Computer Systems Science & Engineering
DOI: 10.32604/csse.2023.035589
Article

Tech Science Press

Artificial Algae Optimization with Deep Belief Network Enabled Ransomware Detection in IoT Environment

Mesfer Al Duhayyim[1,*], Heba G. Mohamed[2], Fadwa Alrowais[3], Fahd N. Al-Wesabi[4], Anwer Mustafa Hilal[5] and Abdelwahed Motwakel[5]

[1]Department of Computer Science, College of Sciences and Humanities-Aflaj, Prince Sattam bin Abdulaziz University, Riyadh, Saudi Arabia
[2]Department of Electrical Engineering, College of Engineering, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh, 11671, Saudi Arabia
[3]Department of Computer Sciences, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh, 11671, Saudi Arabia
[4]Department of Computer Science, College of Science and Art at Muhaeyl, King Khalid University, Abha, Saudi Arabia
[5]Department of Computer and Self Development, Preparatory Year Deanship, Prince Sattam bin Abdulaziz University, AlKharj, Saudi Arabia
*Corresponding Author: Mesfer Al Duhayyim. Email: m.alduhayyim@psau.edu.sa
Received: 26 August 2022; Accepted: 04 November 2022

Abstract: The Internet of Things (IoT) has gained more popularity in research because of its large-scale challenges and implementation. But security was the main concern when witnessing the fast development in its applications and size. It was a dreary task to independently set security systems in every IoT gadget and upgrade them according to the newer threats. Additionally, machine learning (ML) techniques optimally use a colossal volume of data generated by IoT devices. Deep Learning (DL) related systems were modelled for attack detection in IoT. But the current security systems address restricted attacks and can be utilized outdated datasets for evaluations. This study develops an Artificial Algae Optimization Algorithm with Optimal Deep Belief Network (AAA-ODBN) Enabled Ransomware Detection in an IoT environment. The presented AAA-ODBN technique mainly intends to recognize and categorize ransomware in the IoT environment. The presented AAA-ODBN technique follows a three-stage process: feature selection, classification, and parameter tuning. In the first stage, the AAA-ODBN technique uses AAA based feature selection (AAA-FS) technique to elect feature subsets. Secondly, the AAA-ODBN technique employs the DBN model for ransomware detection. At last, the dragonfly algorithm (DFA) is utilized for the hyperparameter tuning of the DBN technique. A sequence of simulations is implemented to demonstrate the improved performance of the AAA-ODBN algorithm. The experimental values indicate the significant outcome of the AAA-ODBN model over other models.

Keywords: Internet of things; deep learning; cybersecurity; ransomware detection; feature selection

1 Introduction

The Internet of Things (IoT) is a gathering of small devices or gadgets equipped with several sensors to assist an individual in executing several daily routines [1]. IoT gadgets are entrenched gadgets linked to home appliances, vehicles, and many more that allow such objects to exchange and connect data. Millions of IoT gadgets were in use, continuing to multiply [2]. This might allure a challenger to mount a malevolent assault by pointing to IoT gadgets. An adversary could alter the default identifications of these gadgets and, after utilizing such gadgets, could raise assaults on other mechanisms [3]. A honeypot was a deception program devised to detect an attacker's behaviour or activity to compromise the production mechanism. A honeypot could help as a surveillance tool and even seizure attack signs [4]. The behavioural analysis of seized signs offers valuable insights into effective system loopholes. Though honeypots do not secure IoT systems straightly, it is employed for strengthening IDS (Intrusion Detection System) and firewall located at the network periphery [5].

Several wide security systems were addressing IoT safety, chiefly by conventional cryptographic concepts [6]. But, the prevailing cryptographic solutions on separate IoT gadgets were inadequate for satisfying the complete spectrum of IoT security due to the dynamic nature of IoT networks and assaults [7]. There are numerous features of gadgets in an IoT system like device-to-device proximity transmission, Low-power and low-cost communications, huge deployment, Self-organization Inter-connectivity, self-healing features, Heterogeneity, Low Latency Communication, and Requirement of Ultra-Reliable, and Dynamic variations in the network that forms the security provision very difficult chore in IoT [8]. It raised the landscape of threats for the invaders. The investigation community in IoT was inspecting the scope of the immense volume of realistic data produced by IoT gadgets [9]. Then, they modelled several Deep Learning (DL) and Machine Learning (ML) systems for IoT security by binding information from the data. Besides, DL-related security systems could learn heterogeneous characteristics in formless data by themselves, and therefore these were heterogeneity tolerant [10]. It is utilized for detecting the new mutated assaults from their previous forms; thus, security systems do not require a patch on IoT gadgets.

The authors in [11] construct a two-phase mixed RF model, ransomware detection technique, and Markov model. Firstly the author focuses on the Windows API call series pattern and constructs a Markov model for capturing the features of ransomware. Then, the author constructs an RF-ML model for the residual data for controlling false negative (FNR) and false positive (FPR) error rates. Baek et al. [12] developed a two-phase hybrid malware recognition (2-MaD) technique for securing IoT devices from obscured malware in smart city settings. The 2-MaD includes two phases of IoT malware detection. Firstly, the opcode is extracted afterwards, acting out static analysis, and the learned data used through the Bi-LSTM model, benign files are discovered. Next, dynamic analysis can be done on files categorized as benign in nested virtual environments. Afterwards, extracting data on behaviour and process memory from the behaviour log, malware is identified through a trained EfficientNet-B3 mechanism depending on the system changes.

Basnet et al. [13] developed the novel DL-based ransomware detection architecture in the Supervisory control and data acquisition (SCADA) controlled electric vehicle charging station (EVCS) with the performance assessment of 3 DL systems, namely DNN, 1D-CNN, and LSTM-RNN. Nisa et al. [14] developed a feature fusion system for integrating the feature extracting in pre-trained Inception-v3 and AlexNet DNN with features accomplished through segmentation-based fractal texture analysis (SFTA) of image expressive the malware code. In the study, the author uses a distinct pre-trained model (Inception-V3 and AlexNet) for extracting features. The objective is to enhance the accuracy of the malware classification since these two models have qualities and characteristics for extracting various features.

The authors in [15] designed the DL method utilized in malware recognition for detecting ransomware in an emulation sequence. The author presents a specified RNN model to capture local event patterns from the ransomware series utilizing the conception of the attention mechanism. Azmoodeh et al. [16] established a DL-based algorithm for detecting Internet of Battlefield Things (IoBT) malware through the device's

Operational Code (OpCode) sequences. The author transmutes OpCodes as vector space and employs a deep Eigenspace learning model for classifying benign and malicious applications. Also, the author demonstrates the sustainability of the presented model in detecting malware and its robustness against junk code insertion attacks.

This study develops an Artificial Algae Optimization Algorithm with Optimal Deep Belief Network (AAA-ODBN) Enabled Ransomware Detection in an IoT environment. The presented AAA-ODBN technique mainly intends to recognize and categorize ransomware in the IoT environment. The presented AAA-ODBN technique follows a three-stage process: feature selection, classification, and parameter tuning. In the first stage, the AAA-ODBN technique uses AAA based feature selection (AAA-FS) technique to elect feature subsets. Secondly, the AAA-ODBN technique employs the DBN model for ransomware detection. At last, the dragonfly algorithm (DFA) is utilized for the hyperparameter tuning of the DBN approach. A sequence of simulations is executed to demonstrate the improved performance of the AAA-ODBN approach. In short, the paper's contribution is summarized as follows.

- Develop a new AAA-ODBN model for Ransomware Detection in an IoT environment.
- Employ the AAA-FS technique to select the optimal set of features.
- Employ DFA with the DBN model for the classification process.

The rest of the paper is organized as follows. Section 2 offers the proposed model, and Section 3 provides the performance validation. Lastly, Section 4 concludes the study.

2 The Proposed Model

This study developed a novel AAA-ODBN model for ransomware detection in the IoT environment. The presented AAA-ODBN technique mainly intends to recognize and categorize ransomware in the IoT environment. Fig. 1 demonstrates the overall process of the AAA-ODBN approach. As shown in the figure, the presented AAA-ODBN technique follows a three-stage process: feature selection, classification, and parameter tuning. Initially, input data is preprocessed, and then an optimal set of features is chosen by AAA. Moreover, the DFA-DBN model is applied to the detection process.

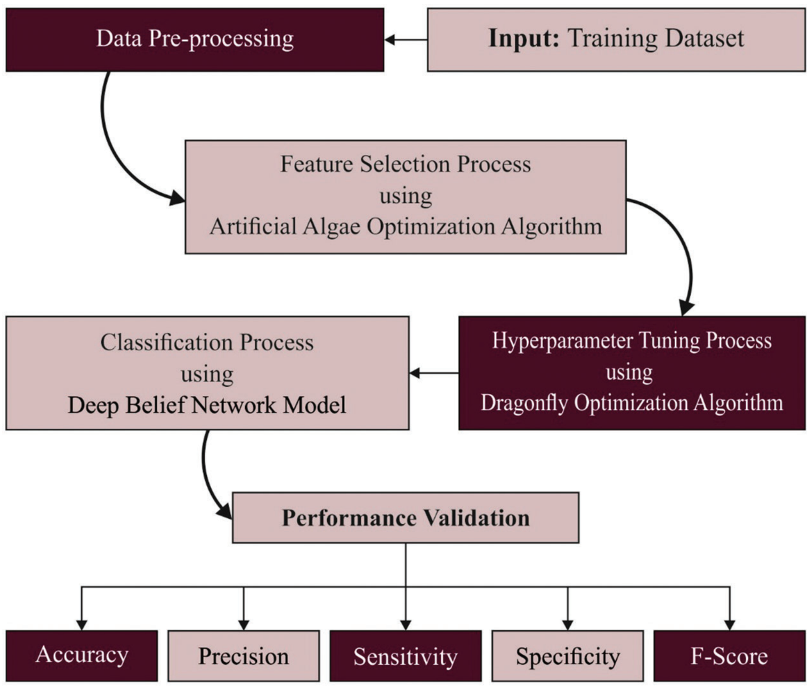
Data Pre-processing
Input: Training Dataset
Feature Selection Process using Artificial Algae Optimization Algorithm
Classification Process using Deep Belief Network Model
Hyperparameter Tuning Process using Dragonfly Optimization Algorithm
Performance Validation
Accuracy
Precision
Sensitivity
Specificity
F-Score

Figure 1: Overall process of AAA-ODBN approach

2.1 Feature Selection Using AAA-FS Model

In the first stage, the AAA-ODBN exploited the AAA-FS technique to elect feature subsets. AAA mimics real algae to survive by determining and moving towards a suitable platform and recreating the upcoming generation [17]. In these subsections, they would concisely examine AAA as follows:

$$\textit{Population of algal colony} = \begin{bmatrix} x_{11} & x_{12} & x_{1d} \\ x_{21} & x_{22} & x_{2d} \\ \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & x_{nd} \end{bmatrix} \tag{1}$$

Set $x_i = (x_{i1},\ x_{i2},\ \cdots,\ x_{id})$, $i = 1, 2, \ldots n$, while x_i represents potential solution. The algal cells in an algal colony (AC) were assumed to move towards suitable locations with numerous sources. Meanwhile, the colony accomplishes an ideal position; the optimum solution was accomplished.

The AAA contains evolutionary, adaptation, and helical movements. The AC tries to move toward that optimum place by developing, adapting, and moving. Note that a critical concept in AAA is the size of AC of i^{th} AC characterized as S_i, $i = 1, 2, \ldots, n$. Interrelated s2 to real algae, the AC will reproduce and increase to a large size in an exact living position. S_i is set to 1 at the earlier stage and transformed to the fitness value of *i-th* AC, for instance, the value of the objective function. The optimum objective function $f(x_i)$ is the high S_i is shown below.

$$S_i = size\ (x_i) \tag{2}$$

$$\mu_i = S_i + \frac{4f(x_i)}{S_i + 2f(x_i)} \tag{3}$$

$$S_i^{t+1} = \mu_i S_i^t, i = 1, 2, \ldots, n \tag{4}$$

From the expression, $f(x_i)$ shows the objective function, μ_i denotes the upgrade coefficient of S_i, t represents the existing generation.

Algae make natural movement towards the water region with adequate light and other nutrients. In AAA, all the ACs move towards an optimum AC with the maximum size or optimum main function. However, this motion is based on selecting 3 dissimilar algal cells and modifying their positions, expressed as follows.

$$x_{im}^{t+1} = x_{im}^t + \left(x_{jm}^t - x_{im}^t\right)(sf - w_i)p \tag{5}$$

$$x_{ik}^{t+1} = x_{ik}^t + \left(x_{jk}^t - x_{ik}^t\right)(sf - w_i)\cos\alpha \tag{6}$$

$$x_{il}^{t+1} = x_{il}^t + \left(x_{jl}^t - x_{il}^t\right)(sf - w_i)\sin\beta \tag{7}$$

Now, m, k, and l specify uniformly distributed arbitrary numbers within one and d, x_{im}, x_{ik} and x_{il} simulate x, y, and z coordinates of i^{th} AC, j specifies the index of neighbor AC and is accomplished by means of tournament choice, p represents independent arbitrary values within [−1, 1], α and β show arbitrary degree of arc amongst [0, 2π], sf represents a shear force that exists as viscous drag, w_i characterizes friction surface area of *i-th* AC that is associated with the size of ACs:

$$w_i = 2\pi r_i^2 \tag{8}$$

$$r_i = \left(\sqrt[3]{\frac{3S_i}{4\pi}}\right) \tag{9}$$

Now, r_i signifies the radius of the hemisphere of i^{th} AC, and S_i denotes their size.

Generally, AC with good nutrient sources develops quickly, and unusual nutrient sources might weaken to pass away. Similarly, in AAA, AC x_i becomes larger once it moves towards the ideal position and accomplishes potential solutions as follows:

$$biggest\ = arg \max \{size(x_i)\},\ i = 1, 2, \ldots, n \tag{10}$$

$$smallest\ = arg \min \{size(x_i)\},\ i = 1, 2, \ldots, n \tag{11}$$

$$smallest_j = biggest_j,\ j = 1, 2, \ldots,\ d. \tag{12}$$

Now, the smallest and biggest signify AC, j signifies arbitrarily chosen algal cells.

AC suffered from starvation in unsatisfactory light and nutrients during this developing procedure. It is the process whereby the starved AC tries to move toward the biggest colony and adapt to the situation. Once the main function accomplishes the best value, the corresponding AC residues the starvation level not changed. Otherwise, the starvation value increases by 1. The later motion of AC terminates, the AC has maximal starvation value (Eq. (13)) adapted to the biggest AC with probability A_p, and it is given below:

$$x_s = arg \max \{starvation(x_i)\},\ i = 1, 2, \ldots, n \tag{13}$$

$$x_{sj}^{t+1} = \begin{cases} x_{sj}^t + \left(biggest_j - x_{sj}^t\right) \times rand1,\ if\ rand2\ < A_p; \\ x_{sj}^t,\ otherwise. \end{cases} j = 1, 2,\ \ldots,\ d \tag{14}$$

Now, s shows an index of AC that is maximum starvation values, and starvation (x_i) measures the starvation level of ACs x_i, j represents the index of the algal cell, rand1 and rand2 produce stochastic values within [0,1], A_p symbolizes adaptation probability that defines adaptation occurs or not, A_p specifies constant usually fixed amongst 0.3 and 0.7.

During the presented method, the fitness function (FF) utilized for balancing betwixt the classifier accuracy (maximal) and the count of particular features from all the solutions (minimal) attained with the particular feature, Eq. (10) symbolizes the FF for calculating the solution.

$$Fitness = \alpha\gamma_R(D) + \beta\frac{|R|}{|C|} \tag{15}$$

At this time, $\gamma_R(D)$ implies the classifier error rate of providing classifications (K-nearest neighbor (KNN) classifier). $|R|$ stands for the cardinality of a particular set, and $|C|$ refers to the entire feature count from the data, α, and β depict the 2 variables equivalent to the impact of subset length and classifier quality. $\in$ [1,0] and $\beta = 1 - \alpha$.

2.2 Ransomware Detection Using DBN

The AAA-ODBN technique employs the DBN model for ransomware detection at this stage. DBN is the base model because it shows improved efficiency in lower speed regulation conditions in pretraining [18].

(1) RBM: DBN is a multi-layer NN comprising a sequence of stacked RBMs. The architecture of RMB is a bipartite graph, viz., nodes in the layer aren't interconnected, whereby the initial layer is the input node V_v, and the next layer is the hidden node V_h, its state space with $\{0, 1\}$ or real number $\mathbb{R}$. W^γ indicates the weight connection coefficient matrices of V_v and V_h.

Every node could take distinct states. Also, the state of these models is defined once the state of every node is defined and represents the degree to which the method takes the state; furthermore, the model was estimated using the energy function. Fig. 2 illustrates the framework of the DBN technique.

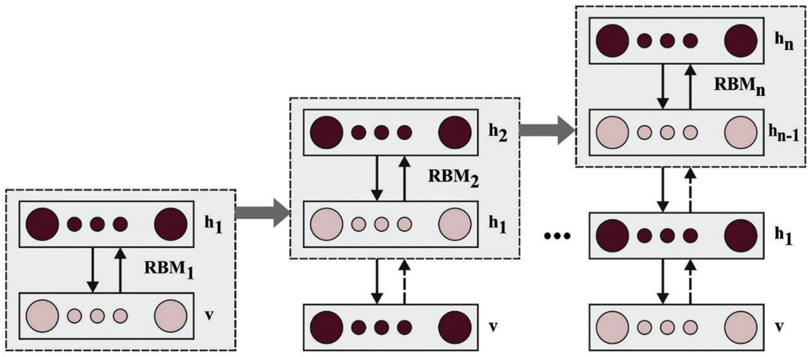

Figure 2: Structure of DBN

The energy function is determined by Eq. (16) once the state space of V_v is $\{0, 1\}$:

$$E(v,\ h) = -\sum\nolimits_{i,j} v_i W_{ij} h_j - \sum\nolimits_{i\epsilon visibile} a_j v_j - \sum\nolimits_{j\epsilon hidden} b_j h_j \tag{16}$$

(2) K-step Contrastive Divergence Methodology (k-CDM): the default Gibbs sampling RBM training model is more ineffective since the k-CDM training method is used. There exist 2 one-step models, “binary to binary” and “Gaussian to Gaussian”. The CD-1 (binary-to-binary) is given below:

$$Loss = \sum (v - p_{v'})^2 \tag{17}$$

Next, based on $(h|v)$, if $p_h > rand(0, 1)$, afterward $h = 1$, then, $h = 0$. The recreated information is returned while the computation is completed, and $p(v|h)$ is defined by computation:

$$\begin{cases} p_h = sigmoid(v \cdot W \ + b) \\ p_{v'} = sigmoid(h \cdot W^{\gamma T} + a) \\ p_{h'} = sigmoid(p_{v'} \cdot W \ + b) \end{cases} \tag{18}$$

The calculation of W^e, Δa^e, and Δb^e are:

$$\begin{cases} \Delta W^e = \left(v^T \cdot p_h - p_v^T,\ p_{h'}\right)/n \\ \Delta a^e = \Sigma(v - p_{v'})/n \\ \Delta b^e = \Sigma(p_h - p_{h'})/n \end{cases} \tag{19}$$

Next, W^{e+1}, a^{e+1}, and b^{e+1} is evaluated:

$$\begin{cases} W^{e+1} = W^e + m\Delta W^{e-1} + r\Delta W^e - dW^e \\ a^{e=1} = a^e + m \cdot \Delta a^{e-1} + r \cdot \Delta a^e \\ b^{e=1} = b^e + m \cdot \Delta b^{e-1} + r \cdot \Delta b^e \end{cases} \tag{20}$$

In CD-1 (Gaussian to Gaussian), the energy function was re-determined by:

$$E(v,\ h) = -\sum\nolimits_{i,j} \frac{v_i}{\sigma_i} \frac{h_j}{\sigma_j} W_{ij} + \sum\nolimits_{i\epsilon visible} \frac{(a_i - v_i)^2}{2\sigma_i^2} + \sum\nolimits_{j\epsilon hidden} \frac{(b_j - h_j)^2}{2\sigma_j^2} \tag{21}$$

When the K-step is set to one-step, afterwards $\sigma_i = 1$, $\sigma_j = 1$ and the Eq. (18) in the procedure is different too:

$$\begin{cases} p_h \sim N(v \cdot W + a, \sigma), h = v \cdot W + a \\ p_{v'} \sim N(h \cdot W^T + b, \sigma), v' = h \cdot W^T + b \\ p_{h'} \sim N(v' \cdot W + a, \sigma), h' = v' \cdot W + a \end{cases} \tag{22}$$

While Eq. (19) changed to:

$$\begin{cases} \Delta W^e = \left(v^T \cdot h - v'^T \cdot p_{h'}\right)/n \\ \Delta a^e = \Sigma(v - v')/n \\ \Delta b^e = \Sigma(h - h')/n \end{cases} \tag{23}$$

(3) Building DBN: the DBN method is constructed by connecting the RBM layer. Furthermore, the HDBN method would be constructed afterwards, resolving the hybrid data fusion problem.

2.3 Parameter Tuning Using DFA

Finally, the DFA is utilized for the hyperparameter tuning of the DBN model. Dragonfly has unique swarming behaviours that encompass migration (dynamic swarm) and hunting (static swarm) [19]. Static swarming has major features such as local movement and fast evolution in the flying direction. In static swarming, the dragonfly hunts for food while flying back and forth in a smaller group, whereas in dynamic swarming, the dragonfly moves in one direction over a longer distance. Such behaviours are comparable to the 2 stages of optimization using metaheuristics, such as exploration and exploitation, correspondingly.

In DFA, five major aspects exist to update the location of dragonflies in a swarm: distraction, separation, alignment, cohesion, and attraction.

The separation S_i indicates the collision avoidance from one another in the neighbour range and is evaluated by Eq. (24),

$$S_i = \sum\nolimits_{k=1}^{N} X - X_k \tag{24}$$

In Eq. (24), X indicates the existing location, X_k indicates the location of k-th neighbouring individuals, and N indicates the number of neighbours.

The alignment A_j is while the individual matches the velocity of another neighbour and is shown as follows.

$$A_j = \frac{\sum_{K=1}^{N} V_k}{N} \tag{25}$$

In Eq. (25), V_k characterizes the velocity of k-th neighbouring individuals.

The cohesion C_j refers to the tendency of an individual towards the centre and is evaluated by Eq. (26)

$$C_j = \frac{\sum_{K=1}^{N} X_k}{N} - X \tag{26}$$

The attraction F_j towards the source of food is evaluated as follows

$$F_i = X^+ - X \tag{27}$$

In Eq. (27), X^+ indicates the location of food sources.

The distraction E_i from the enemy is shown as follows

$$E_i = X^- - X \tag{28}$$

In Eq. (28), X^- indicates the enemy location.

The swarm behaviour of dragonflies is a grouping of those factors. For upgrading the location of the dragonfly, 2 vectors, step (ΔX) and location (X), are taken into account. The step vector illustrates the direction of movement as follows.

$$\Delta X_{t+1} = \left(sS_i + aA_i + cC_j + fF_i + eE_i\right) + w\Delta X_t \tag{29}$$

In Eq. (29), s, a, and c indicate the separation, alignment, and cohesion weights, whereas f and e show the food and enemy factors correspondingly. w denotes the inertial weight, and t indicates the present iteration.

Afterwards, the computation of the step vector, the location vector is evaluated by Eq. (30),

$$X_{t+1} = X_t + \Delta X_{t+1} \tag{30}$$

For exploration, searching space can be performed by a random walk (Levy Ilight) and is evaluated using Eq. (31).

$$\text{Levy}(x) = 0.01 \times \frac{r_1 \times \sigma}{|r_2|^{\frac{1}{\beta}}} \tag{31}$$

Now, r_1 and r_2 indicate the arbitrary number within 0 and 1, β indicates the constant as follows

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2\left(\frac{\beta-1}{2}\right)}\right)^{1/\beta} \tag{32}$$

where $\Gamma(x) = (x-1)!$.

The location of the dragonfly is upgraded as follows.

$$X_{t+1} = X_t + Levy(d) \times X_t \tag{33}$$

Now, d indicates the dimension of the location vector.

3 Results and Discussion

This section assesses the experimental validation of the AAA-ODBN approach utilizing a dataset [20], as depicted in Table 1. IoT-23 is a new network traffic dataset from the Internet of Things (IoT) devices. It has 20 malware captures executed in IoT devices and 3 captures for benign IoT device traffic.

The confusion matrices provided by the AAA-ODBN model under distinct TR and TS dataset is given in Fig. 3. With 80% of TR data, the AAA-ODBN approach has categorized 360 samples into M-1, 341 samples into M-2, 343 samples into M-3, 351 samples into M-4, 366 samples into M-5, 360 samples into M-6, 342 samples into M-7, 369 samples into M-8, and 372 samples into B. Simultaneously, with 20% of TS data, the AAA-ODBN system has categorized 91 samples into M-1, 99 samples into M-2, 99 samples into M-3, 93 samples into M-4, 82 samples into M-5, 94 samples into M-6, 67 samples into M-7, 75 samples into M-8, and 88 samples into B. Concurrently, with 70% of TR data, the AAA-ODBN technique has categorized 278 samples into M-1, 287 samples into M-2, 292 samples into M-3, 308 samples into M-4, 280 samples into M-5, 298 samples into M-6, 283 samples into M-7, 308 samples into M-8, and 318 samples into B. Finally, with 30% of TS data, the AAA-ODBN methodology has categorized 116 samples into M-1, 1230 samples into M-2, 123 samples into M-3, 122 samples into M-4, 124 samples into M-5, 128 samples into M-6, 140 samples into M-7, 132 samples into M-8, and 144 samples into B.

Table 1: Dataset details

Label	Description	No. of samples
M-1	CC	500
M-2	FileDownload	500
M-3	HeartBeat	500
M-4	PartofHorizontalPortScan	500
M-5	Torii	500
M-6	Okiru	500
M-7	Mirai	500
M-8	DDos	500
B	Benign	500
Total number of samples		4500

Table 2 provides an overall result analysis of the AAA-ODBN model on 80% of TR data and 20% of TS data.

Fig. 4 illustrates a brief ransomware detection outcome of the AAA-ODBN approach on 80% of TR data. These results ensured that the AAA-ODBN model has improved under all classes. For the sample, in the M-1 class, the AAA-ODBN technique has an accessible $accu_y$ of 97.53%, $prec_n$ of 88.24%, $sens_y$ of 89.78%, $spec_y$ of 98.50%, and F_{score} of 89%. Also, in M-2 class, the AAA-ODBN approach has obtainable $accu_y$ of 97.47%, $prec_n$ of 87.89%, $sens_y$ of 88.57%, $spec_y$ of 98.54%, and F_{score} of 88.23%. Besides, in M-3 class, the AAA-ODBN system has offered $accu_y$ of 97.44%, $prec_n$ of 88.86%, $sens_y$ of 87.50%, $spec_y$ of 98.66%, and F_{score} of 88.17%. Moreover, in M-4 class, the AAA-ODBN algorithm has obtainable $accu_y$ of 97.69%, $prec_n$ of 89.77%, $sens_y$ of 89.09%, $spec_y$ of 98.75%, and F_{score} of 89.43%.

Fig. 5 depicts a brief ransomware detection result of the AAA-ODBN technique on 20% of TS data. These outcomes ensured that the AAA-ODBN technique had demonstrated enhanced results under all classes. For sample, in the M-1 class, the AAA-ODBN methodology has an accessible $accu_y$ of 97.22%, $prec_n$ of 84.26%, $sens_y$ of 91.92%, $spec_y$ of 97.88%, and F_{score} of 87.92%. Besides, on M-2 class, the AAA-ODBN methodology has obtainable $accu_y$ of 96.89%, $prec_n$ of 89.19%, $sens_y$ of 86.09%, $spec_y$ of 98.47%, and F_{score} of 87.61%. Besides, in M-3 class, the AAA-ODBN technique has an accessible $accu_y$ of 97.44%, $prec_n$ of 87.61%, $sens_y$ of 91.67%, $spec_y$ of 98.23%, and F_{score} of 89.59%. Finally, in M-4 class, the AAA-ODBN technique has presented $accu_y$ of 98.11%, $prec_n$ of 95.88%, $sens_y$ of 87.74%, $spec_y$ of 99.50%, and F_{score} of 91.63%.

Table 3 provides an overall result investigation of the AAA-ODBN technique on 70% of TR data and 30% of TS data. Fig. 6 showcases a detailed ransomware detection outcome of the AAA-ODBN technique on 70% of TR data. These outcomes indicated that the AAA-ODBN approach had enhanced outcomes in all classes. For sample, in the M-1 class, the AAA-ODBN system has an existing $accu_y$ of 96.29%, $prec_n$ of 86.88%, $sens_y$ of 78.75%, $spec_y$ of 98.50%, and F_{score} of 82.62%. Moreover, in M-2 class, the AAA-ODBN model has offered $accu_y$ of 96.38%, $prec_n$ of 84.91%, $sens_y$ of 82%, $spec_y$ of 98.18%, and F_{score}

of 83.43%. Besides, in M-3 class, the AAA-ODBN technique has presented $accu_y$ of 96.19%, $prec_n$ of 82.02%, $sens_y$ of 83.91%, $spec_y$ of 97.72%, and F_{score} of 82.95%. Ay last, in M-4 class, the AAA-ODBN system has offered $accu_y$ of 96.35%, $prec_n$ of 83.70%, $sens_y$ of 84.85%, $spec_y$ of 97.85%, and F_{score} of 84.27%.

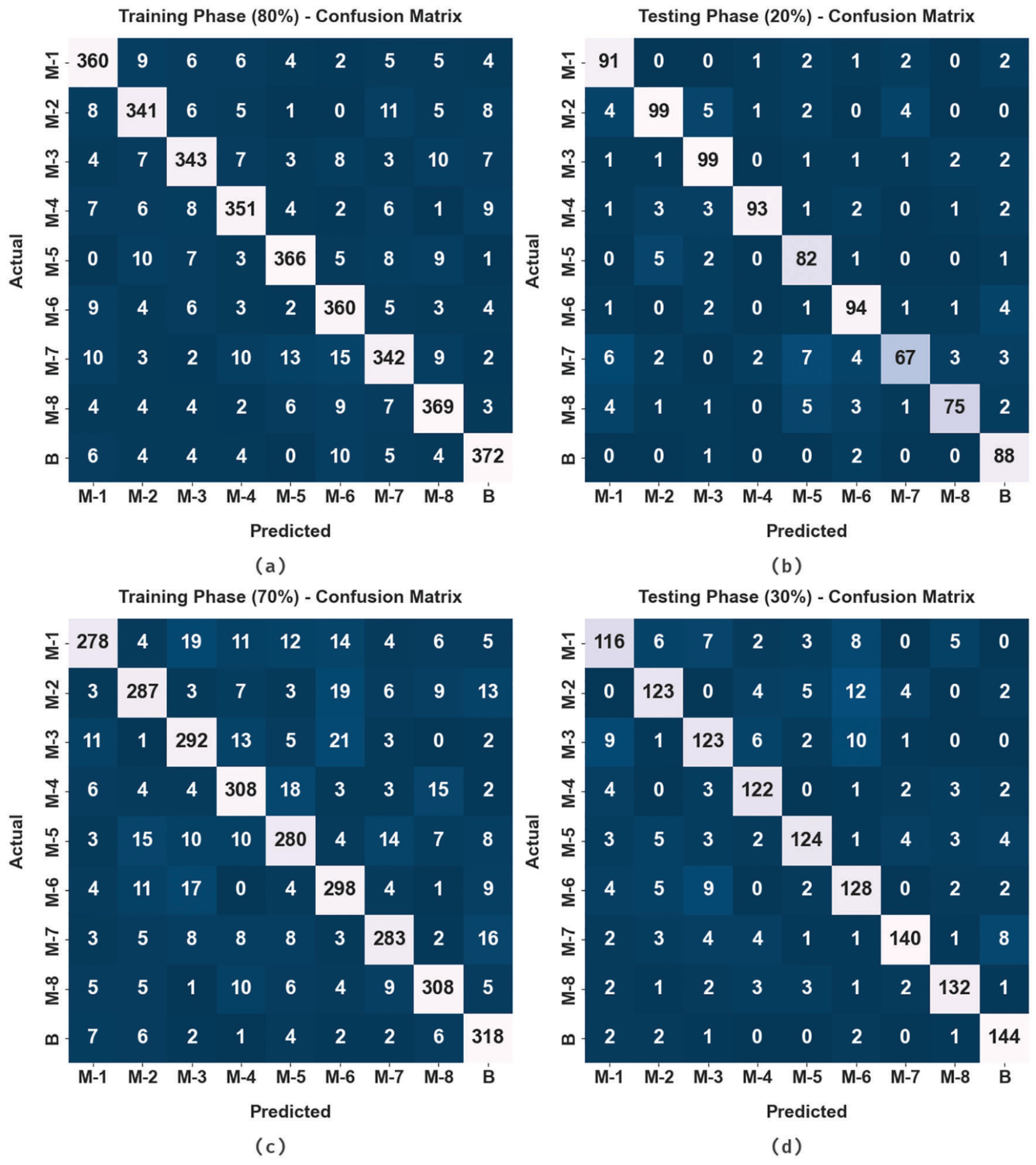

Figure 3: Confusion matrices of AAA-ODBN approach (a) 80% of TR dataset, (b) 20% of TS dataset, (c) 70% of TR dataset, and (d) 30% of TS dataset

Table 2: Result analysis of AAA-ODBN approach with distinct measures under 80:20 of TR and TS data

Labels	Accuracy	Precision	Sensitivity	Specificity	F-score
Training phase (80%)					
M-1	97.53	88.24	89.78	98.50	89.00
M-2	97.47	87.89	88.57	98.54	88.23
M-3	97.44	88.86	87.50	98.66	88.17
M-4	97.69	89.77	89.09	98.75	89.43
M-5	97.89	91.73	89.49	98.97	90.59
M-6	97.58	87.59	90.91	98.41	89.22
M-7	96.83	87.24	84.24	98.43	85.71
M-8	97.64	88.92	90.44	98.56	89.67
B	97.92	90.73	90.95	98.81	90.84
Average	**97.56**	**89.00**	**89.00**	**98.63**	**88.99**
Testing phase (20%)					
M-1	97.22	84.26	91.92	97.88	87.92
M-2	96.89	89.19	86.09	98.47	87.61
M-3	97.44	87.61	91.67	98.23	89.59
M-4	98.11	95.88	87.74	99.50	91.63
M-5	96.89	81.19	90.11	97.65	85.42
M-6	97.33	87.04	90.38	98.24	88.68
M-7	96.00	88.16	71.28	98.88	78.82
M-8	97.33	91.46	81.52	99.13	86.21
B	97.89	84.62	96.70	98.02	90.26
Average	**97.23**	**87.71**	**87.49**	**98.45**	**87.35**

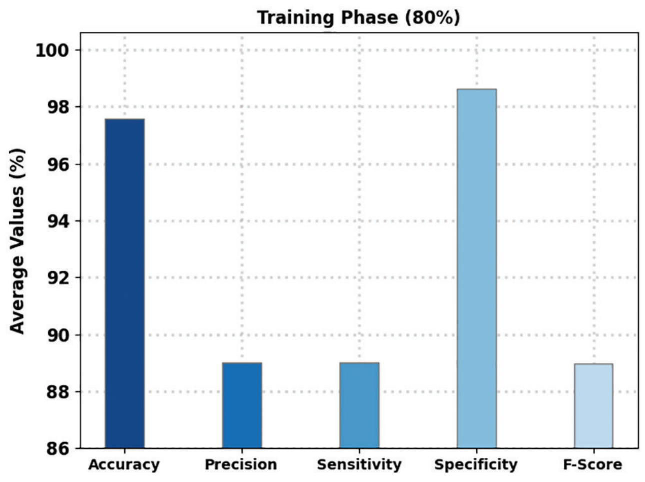

Figure 4: Average analysis of AAA-ODBN algorithm under 80% of TR data

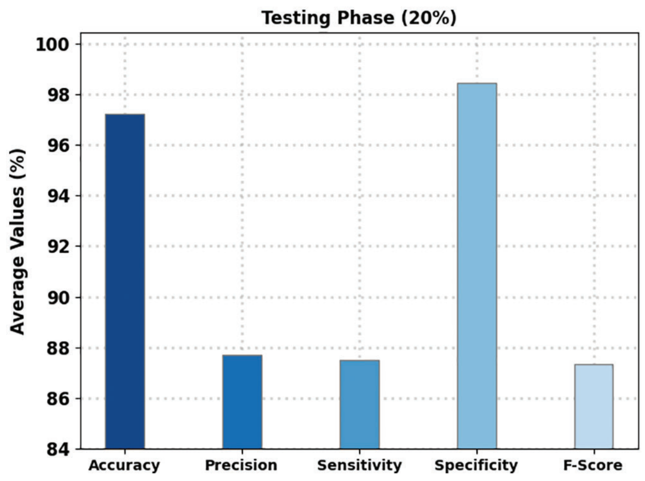

Figure 5: Average analysis of AAA-ODBN algorithm under 20% of TS data

Table 3: Result analysis of AAA-ODBN approach with distinct measures under 70:30 of TR and TS data

Labels	Accuracy	Precision	Sensitivity	Specificity	F-Score
Training phase (70%)					
M-1	96.29	86.88	78.75	98.50	82.62
M-2	96.38	84.91	82.00	98.18	83.43
M-3	96.19	82.02	83.91	97.72	82.95
M-4	96.35	83.70	84.85	97.85	84.27
M-5	95.84	82.35	79.77	97.86	81.04
M-6	96.19	80.98	85.63	97.50	83.24
M-7	96.89	86.28	84.23	98.40	85.24
M-8	97.11	87.01	87.25	98.36	87.13
B	97.14	84.13	91.38	97.86	87.60
Average	**96.49**	**84.25**	**84.20**	**98.02**	**84.17**
Testing phase (30%)					
M-1	95.78	81.69	78.91	97.84	80.28
M-2	96.30	84.25	82.00	98.08	83.11
M-3	95.70	80.92	80.92	97.58	80.92
M-4	97.33	85.31	89.05	98.27	87.14
M-5	96.96	88.57	83.22	98.67	85.81
M-6	95.56	78.05	84.21	96.99	81.01
M-7	97.26	91.50	85.37	98.90	88.33
M-8	97.78	89.80	89.80	98.75	89.80
B	98.00	88.34	94.74	98.41	91.43
Average	**96.74**	**85.38**	**85.36**	**98.17**	**85.31**

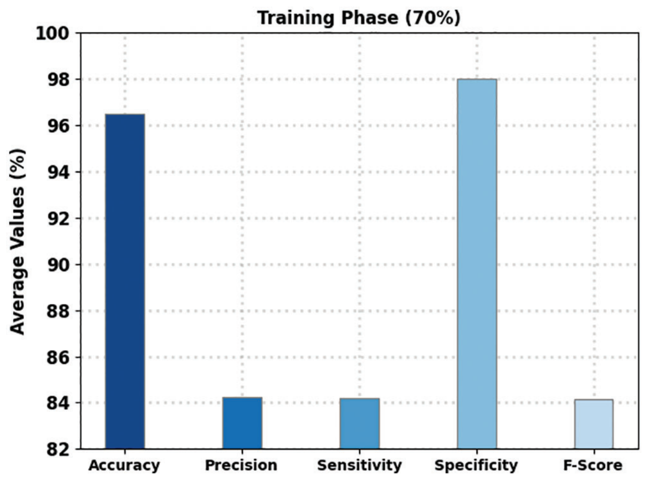

Figure 6: Average analysis of AAA-ODBN algorithm under 70% of TR data

Fig. 7 defines a brief ransomware detection outcome of the AAA-ODBN methodology on 30% of TS data. These outcomes ensured that the AAA-ODBN technique had outperformed improved outcomes under all classes. For sample, in the M-1 class, the AAA-ODBN model has obtainable $accu_y$ of 95.78%, $prec_n$ of 81.69%, $sens_y$ of 78.91%, $spec_y$ of 97.84%, and F_{score} of 80.28%. Also, in M-2 class, the AAA-ODBN system has obtainable $accu_y$ of 96.30%, $prec_n$ of 84.25%, $sens_y$ of 82%, $spec_y$ of 98.08%, and F_{score} of 83.11%. Besides, in M-3 class, the AAA-ODBN technique has presented $accu_y$ of 95.70%, $prec_n$ of 80.92%, $sens_y$ of 80.92%, $spec_y$ of 97.58%, and F_{score} of 80.92%. Eventually, in M-4 class, the AAA-ODBN approach has an accessible $accu_y$ of 87.33%, $prec_n$ of 85.31%, $sens_y$ of 89.05%, $spec_y$ of 98.27%, and F_{score} of 87.14%.

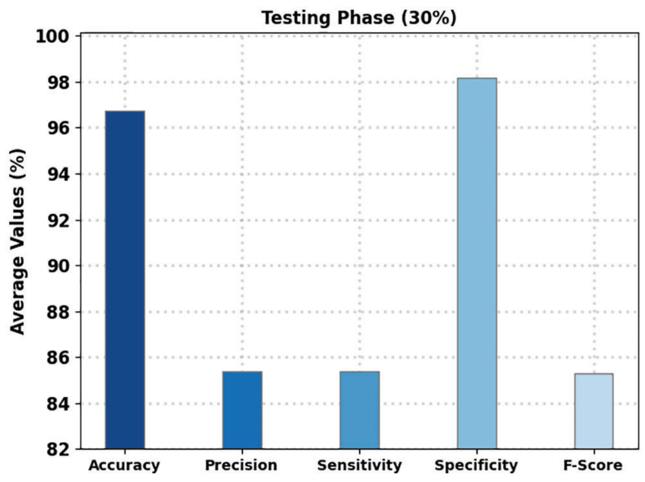

Figure 7: Average analysis of AAA-ODBN algorithm under 30% of TS data

The training accuracy (TRA) and validation accuracy (VLA) acquired by the AAA-ODBN approach under the test dataset is displayed in Fig. 8. The experimental result outperformed that the AAA-ODBN method has reached enhanced values of TRA and VLA. In specific, the VLA looked that superior to TRA.

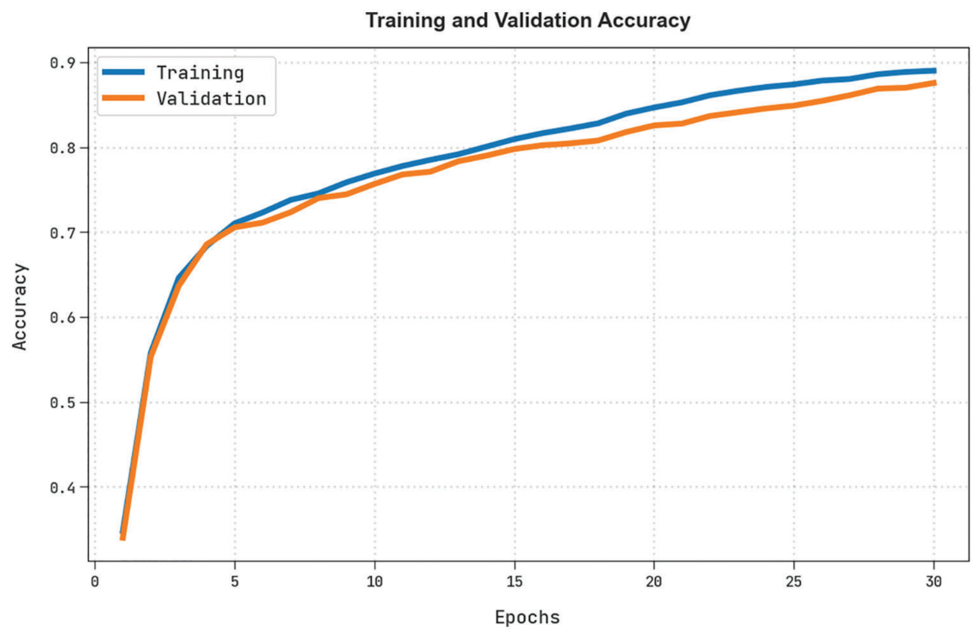

Figure 8: TRA and VLA analysis of AAA-ODBN algorithm

The training loss (TRL) and validation loss (VLL) realized by the AAA-ODBN methodology under the test dataset are shown in Fig. 9. The experimental result stated that the AAA-ODBN approach had decreased values of TRL and VLL. In certain, the VLL is lesser than TRL.

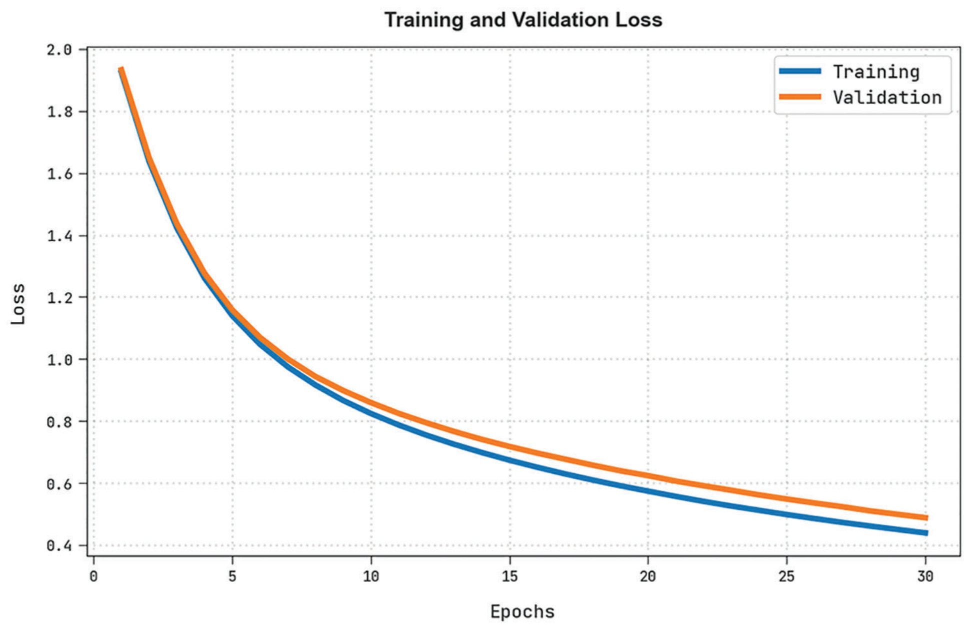

Figure 9: TRL and VLL analysis of AAA-ODBN algorithm

An obvious precision-recall investigation of the AAA-ODBN methodology under the test dataset is depicted in Fig. 10. The figure states that the AAA-ODBN technique has improved precision-recall values under distinct class value labels.

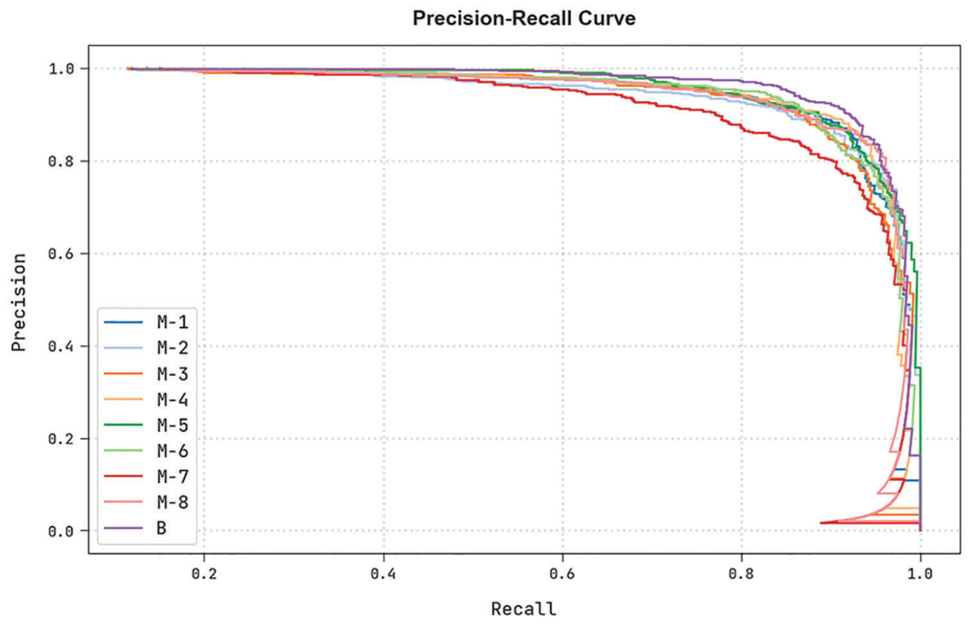

Figure 10: Precision-recall analysis of AAA-ODBN algorithm

A detailed ROC analysis of the AAA-ODBN system under the test dataset is displayed in Fig. 11. The outcome exhibited by the AAA-ODBN technique demonstrated its ability to classify various class labels.

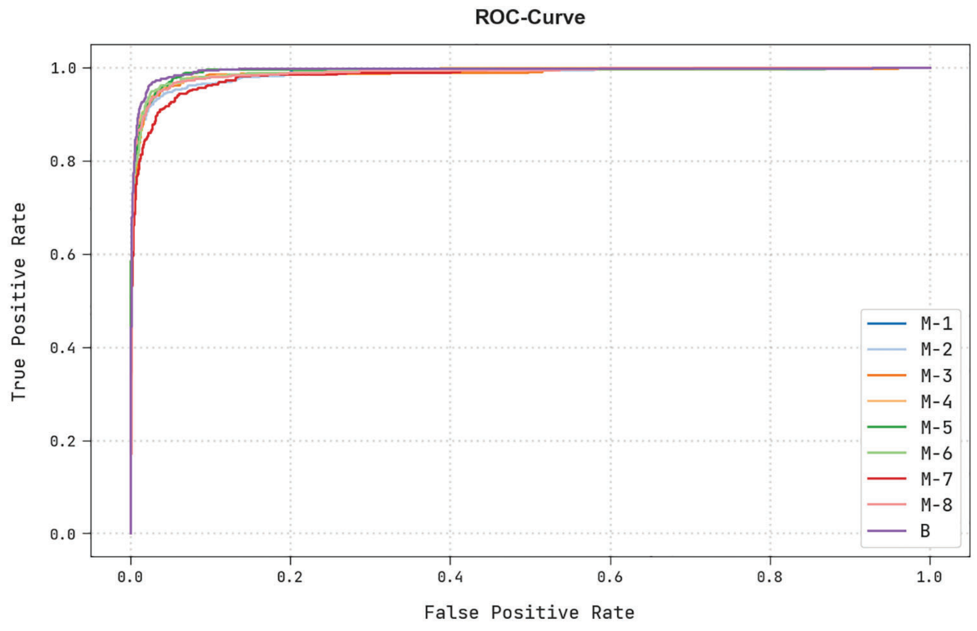

Figure 11: ROC curve analysis of AAA-ODBN algorithm

At last, a comparison analysis of the AAA-ODBN approach with existing methodologies in terms of $accu_y$ is shown in Table 4 and Fig. 12 [21]. The obtaine0d values indicate that the LSTM-BRNN model has attained a minimal $accu_y$ of 76.40%. In addition, the LSTM-BNN and CNN models have obtained slightly improved $accu_y$ values of 83.30% and 88.58%, respectively. Next, the CNN-LSTM, SGD, and DRNN models have reached a reasonable $accu_y$ of 96.27%, 95.34%, and 94.67%, respectively. Eventually, the projected AAA-ODBN technique has shown an enhanced $accu_y$ of 97.56%. These results assured the better performance of the AAA-ODBN technique over other DL models.

Table 4: Comparative analysis of AAA-ODBN technique with existing methodologies

Methods	Accuracy (%)
AAA-ODBN	97.56
LSTM-BRNN	73.40
LSTM-BNN	83.30
CNN model	88.58
CNN-LSTM	96.27
SGD model	95.34
DRNN model	94.67

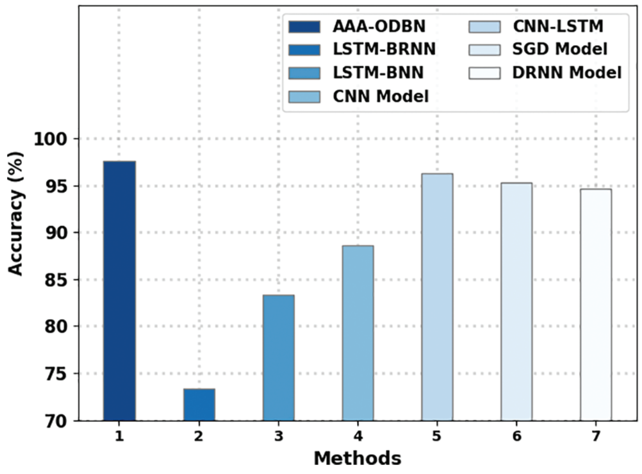

Figure 12: Comparative analysis of AAA-ODBN approach with existing methodologies

4 Conclusion

In this study, a novel AAA-ODBN technique has been developed for ransomware detection in the IoT environment. The presented AAA-ODBN technique mainly intends to recognize and categorize ransomware in the IoT environment. The presented AAA-ODBN technique follows a three-stage process: feature selection, classification, and parameter tuning. In the first stage, the AAA-ODBN exploited the AAA-FS technique to elect feature subsets. Secondly, the AAA-ODBN technique employs the DBN model for ransomware detection. At last, the DFA is utilized for the hyperparameter tuning of the DBN approach. A sequence of simulations is executed to demonstrate the improved performance of the AAA-ODBN system. The experimental values indicate the significant outcomes of the AAA-ODBN model over other models. As a part of the future scope, feature reduction approaches can improve the outcome of the AAA-ODBN model.

Funding Statement: This work was funded by the Deanship of Scientific Research at Princess Nourah bint Abdulrahman University, through the Research Groups Program Grant no. (RGP-1443-0048).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] D. W. Fernando, N. Komninos and T. Chen, "A study on the evolution of ransomware detection using machine learning and deep learning techniques," *IoT*, vol. 1, no. 2, pp. 551–604, 2020.

[2] U. Urooj, B. A. S. Al-rimy, A. Zainal, F. A. Ghaleb and M. A. Rassam, "Ransomware detection using the dynamic analysis and machine learning: A survey and research directions," *Applied Sciences*, vol. 12, no. 1, pp. 172, 2021.

[3] R. Damaševičius, A. Venčkauskas, J. Toldinas and Š. Grigaliūnas, "Ensemble-based classification using neural networks and machine learning models for windows PE malware detection," *Electronics*, vol. 10, no. 4, pp. 485, 2021.

[4] C. W. Tien, S. W. Chen, T. Ban and S. Y. Kuo, "Machine learning framework to analyze IoT malware using elf and opcode features," *Digital Threats: Research and Practice*, vol. 1, no. 1, pp. 1–19, 2020.

[5] S. I. Bae, G. B. Lee and E. G. Im, "Ransomware detection using machine learning algorithms," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 18, pp. e5422, 2020.

[6] S. Sharma, C. R. Krishna and R. Kumar, "Android ransomware detection using machine learning techniques: A comparative analysis on GPU and CPU," in *21st Int. Arab Conf. on Information Technology (ACIT)*, Giza, Egypt, pp. 1–6, 2020.

[7] Y. Dion and S. N. Brohi, "An experimental study to evaluate the performance of machine learning algorithms in ransomware detection," *Journal of Engineering Science and Technology*, vol. 15, no. 2, pp. 967–981, 2020.

[8] F. Noorbehbahani, F. Rasouli and M. Saberi, "Analysis of machine learning techniques for ransomware detection," in *16th Int. ISC (Iranian Society of Cryptology) Conf. on Information Security and Cryptology (ISCISC)*, Mashhad, Iran, pp. 128–133, 2019.

[9] M. Asam, S. J. Hussain, M. Mohatram, S. H. Khan, T. Jamal *et al.,* "Detection of exceptional malware variants using deep boosted feature spaces and machine learning," *Applied Sciences*, vol. 11, no. 21, pp. 10464, 2021.

[10] M. Dib, S. Torabi, E. Bou-Harb and C. Assi, "A multi-dimensional deep learning framework for iot malware classification and family attribution," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1165–1177, 2021.

[11] J. Hwang, J. Kim, S. Lee and K. Kim, "Two-stage ransomware detection using dynamic analysis and machine learning techniques," *Wireless Personal Communications*, vol. 112, no. 4, pp. 2597–2609, 2020.

[12] S. Baek, J. Jeon, B. Jeong and Y. S. Jeong, "Two-stage hybrid malware detection using deep learning," *Human-Centric Computing and Information Sciences*, vol. 11, no. 27, pp. 10–22967, 2021.

[13] M. Basnet, S. Poudyal, M. H. Ali and D. Dasgupta, "Ransomware detection using deep learning in the SCADA system of electric vehicle charging station," in *IEEE PES Innovative Smart Grid Technologies Conf.-Latin America (ISGT Latin America)*, Lima, Peru, pp. 1–5, 2021.

[14] M. Nisa, J. H. Shah, S. Kanwal, M. Raza, M. A. Khan *et al.,* "Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features," *Applied Sciences*, vol. 10, no. 14, pp. 4966, 2020.

[15] R. Agrawal, J. W. Stokes, K. Selvaraj and M. Marinescu, "Attention in recurrent neural networks for ransomware detection," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, pp. 3222–3226, 2019.

[16] A. Azmoodeh, A. Dehghantanha and K. K. R. Choo, "Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 1, pp. 88–95, 2018.

[17] A. Abugabah, A. A. AlZubi, M. Al-Maitah and A. Alarifi, "Brain epilepsy seizure detection using bio-inspired krill herd and artificial alga optimized neural network approaches," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 3, pp. 3317–3328, 2021.

[18] W. Deng, H. Liu, J. Xu, H. Zhao and Y. Song, "An improved quantum-inspired differential evolution algorithm for deep belief network," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 10, pp. 7319–7327, 2020.

[19] E. Yousefpoor, H. Barati and A. Barati, "A hierarchical secure data aggregation method using the dragonfly algorithm in wireless sensor networks," *Peer-to-Peer Networking and Applications*, vol. 14, no. 4, pp. 1917–1942, 2021.

[20] A. Parmisano, S. Garcia and M. J. Erquiaga, "A labeled dataset with malicious and benign IoT network traffic, https://www.stratosphereips.org/datasets-iot23.

[21] A. Sahu, S. Sharma, M. Tanveer and R. Raja, "Internet of Things attack detection using hybrid deep learning model," *Computer Communications*, vol. 176, no. 3, pp. 146–154, 2021.