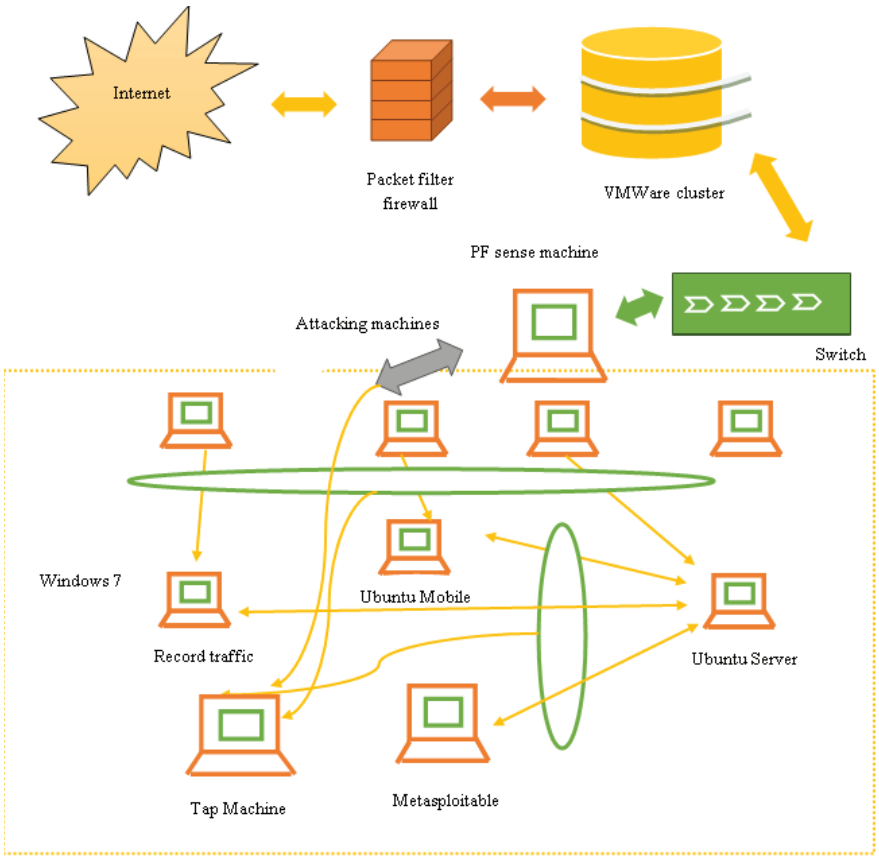

Figure 1 System configuration model of IoT network.

on the PFSense machine, pass the traffic using switch, and second firewall are further routed into the Internet. The VMs of attacking machines works DDoS, port scanning, and other Botnet related attacks [37].

4 Proposed Water Moth Search Algorithm-based Deep RNN for Intrusion Detection in IoT

This research aims to devise an intrusion detection strategy using the proposed WMSA-based Deep RNN [32]. Initially, the redundant data is eliminated using the pre processing step. Then the feature selection is done by using the wrapper approach to employ the further processing. Then, the classification is employed using Deep RNN, which is learned by newly devised WMSA. The developed WMSA algorithm integrates WWO [34] and MSA [35], The schematic view of the developed WMSA-based Deep RNN as shown in Figure 2.

Let, G be the input data with the attributes is represented as

$$G = \{G_{MN}\}; (1 \leq M \leq D); (1 \leq N \leq Y) \tag{1}$$

where, the term G_{MN} refers to the M^{th} data record in N^{th} attribute, D is the data point, and Y is the attributes, and $[D \times Y]$ is the database size.

4.1 Pre-processing

Pre-processing is considered an important stage used for processing of thousands of data for the efficient performance enhancement. The pre-processing

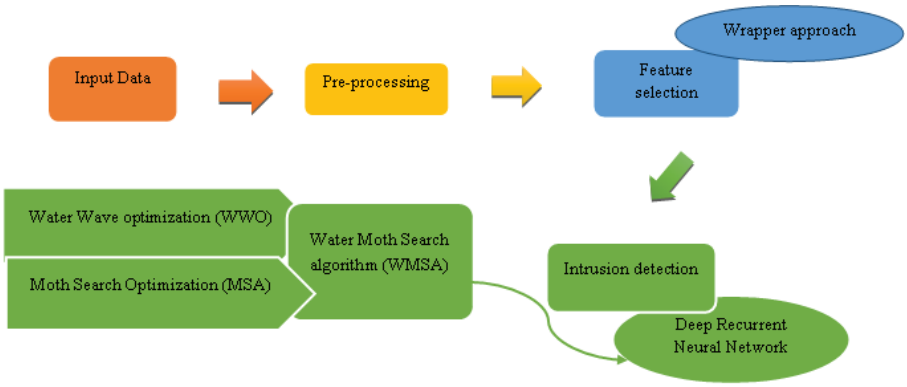

Figure 2 Proposed WMSA-based Deep RNN model for intrusion detection in IoT.

is utilized for describing the processing of data for better representations. Then, the pre-processed data is expressed as K with $[D \times Y]$ dimension.

4.2 Wrapper Approach for Feature Selection

After pre-processing, the feature selection is considered as a significant method for mining the significant data. This step is essential for eliminating the noisy features from the massive datasets to design robust training models. A large number of features causes may generate the complexity by slowing the training process and maximizes the risk of a learned classifier, which may lead to overfitting of data confusion. Thus, the selection of appropriate features may bring more advantages to the real-world application. The data K is taken for selecting the features. The feature selection is employed using the wrapper approach [33] for choosing optimal features for better detection. The resultant selected features are represented as R. Now, the dimension G becomes $[D \times \mathrm{O}]$, such that $\mathrm{O} < Y$.

4.3 Proposed Water Moth Search Algorithm-based Deep RNN for Detecting Intrusion in IoT

The intrusion detection in IoT using the developed WMSA-based Deep RNN is presented. The selected features$R$are taken as the input of intrusion detection using Deep RNN, and the training is carried out based on developed WMSA, which is the combination of WWO [34] and MSO [35]. The proposed WMSA-based Deep RNN detects the intrusion by tuning optimal weights in Deep RNN for intrusion detection in IoT. The explanation and the architecture of Deep RNN and its training procedure are given below.

(a) System model of Deep recurrent neural network

Deep RNN [32] can accept the input feature vector of any length. Besides, it uses the information of the previous iteration for the processing of the current information. Hence, the above mentioned features help the classifier to generate the output more accurately and effectively. Also, it is better compared the conventional classifiers due to its sequential processing. Figure 3 shows the system model of Deep RNN.

The input vector of a^{th} layer at s^{th} time, and is represented as $R^{(a,s)} = \{r_1^{(a,s)}, r_2^{(a,s)}, \ldots r_i^{(a,s)}, \ldots r_y^{(a,s)}\}$ and $H^{(a,s)} = \{H_1^{(a,s)}, H_2^{(a,s)}, \ldots H_i^{(a,s)}, \ldots H_x^{(a,s)}\}$ refers to the output of a at time s as $J^{(b,r)} = \{J_1^{(b,r)}, J_2^{(b,r)}, \ldots J_i^{(b,r)}, \ldots J_y^{(b,r)}\}$, respectively. The unit is the combination

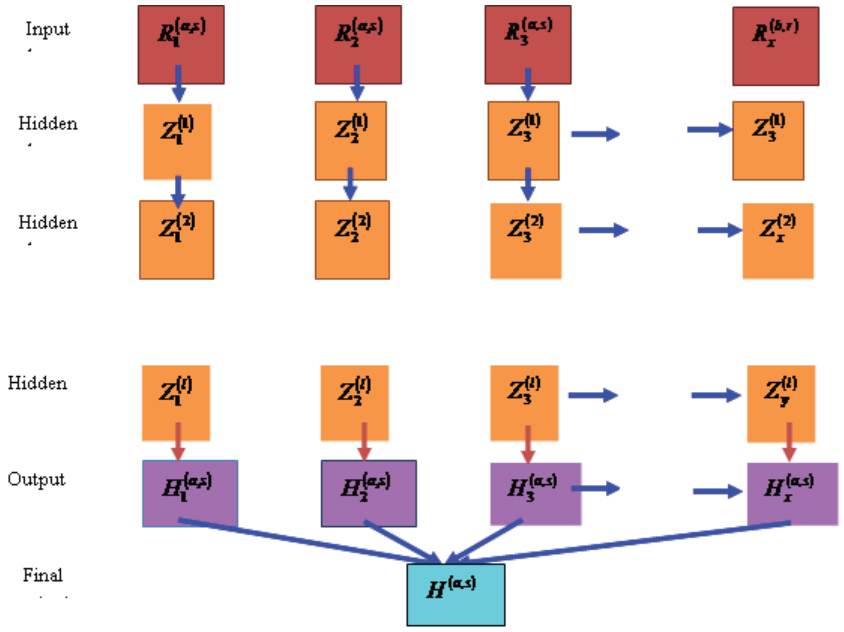

Figure 3 Structure of Deep RNN classifier.

of the output and input pair. Let y is the units present in a^{th} layer, and 1 is the arbitrary unit. Besides, the total units of $(a-1)^{th}$ layer is k and D. The weights of $(a-1)$ to a is represented as $U^{(a)} \in I^{x \times D}$, and $y^{(a)} \in I^{x \times x}$ is the layer b's recurrent weight. I is the weights set. The input vector components are given by,

$$R_j^{(a,s)} = \sum_{u=1}^{D} q_{ju}^{(a)} H_u^{(a-1,s)} + \sum_{j'}^{x} v_{jj'}^{(a)} H_{j'}^{(a,s-1)} \tag{2}$$

where, the terms $q_{ju}^{(a)}$ and $v_{jj'}^{(a)}$ refer to the elements of $U^{(a)}$ and $y^{(a)}$, the term j' is the layer a's arbitrary number. The output vector at layer a is,

$$H_j^{(a,s)} = \alpha^{(a)}(R_j^{(a,s)}) \tag{3}$$

where, $\alpha^{(a)}$ is the activation function In addition, $\alpha(R) = \frac{1}{(1+e^{-R})}$ be the logistic sigmoid function, the sigmoid function, denoted as $\beta(R) = \tanh(R)$, and rectified linear unit function (ReLU), is specified as $\beta(F) = \max(R, 0)$.

The process is simplified and is expressed by,

$$H^{(a,s)} = \alpha^{(a)} \cdot (U^{(a)} H^{(a-1,s)} + v^{(a)} \cdot H^{(a,s-1)}) \tag{4}$$

Where, $H^{(a,s)}$ is the classifier's output.

(b) Training of Deep recurrent neural network using Water Moth Search algorithm

The proposed WWSA integrates WWA and MSA and is used to train the Deep RNN. The searching behavior of moths inspires the MSA [35], and the solution is an update of its position change. The behavior of moths consider in MSA are phototaxis and the levy flight. In addition, it negotiates the complex operations, so the execution of MSA is flexible and easy. Similarly, the WWO [34] is motivated by the waves to enhance the optimization issues by enhancing the dimensionality. As a result, the method can attain a trade-off between exploration and exploitation. Moreover, it speeded the convergence process. Hence, the combination of WWO and MSA modelled to improve the performance. The algorithmic procedure is illustrated below.

(a) Initialization: The moth's location is initialized randomly. The solution space containing $e$number of the moths, and it is given below,

$$\mathrm{X} = [\mathrm{X}_q]; \quad 1 \leq q \leq S \tag{5}$$

where, X_q signifies the q^{th} moth, and S denotes the total count of moths.

(b) Fitness evaluation: It is calculated for the estimation of optimal solution to find best solution by updating the location.

$$MSE = \frac{1}{g}\left[\sum_{p=1}^{g} Q_{target} - H^{(a,s)}\right] \tag{6}$$

where, g is the total sample of training, $H^{(a,s)}$ and Q_{target} are the estimated and target output of the classifier.

(c) Location update using levy flights: Once fitness is evaluated, then the updated solution using levy flight, and the equation is expressed as,

$$\mathrm{X}_q^{s+1} = \mathrm{X}_q^s + \eta \cdot P(x) \tag{7}$$

where, the term X_q^s indicates the moth location at s^{th} iteration, and levy flight movement is denoted as $P(x)$. The parameter η represents the scaling factor and is given by,

$$\eta = \frac{H_{\max}}{s^2} \tag{8}$$

where, the term $H_{\max}$ signifies the maximal step walk, and the term s refer to the current iteration. The levy distribution equation is given by,

$$P(x) = \frac{(\mu-1)\Gamma(\mu-1)\sin\left(\frac{\pi(\mu-1)}{2}\right)}{\pi x^{\mu}} \tag{9}$$

where, the term x exceeds $0.\Gamma(z)$, then it is said to be gamma function.

(d) Fly straightly: The updated solution based on the position of the moth$q$; their flight is formulated by,

$$\mathrm{X}_q^{s+1} = \vartheta \times (\mathrm{X}_q^s + \alpha \cdot (\mathrm{X}_{best}^s - \mathrm{X}_q^s)) \tag{10}$$

$$\mathrm{X}_q^{s+1} = \vartheta \mathrm{X}_q^s + \vartheta\alpha \mathrm{X}_{best}^s - \vartheta\alpha \mathrm{X}_q^s \tag{11}$$

$$\mathrm{X}_q^{s+1} = \mathrm{X}_q^s(\vartheta - \vartheta\alpha) + \vartheta\alpha \mathrm{X}_{best}^s \tag{12}$$

where, the term X_{best}^s refer to the optimal position of moth, and the term α refer to acceleration, ϑ signifies the scaling function. When the movement of fly is straight, the moth's position is influenced by the position of light source. To enhance the system's performance and tackle the optimization problems of MSA, the WWO algorithm is utilized. As per WWO, its propagation for updation is represented by,

$$\mathrm{X}_q^{s+1} = \mathrm{X}_q^s + Rand(-1,1)\rho J_b \tag{13}$$

$$\mathrm{X}_q^s = \mathrm{X}_q^{s+1} - Rand(-1,1)\rho J_b \tag{14}$$

Substitute Equation (14) in Equation (12), the solution becomes,

$$\mathrm{X}_q^{s+1} = (\mathrm{X}_q^{s+1} - Rand(-1,1)\rho J_b)(\vartheta - \vartheta\alpha) + \vartheta\alpha \mathrm{X}_{best}^s \tag{15}$$

$$\mathrm{X}_q^{s+1} = \mathrm{X}_q^{s+1}(\vartheta - \vartheta\alpha) - Rand(-1,1)\rho J_b(\vartheta - \vartheta\alpha) + \vartheta\alpha \mathrm{X}_{best}^s \tag{16}$$

$$\mathrm{X}_q^{s+1} - \mathrm{X}_q^{s+1}(\vartheta - \vartheta\alpha) = \vartheta\alpha \mathrm{X}_{best}^s - Rand(-1,1)\rho J_b(\vartheta - \vartheta\alpha) \tag{17}$$

$$\mathrm{X}_q^{s+1}(1 - \vartheta + \vartheta\alpha) = \vartheta\alpha \mathrm{X}_{best}^s - Rand(-1,1)\rho J_b(\vartheta - \vartheta\alpha) \tag{18}$$

Thus, the final update equation of the proposed WMSA-based Deep RNN is expressed as,

$$\mathrm{X}_q^{s+1} = \frac{1}{1 - \vartheta + \vartheta\alpha}[\vartheta\alpha \mathrm{X}_{best}^s - Rand(-1,1)\rho J_b(\vartheta - \vartheta\alpha)] \tag{19}$$

where, the term ρ denotes the wavelength, the J_b signifies the length of b^{th} dimension of search space, and the terms ϑ and α represent the scaling factor and the acceleration.

(e) Best solution evaluation: After updating the location, the solution space calculates the fitness. After the identification of new solution, the older solution is replaced by new.

(f) Termination: The steps (b–e) are repeated until the attainment of the best solution. Algorithm 1 portrays the pseudo-code of the proposed WMSA-based Deep RNN algorithm.

Algorithm 1 Pseudo-code for the proposed WMSA-based Deep RNN algorithm.

Input: Moths population $\mathrm{X} = \lfloor \mathrm{X}_q \rfloor; 1 \le q \le S$
Output: Best solution
Procedure:
Begin
Initialized the moth population
Fitness function computation
While $s < MaxGen$ (*Maximal Iterations*
Arrange the moth individuals
for $q = 1$ **to** $\frac{S}{2}$ **do**
levy flights equation is generated using equation (7)
end for q
for $q = \frac{S}{2+1}$ **to** S **do**
If $rand > 0.5$ **then**
Compute X_q^{s+1} **using equation (10)**
Else
Generate X_q^{s+1} **using equation (19)**
End if
End for q
New solutions are found out and intensities are compared again
$s = s + 1$
end while
Best solution is obtained
End

5 Results and Discussion

The evaluation of the proposed methodology by considering the performance metrics is illustrated in this section.

5.1 Experimental Setup

The experimental analysis is employed in Python using a PC with 2GB RAM, the Intel i3 core processor, and Windows 10 OS.

5.2 Description of Dataset

The performance is employed using the BoT-IoT dataset [36]. This dataset is created to design the network environment for performing the intrusion detection mechanism. It integrates the Botnet and normal traffic. The CSV files, argus files, and pcap files, are the different format of source file. The pcap files are 69.3 GB in dimension, with 72.000.000 records. The csv format in extracted flow traffic is 16.7 GB in size. Besides, it has DoS, DDoS, key-logging, and service scan attacks based on the protocol. The KDD cup 1999 dataset [38] comprises of content features, traffic features, and basic features. In addition, it has 24 training attack and additional 14 test data attack.

5.3 Evaluation Metrics

The performance of the proposed RWW-based NN is analyzed by concerning metrics, like sensitivity, accuracy, and specificity.

5.3.1 Accuracy

It is closeness of the expected output, and is represented as,

$$Accuracy = \frac{\mathrm{T}_p + \mathrm{T}_n}{\mathrm{T}_p + \mathrm{T}_n + \mathrm{E}_p + \mathrm{E}_n} \tag{20}$$

where, T_p represent true positive, E_p indicate false positive, T_n indicate true negative, and E_n represents false negative, respectively.

5.3.2 Sensitivity

It is the measure of positives detected by the developed method, and it is represented as,

$$Sensitivity = \frac{\mathrm{T}_p}{\mathrm{T}_p + \mathrm{E}_n} \tag{21}$$

5.3.3 Specificity

It is the measure of negatives detected by the developed method and is formulated as.

$$\textit{Specificity} = \frac{\mathrm{T}_n}{\mathrm{T}_n + \mathrm{E}_p} \tag{22}$$

5.4 Performance Analysis

The performance of the developed technique is evaluated by varying the input features and the hidden neurons numbers. Finally, the analysis is evaluated based on the metrics.

5.4.1 Analysis using dataset-1

The WMSA-based Deep RNN's performance is analyzed by BoT-IoT dataset is detailed in this section.

(a) Analysis based on input features

Figure 4 depicts the based on input features. The analysis using accuracy is shown in Figure 4(a). The accuracy value estimated by WMSA-based Deep RNN with number of features 6, 9, 12, and 15 with 90% of training data is 0.929, 0.937, 0.938, and 0.935. The analysis using sensitivity is depicted in Figure 4(b). The sensitivity evaluated by proposed model with number of features 6, 9, 12, and 15 for 90% of training data are 0.929, 0.938, 0.942, and 0.958. The specificity analysis is depicted in Figure 4(c). The specificity evaluated by the proposed model with number of features 6, 9, 12, and 15 for 90% of training data are 0.974, 0.976, 0.977, and 0.982.

(b) Analysis using hidden neurons

The analysis by considering different hidden neurons is illustrated in Figure 5. Figure 5(a) portrays the performance by evaluating the accuracy. The accuracy value estimated by WMSA-based Deep RNN with hidden neurons 40, 60, 80, and 100 for 90% of training data is 0.907, 0.909, 0.928, and 0.948 respectively. The analysis using sensitivity is depicted in Figure 5(b). For the hidden neurons 40, 60, 80, and 100 the sensitivity estimated by WMSA-based Deep RNN using 90% of training data are 0.906, 0.948, 0.950, and 0.957. The analysis using specificity is depicted in Figure 5(c). For hidden neurons 40, 60, 80, and 100 the specificity evaluated by WMSA-based Deep RNN for 90% of training data are 0.933, 0.934, 0.938, and 0.950.

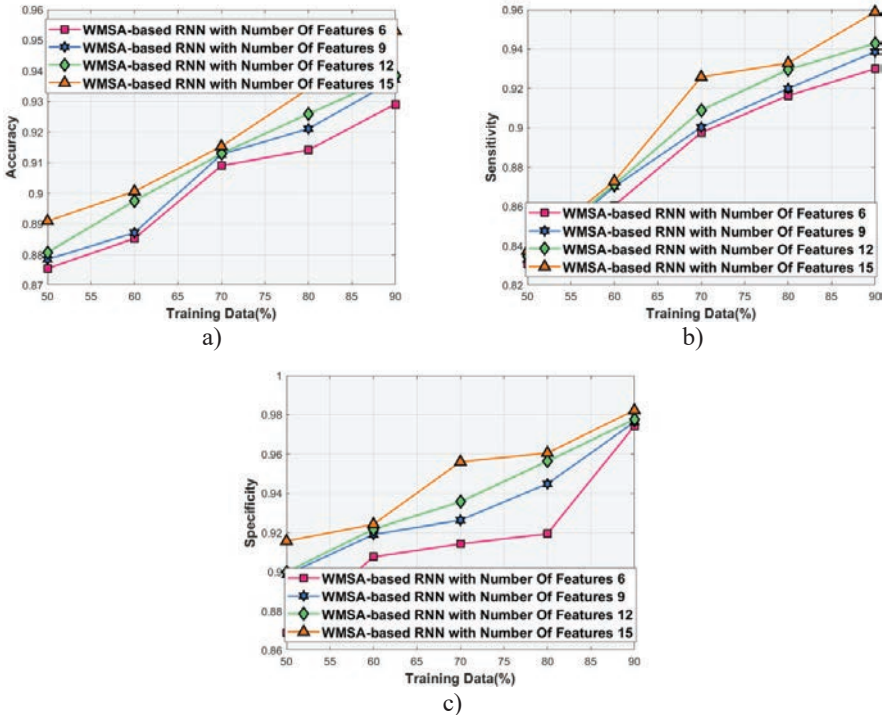

Figure 4 Performance analysis with number of features (a) accuracy, (b) sensitivity, (c) and specificity.

5.4.2 Analysis using dataset-2

The WMSA-based Deep RNN's performance is analyzed by KDD Cup 1999 Data dataset is detailed in this section.

(a) Analysis based on input features

Figure 6 depicts the analysis of input features. The analysis by considering the accuracy is shown in Figure 6(a). The accuracy value estimated by WMSA-based Deep RNN with number of features 6, 9, 12, and 15 using 90% of training data is 0.912, 0.923, 0.932, and 0.944. The analysis based on sensitivity is shown in Figure 6(b). The sensitivity evaluated by proposed model with number of features 6, 9, 12, and 15 using 90% of training data are 0.907, 0.910, 0.916, and 0.957. The analysis of specificity is depicted in Figure 6(c). The specificity evaluated by the proposed model with number of

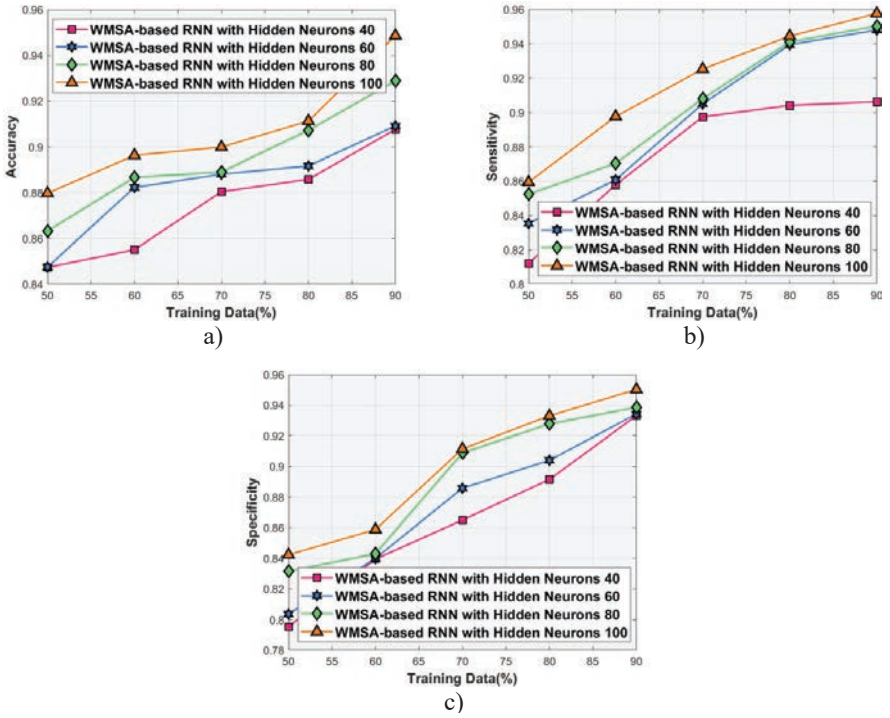

Figure 5 Performance analysis with hidden neurons (a) accuracy, (b) sensitivity, (c) and specificity.

features 6, 9, 12, and 15 using 90% of training data are 0.939, 0.945, 0.960, and 0.968.

(b) Analysis using hidden neurons

The analysis by considering different hidden neurons is illustrated in Figure 7. Figure 7(a) shows the performance analysis using accuracy. The accuracy value estimated by WMSA-based Deep RNN with hidden neurons 40, 60, 80 and 100 using 90% of training data is 0.873, 0.894, 0.898, and 0.921. The analysis by considering the sensitivity is shown in Figure 7(b). The sensitivity with hidden neurons 40, 60, 80, and 100 estimated by WMSA-based Deep RNN are 0.894, 0.921, 0.935, and 0.945. The analysis using specificity is depicted in Figure 7(c). The specificity with hidden neurons 40, 60, 80, and 100 using 90% of training data are 0.918, 0.927, 0.929, and 0.932.

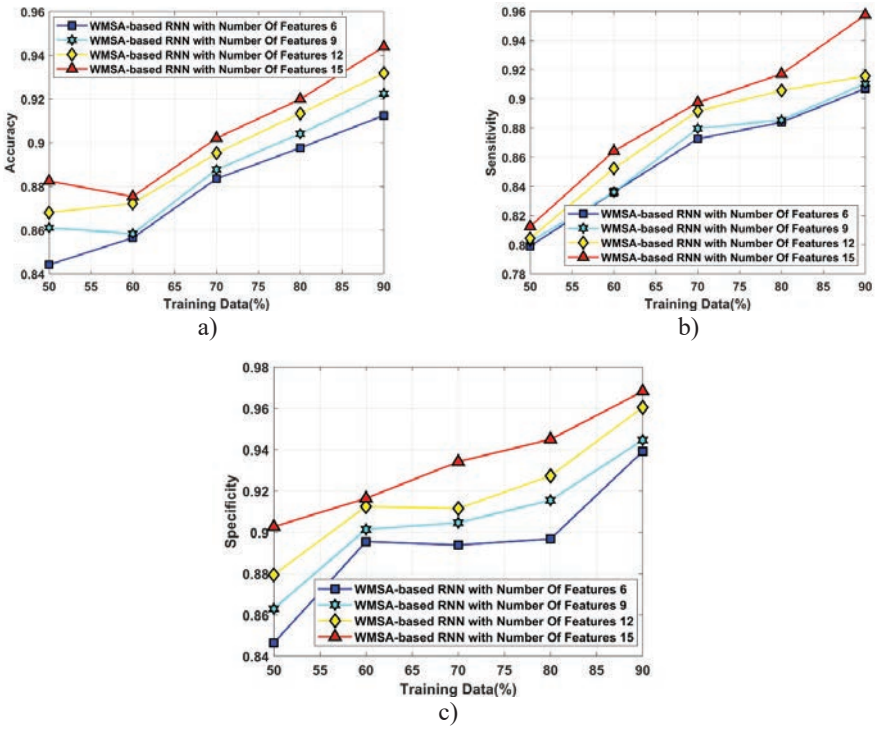

Figure 6 Performance analysis with number of features (a) accuracy, (b) sensitivity, (c) and specificity.

5.5 Comparative Techniques

The conventional techniques, Artificial Neural Network (ANN) [1], Improved Genetic Algorithm and Deep Belief Network (Improved GA+DBN) [2], and the Deep Autoencoder [4] are considered for the comparison analysis with the developed WMSA-based Deep RNN.

5.5.1 Analysis using dataset-1

(a) Analysis of the intrusion detection with respect to training data percentage

The comparative analysis of intrusion detection is shown in Figure 8. Figure 8(a) depicts the analysis using accuracy based on training data percentages. The accuracy computed by the proposed WMSN-based Deep RNN is 0.95, which is 9.4%, 5.2%, and 8.4% better than ANN, Improved GA+DBN, and Deep Autoencoder, respectively for 80% of training data. The analysis

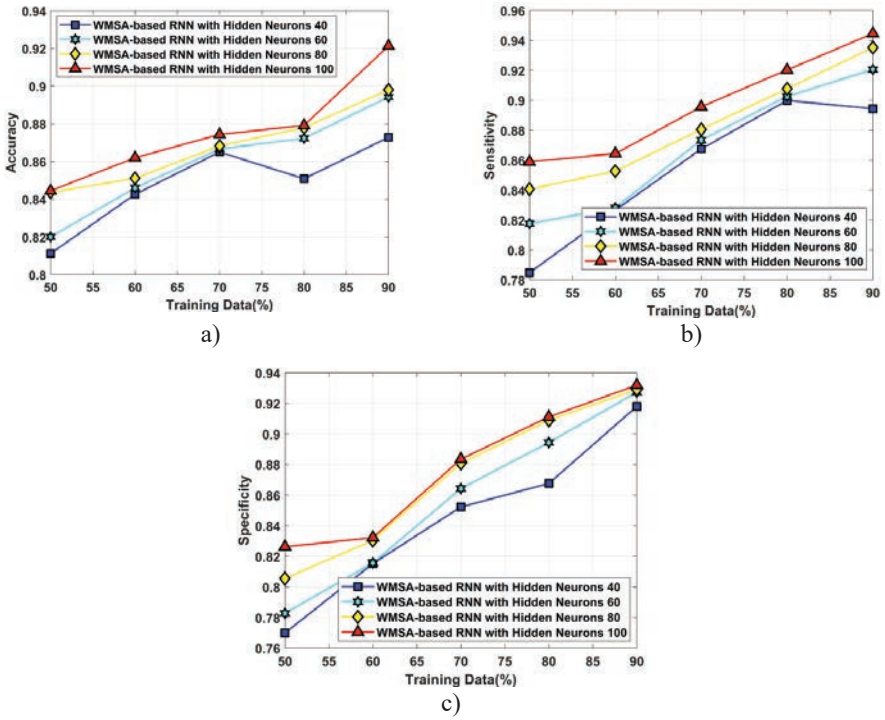

Figure 7 Performance analysis with hidden neurons (a) accuracy, (b) sensitivity, (c) and specificity.

by sensitivity is deliberated in Figure 8(b). The sensitivity computed by the proposed model with 50% of training data is 0.883, which is 9.3%, 9.3%, and 14.5% better than ANN, Improved GA+DBN, and Deep Autoencoder, respectively. The analysis using specificity parameter is deliberated in Figure 8(c). The specificity computed by proposed WMSN-based Deep RNN with 50% of training data is 0.915, which is 8.96%, 2.51%, and11.03%, better than ANN, Improved GA+DBN, Deep Autoencoder.

(b) Analysis of the intrusion detection with respect to hidden neurons

Figure 9 presents the analysis of methods by varying the hidden neurons using performance metrics. The analysis using accuracy is deliberated in Figure 9(a). When hidden neurons = 80, the accuracy of methods computed by ANN, Improved GA+DBN, Deep Autoencoder, and proposed WMSN-based

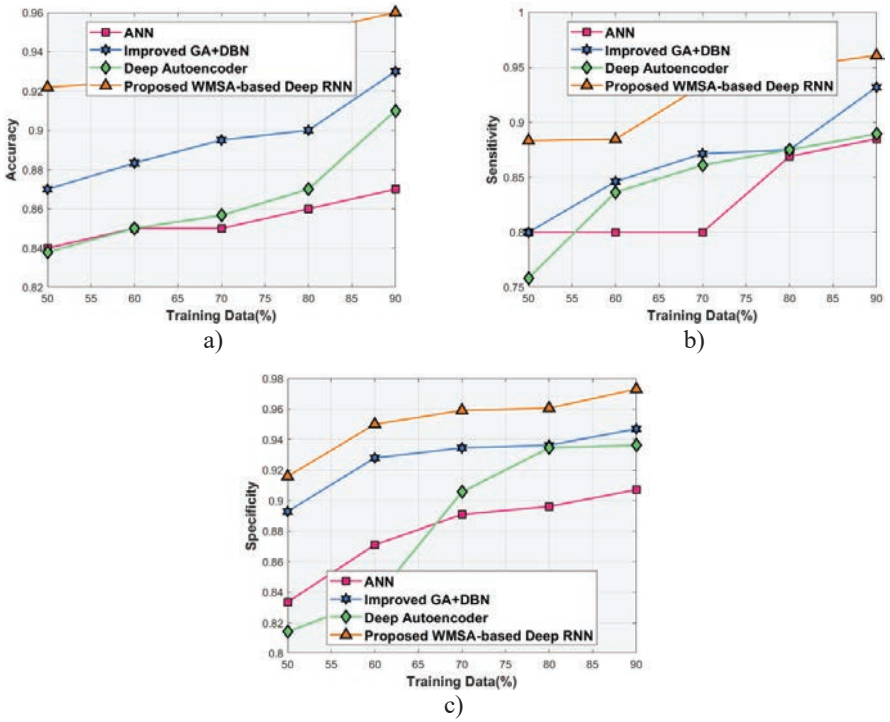

Figure 8 Analysis of methods by varying the training data percentage (a) Accuracy (b) Sensitivity (c) Specificity.

Deep RNN are 0.838, 0.91, 0.865, and 0.91. The sensitivity analysis is deliberated in Figure 9(b). When hidden neurons = 100, the sensitivity of methods computed by ANN, Improved GA+DBN, Deep Autoencoder, and proposed WMSN-based Deep RNN are 0.871, 0.948, 0.880, and 0.954. The specificity analysis is deliberated in Figure 9(c). When hidden neurons = 80, the specificity of methods computed by ANN, Improved GA+DBN, Deep Autoencoder, and proposed WMSN-based Deep RNN are 0.810, 0.892, 0.840, and 0.902.

5.5.2 Analysis using dataset-2

(a) Analysis based on training data percentage

The comparative analysis is shown in Figure 10 by varying the training percentage. Figure 10(a) depicts the analysis using accuracy. The accuracy computed by the ANN, Improved GA+DBN, and Deep Autoencoder using

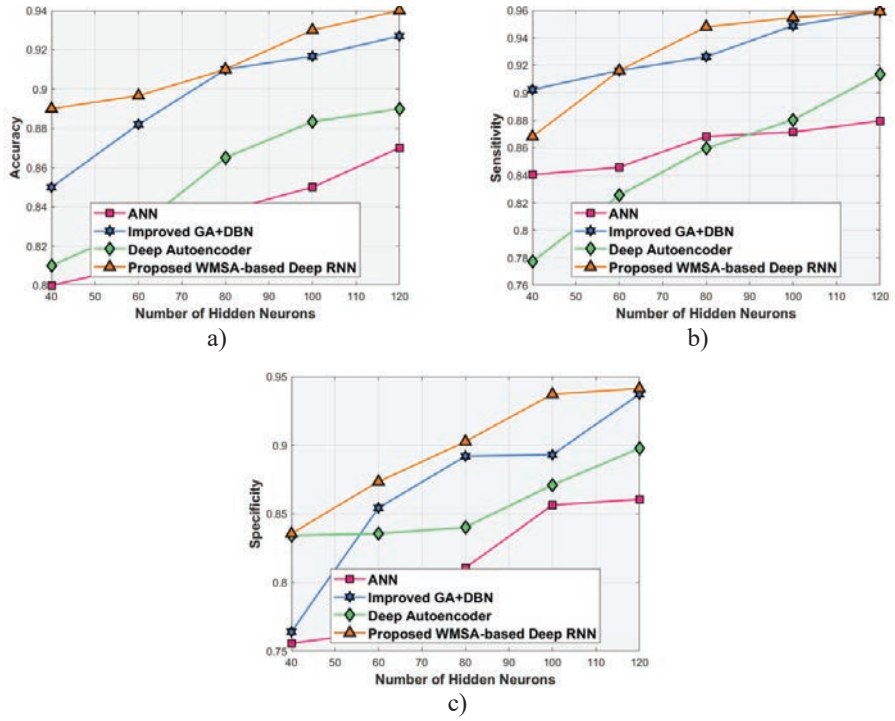

Figure 9 Analysis of methods by varying the hidden neurons (a) Accuracy (b) Sensitivity (c) Specificity.

80% of training data are 10.8%, 5.52%, and 11.3% worse than the proposed WMSN-based Deep RNN with the value of 0.941. The analysis by sensitivity is deliberated in Figure 10(b). The sensitivity computed by ANN, Improved GA+DBN, and Deep Autoencoder using 50% of training data are 10.18%, 8.33% and 12.9% worse than the proposed model with the value of 0.864. The analysis using specificity parameter is deliberated in Figure 10(c). The specificity computed by ANN, Improved GA+DBN, and Deep Autoencoder using 80% of training data are 9.3%, 2.9% and 14.7% worse than the proposed WMSN-based Deep RNN with the value of 0.909.

(b) Analysis of the intrusion detection with respect to hidden neurons

Figure 11 presents the analysis of methods by varying the hidden neurons in terms of performance metrics. Figure 11(a) shows the analysis using accuracy. When hidden neurons = 80, the accuracy of methods computed

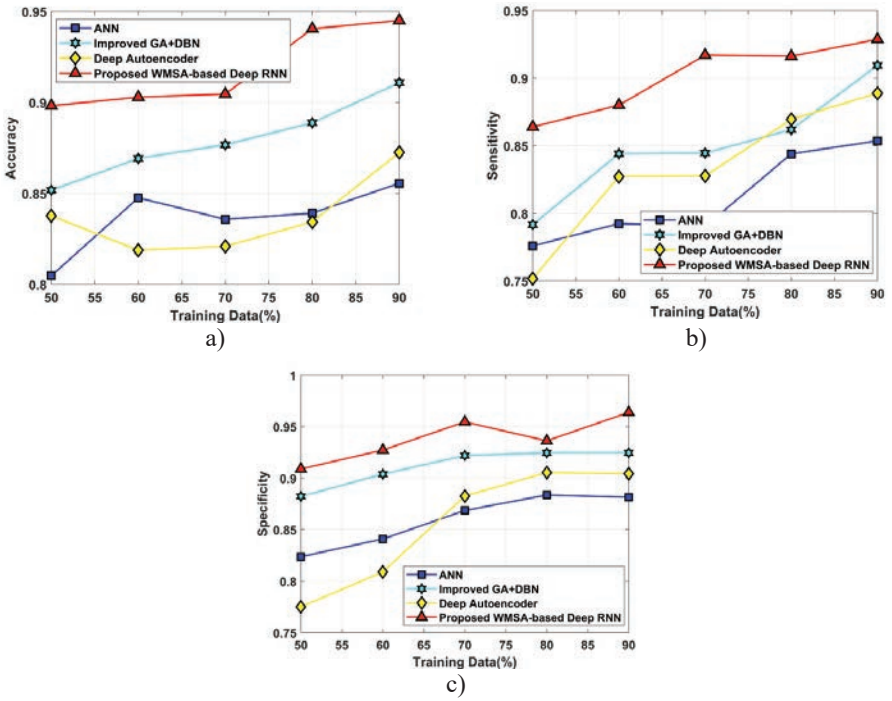

Figure 10 Analysis of methods by varying the training data percentage (a) Accuracy (b) Sensitivity (c) Specificity.

by ANN, Improved GA+DBN, Deep Autoencoder, and proposed WMSN-based Deep RNN are 0.831, 0.874, 0.845, and 0.907. Figure 11(b) shows the analysis using sensitivity. When hidden neurons = 100, the sensitivity computed by ANN, Improved GA+DBN, Deep Autoencoder, and proposed model are 0.848, 0.914, 0.878, and 0.952. Figure 11(c) shows the analysis using specificity. When hidden neurons = 80, the specificity of methods computed by ANN, Improved GA+DBN, Deep Autoencoder, and proposed WMSN-based Deep RNN are 0.800, 0.889, 0.820, and 0.864.

5.5.3 Comparative analysis by considering Neptune attack

(a) Using dataset-1

Figure 12 depicts the comparative analysis by considering Neptune attack by utilizing BoT-IoT dataset. The analysis by considering the accuracy is illustrated in Figure 12(a). The accuracy evaluated by the proposed WMSN-based Deep RNN is 13.7% better than ANN, 5.2% better than Improved

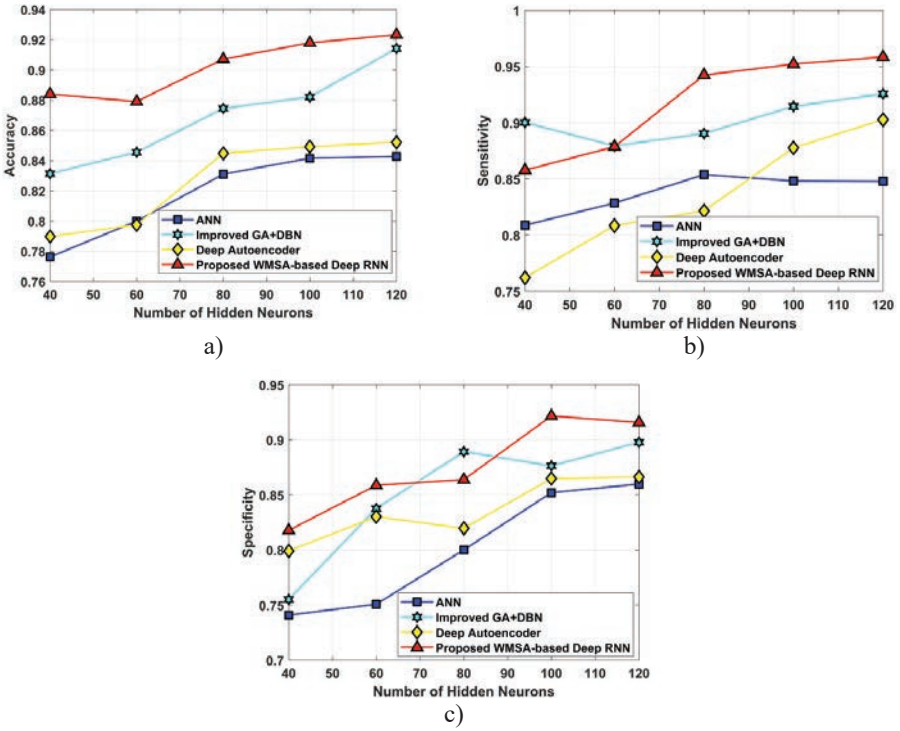

Figure 11 Analysis of methods by varying the hidden neurons (a) Accuracy (b) Sensitivity (c) Specificity.

GA+DBN and 11.19% better than Deep Autoencoder for 80% of training data. The analysis by considering the sensitivity is illustrated in Figure 12(b). The sensitivity computed by ANN, Improved GA+DBN, Deep Autoencoder, and proposed WMSN-based Deep RNN using 90% of training data are 0.768, 0.802, 0.774, and 0.837. Figure 12(c) shows the analysis by considering the specificity. The specificity computed by the proposed WMSN-based Deep RNN is 9.3% superior to ANN, 1.7% superior than Improved GA+DBN and 14.76% superior than Deep Autoencoder for 60% of training data.

(b) Using dataset-2

Figure 13 depicts the comparative analysis of the proposed method by considering the Neptune attack using KDD Cup 1999 Data dataset. The accuracy of the methods is illustrated in Figure 13(a). The accuracy computed by ANN, Improved GA+DBN, and Deep Autoencoder are 9.3%, 5.8%, and

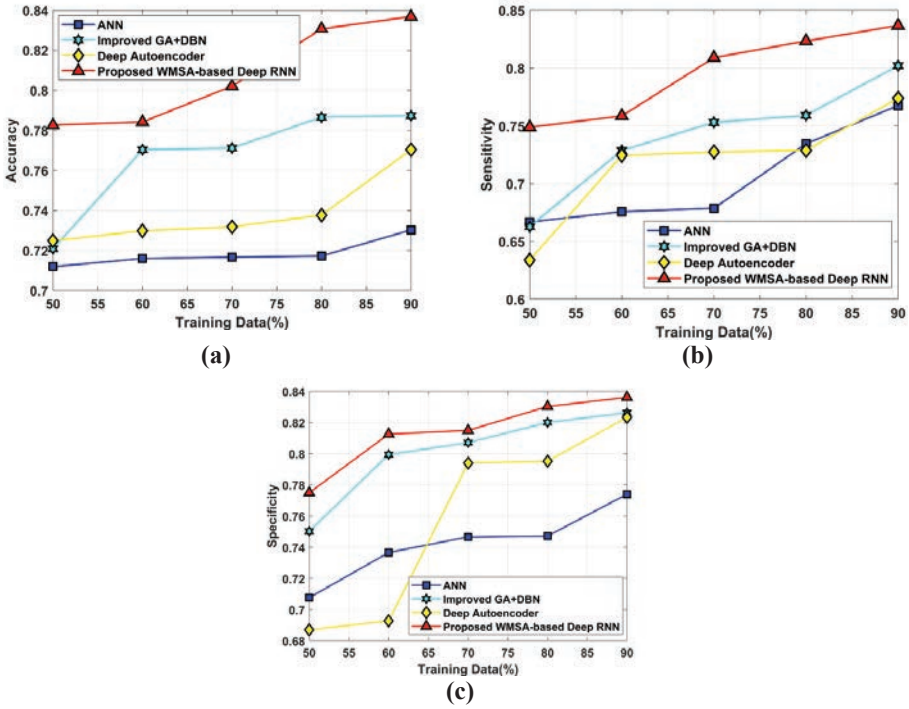

Figure 12 The comparative analysis using the Neptune attack for dataset-1 in terms of (a) accuracy, (b) sensitivity and (c) specificity.

2.6% worse than the proposed WMSN-based Deep RNN with the value of 0.805 for 80% of training data. The analysis by considering the sensitivity is illustrated in Figure 13(b). The sensitivity of methods computed by ANN, Improved GA+DBN, Deep Autoencoder, and proposed WMSN-based Deep RNN are 0.768, 0.830, 0.791, and 0.806 for 90% of training data. The analysis by considering the specificity is illustrated in Figure 13(c). The specificity computed by ANN, Improved GA+DBN, and Deep Autoencoder, are 11.8%, 2% and 6.9% worse than the and proposed WMSN-based Deep RNN with the value of 0.750 for 60% of training data.

5.5.4 Analysis using computation time

The comparative analysis based on the computation time is illustrated in Figure 14. The analysis by considering the dataset-1 is depicted in Figure 14(a). The computation time computed by ANN, Improved GA+DBN, Deep Autoencoder, and proposed WMSN-based Deep RNN are 17.586,

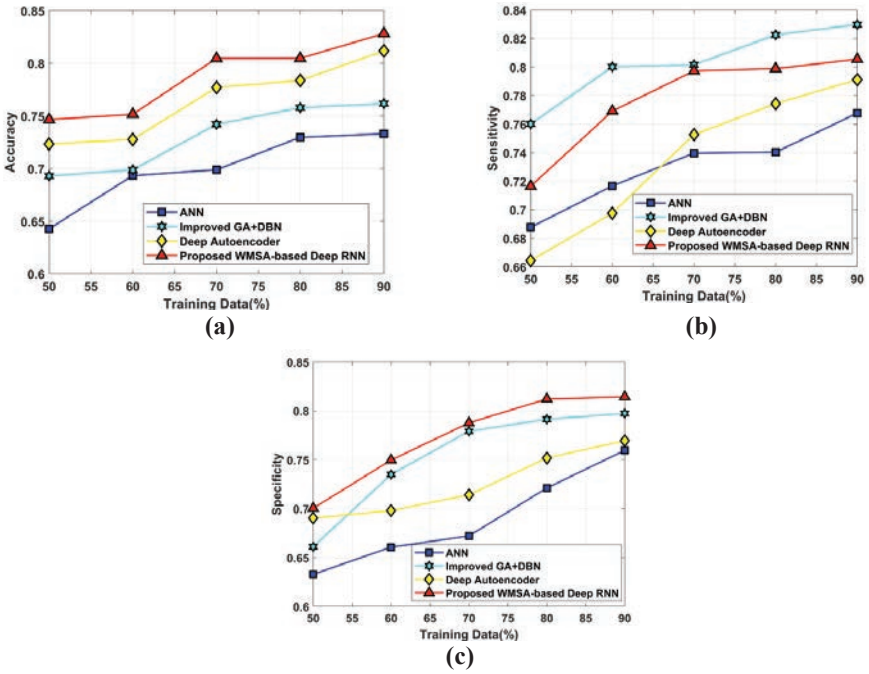

Figure 13 The comparative analysis using the Neptune attack for dataset-2 in terms of (a) accuracy, (b) sensitivity and (c) specificity.

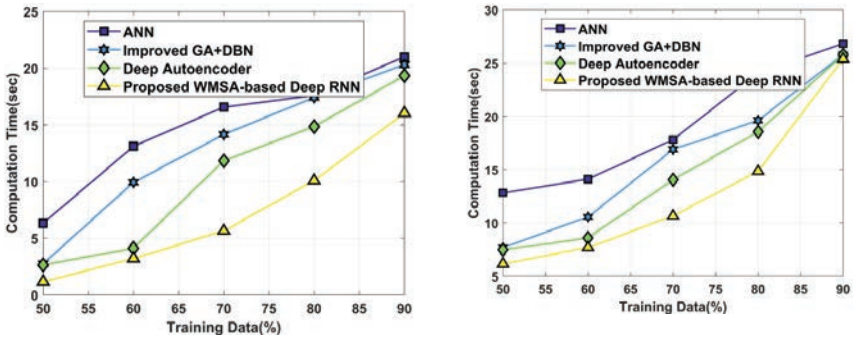

Figure 14 Comparative analysis based on the computation time (a) dataset-1 and (b) dataset-2.

17.416, 14.854, and 10.081 with 80% of training data. The analysis by considering the dataset-2 is depicted in Figure 14(b). The computation time computed by ANN, Improved GA+DBN, Deep Autoencoder, and proposed WMSN-based Deep RNN are 26.836, 25.802, 25.787, and 25.396 with 90% of training data. Thus, by considering the analysis the computation time estimated by the proposed method is lower compared to the existing technique.

5.6 Comparative Discussion

Table 1 shows the comparative discussion using sensitivity, accuracy, and specificity. The maximal accuracy of 0.96 estimated by proposed WMSN-based Deep RNN. In contrast, the existing ANN, Improved GA+DBN, and Deep Autoencoder are 0.87, 0.93, and 0.91. The maximal sensitivity computed by proposed WMSN-based Deep RNN is 0.973, whereas the existing ANN, Improved GA+DBN, and Deep Autoencoder are 0.884, 0.931, and 0.889, respectively. The maximal specificity value measured by WMSN-based Deep RNN is 0.973, whereas the existing ANN, Improved GA+DBN,

Table 1 Comparative discussion

	Variation	Metrics	ANN	Improved GA+DBN	Deep Autoencoder	Proposed WMSN-based Deep RNN
Dataset 1	Training	*Accuracy*	0.87	0.93	0.91	**0.96**
	data	*Sensitivity*	0.884	0.931	0.889	**0.96**
	percentage	*Specificity*	0.907	0.946	0.936	**0.973**
	Hidden	*Accuracy*	0.87	0.927	0.89	**0.94**
	neurons	*Sensitivity*	0.879	0.959	0.913	**0.959**
		Specificity	0.86	0.937	0.897	**0.941**
Dataset 2	Training	*Accuracy*	0.856	0.911	0.873	0.945
	data	*Sensitivity*	0.854	0.909	0.889	0.929
	percentage	*Specificity*	0.882	0.925	0.904	0.964
	Hidden	*Accuracy*	0.843	0.914	0.852	0.923
	neurons	*Sensitivity*	0.848	0.926	0.903	0.958
		Specificity	0.860	0.898	0.866	0.916

and Deep Autoencoder are 0.907, 0.946, and 0.936 by considering the training data. Similarly, for the KDD cup1999 dataset outperformed other state of art techniques.

The proposed WMSN-based Deep RNN outperformed other state of art technique in terms of accuracy, sensitivity and the specificity. This performance enhancement is achieved because; the proposed optimization algorithm has the fast convergence rate by the avoidance of the local minima. Thus, by using this, the optimization efficiency is achieved with faster performance. Besides, the deep RNN has the ability to process the input of any length and has the ability to remember the information because it depends on the previous iteration. Thus, the proposed method obtains the effective overall performance enhancement for the intrusion detection.

6 Conclusion

The intrusion detection in IoT is evaluated using the Deep RNN to enhance the detection accuracy. Existing techniques of the intrusion detection based on ANN reveal poor performance, which is addressed based on developed model of detection. This is carried out based on selected features derived from pre-processing data. The Wrapper approach is used for feature selection that provides better features for detecting the intrusion effectively in IoT. Using the feature vectors, the intrusion detection is done by the proposed WMSA-based Deep RNN that detects the intrusion behavior using the lowest fitness value. Experimentation of the developed model is performed using BOT-IoT dataset. The classification is highly precise and the developed method using performance metrics reveal that the proposed WMSA-based Deep RNN attained a maximal accuracy, sensitivity and specificity of 0.96, 0.960, and 0.973. However, the proposed method has the gradient vanishing problem and the training time required is little greater. Hence, in future, the efficiency of proposed method will be evaluated using other datasets. Besides, the efficiency of the classifier will be explored using the other optimization technique.

References

[1] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki. 2019. Network intrusion detection for IoT security based on training

techniques. *IEEE Communications Surveys and Tutorials*, 21: 2671–2701.

[2] Y. Zhang, P. Li, and X. Wang. 2019. Intrusion detection for IoT based on improved genetic algorithm and deep belief network. *IEEE Access*, 7: 31711–31722.

[3] J. Li, Z. Zhao, R. Li, H. Zhang, and T. Zhang. 2018. AI-based two-stage intrusion detection for software defined iot networks. *IEEE Internet of Things Journal*, 6(2): 2093–2102.

[4] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici. 2018. N-baiot—network-based detection of iot botnet attacks using deep auto encoders. *IEEE Pervasive Computing*, 17(3): 12–22.

[5] V. Sharma, I. You, K. Yim, R. Chen, and J.-H. Cho. 2019. BRIoT: Behavior Rule Specification-Based Misbehavior Detection for IoT-Embedded Cyber-Physical Systems. *IEEE Access*, 7: 118556–118580.

[6] J. Arshad, M. A. Azad, M. M. Abdellatif, M. H. U. Rehman, and K. Salah. 2018. COLIDE: a collaborative intrusion detection framework for Internet of Things. *IET Networks*, 8(1): 3–14.

[7] Y. Fu, Z. Yan, J. Cao, O. Koné, and X. Cao. 2017. An automata based intrusion detection method for internet of things. *Mobile Information Systems.*

[8] L. Deng, D. Li, X. Yao, D. Cox, and H. Wang. 2019. Mobile network intrusion detection for IoT system based on transfer training algorithm. *Cluster Computing*, 22(4): 9889–9904.

[9] J. Rifkin. 2014. The Zero Marginal Cost Society: The Internet of Things, the Collaborative Commons, and the Eclipse of Capitalism: Book.

[10] Grau. 2014. The Internet of Secure Things What is Really Needed to Secure the Internet of Things? j Icon Labs.

[11] O. Vermesan, and P. Friess. 2014. Internet of Things Applications – From Research and Innovation to Market Deployment Book. *River Publishers.*

[12] A.-S. Abduvaliyev, K. Pathan, J. Zhou, R. Roman, and W.-C. Wong. 2013. On the Vital Areas of Intrusion Detection Systems in Wireless Sensor Networks. Communications Surveys and Tutorials, *IEEE*, 15: 1223–1237.

[13] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini. 2015. Security, privacy and trust in internet of things: The road ahead. *Computer Networks*, 76: 146–164.

[14] F. Muhammad, W. Anjum, and K. S. Mazhar. 2015. A Critical Analysis on the Security Concerns of Internet of Things (IoT). *Perception*, 111: 1–6.

[15] T. Borgohain, U. Kumar, and S. Sanyal. 2015. Survey of Operating Systems for the IoT Environment. *arXiv preprint arXiv:1504.02517*, 6: 2479–2483.

[16] H. Haddad Pajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo. 2018. A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting. *Future Generation Computer Systems*, 85: 88–96.

[17] M. Conti, A. Dehghantanha, K. Franke, and S. Watson. 2017. Internet of Things Security and Forensics: Challenges and Opportunities. *Elsevier Future Generation Computer Systems Journal*, 78: 544–546.

[18] H. Haddad Pajouh, R. Javidan, R. Khayami, D. Ali, and K.-K. R. Choo. 2016. A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks. *IEEE Transactions on Emerging Topics in Computing*, (99): 1–1.

[19] T. A. Tang, L. Mhamdi, D. M. Lernon, S. A. R. Zaidi, and M. Ghogho. 2018. Deep training approach for network intrusion detection in software defined networking. *In proceedings of International Conference on Wireless Networks and Mobile Communications.*

[20] B. Arrington, L. E. Barnett, R. Rufus, and A. Esterline. 2016. Behavioral modeling intrusion detection system (BMIDS) using internet of things (IoT) behavior-based anomaly detection via immunity inspired algorithms. *In Proceedings of the 25th International Conference on Computer Communication and Networks (ICCCN'16)*, 1–6.

[21] V. Kumar, and P. Sangwan. 2012. Signature Based Intrusion Detection System Using SNORT. *International Journal of Computer Applications & Information Technology*, 1(3).

[22] S. Raza, L. Wallgren, and T. Voigt. 2017. SVELTE: Real-time intrusion detection in the Internet of Things. *Ad Hoc Networks*, 11(8): 2661–2674.

[23] E. Bertino, and N. Islam. 2017. Botnets and Internet of Things Security. *Computer*, 50(2): 76–79.

[24] W. Lee and S. J. Stolfo. 1998. Data mining approaches for intrusion detection. In Proceedings of the 7th USENIX Security Symposium. *San Antonio, TX*, 120–132.

[25] L. Khan, M. Awad, and B. Thuraisingham. 2007. A new intrusion detection system using support vector machines and hierarchical clustering. *VLDB J.*, 16(4): 507–521.

[26] E. Hodo, X. Bellekens, A. Hamilton, P.-L. Dubouilh, E. Iorkyase, C. Tachtatzis, and R. Atkinson. 2016. Threat analysis of iot networks using artificial neural network intrusion detection system. *in 2016 International Symposium on Networks, Computers and Communications (ISNCC)*, 1–6.

[27] A. Diro, and N. Chilamkurti. 2017. Distributed attack detection scheme using deep training approach for Internet of Things. *Future Generat. Comput. Syst.*, 282: 761–768.

[28] R. Beghdad. 2008. Critical study of neural networks in detecting intrusions. *Computers and Security*, 27: 168–175.

[29] L. Buczak, and E. Guven. 2017. A survey of data mining and machine training methods for cyber security intrusion detection. *IEEE Communications Surveys and Tutorials*, 18(2): 1153–1176.

[30] R. Li, Z. Zhao, X. Zhou, G. Ding, Y. Chen, Z. Wang, and H. Zhang. 2017. Intelligent 5G: When cellular networks meet artificial intelligence. *IEEE Wireless Communications*, (99): 2–10.

[31] Garcia-Teodoro, and J. Diaz-Verdejo. 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1–2): 18–28.

[32] M. Inoue, S. Inoue, and T. Nishida. 2018. Deep recurrent neural network for mobile human activity recognition with high throughput. *Artificial Life and Robotics*, 23(2): 173–185.

[33] R. Kohavi, and G. H. John. 1997. Wrappers for feature subset selection. *Artificial intelligence*, 97(1–2): 273–324.

[34] Y.-J. Zheng. 2015. Water wave optimization: a new nature-inspired metaheuristic. *Computers & Operations Research*, 55: 1–11.

[35] G.-G. Wang. 2018. Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. *Memetic Computing*, 10(2): 151–164.

[36] BOT IOT dataset taken from. https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/bot_iot.php. accessed on April 2020.

[37] N. Koroniotis, N. Moustafa, E. Sitnikova, B. Turnbull. 2018. Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset.

[38] KDD cup 1999 dataset. "http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html" last accessed on July, 2021.

Biographies

Rekha P. M. is currently working as an Associate Professor and Head of the Department Information Science and Engineering, JSS Academy of Technical Education, Bangalore and has twenty years of teaching experience. Completed PhD in Cloud Computing, VTU with BMS as research center. Published several 20 papers in national and international journals which includes Scopus indexed and web of science journals. Have carried out invited talks which includes technical talk on IOT and workshop on Cloud Simulators. She is a member of IRED, IAENG, ISTE. FIELD OF RESEARCH: Internet of Things, Cloud Computing, Microcontroller & Embedded systems.

Nagamani H Shahapure has been associated with JSSATE, Bangalore since 2001. Has over 23 years of experience, which includes 3 years in Infosys and 20 years in JSS. Completed BE in CSE from PDA College of Engg. Gulbarga University in 1995. MTech from BMS College of Engineering, VTU. Completed PhD in Cloud Computing, VTU with BMS as research center. Published several (13) papers in national and international journals which includes Scopus indexed journals.Have carried out invited talks which includes technical talk on chatbots and workshop on Cloud Simulators,

FIELD OF RESEARCH: Cloud Computing, Edge Computing, Fog Computing, Serverless Computing, Open Source and Web Technologies.

Punitha M. is currently working as an Assistant Professor in Information Science and Engineering Department, JSS Academy of Technical Education, Bangalore and has two years of teaching experience. Research interest in Internet of Things, Cloud computing and Python Programming. Attended several FDP's and Workshop.

Sudha P. R. is currently working as an Assistant Professor in Information Science and Engineering Department, JSS Academy of Technical Education, Bangalore and has fifteen years of teaching experience. Research interest in Cloud computing and Network Security. Attended several FDP's and Workshop.