Journal of Cyberspace Studies
eISSN: 2588-5502
http://jcss.ut.ac.ir
University of Tehran

An Arctic Puffin optimization with SCA approach, enhanced by a random neural network model for detecting attacks on the Internet of Things

Mohammad Arefi[1*], Parisa Rahmani[2], Hamid Shokrzadeh[2]

1. Department of Computer Engineering, ST.C., Islamic Azad University, Tehran, Iran. (*Corresponding author: St_m_arefi@azad.ac.ir, https://orcid.org/0009-0008-2970-0859)
2. Department of Computer Engineering, Par.C., Islamic Azad University, Tehran, Iran.

Article Info

Original article

Main Object: Computer Science & Technology, Artificial Intelligence, Computer Networks

Received: 10 May 2025
Revised: 10 September 2025
Accepted: 13 September 2025
Published online: 20 September 2025

Keywords:
Intrusion Detection System (IDS),
IoT,
machine learning algorithm,
meta-heuristic algorithms,
network security,
Sine-Cosine Algorithm (SCA).

Abstract

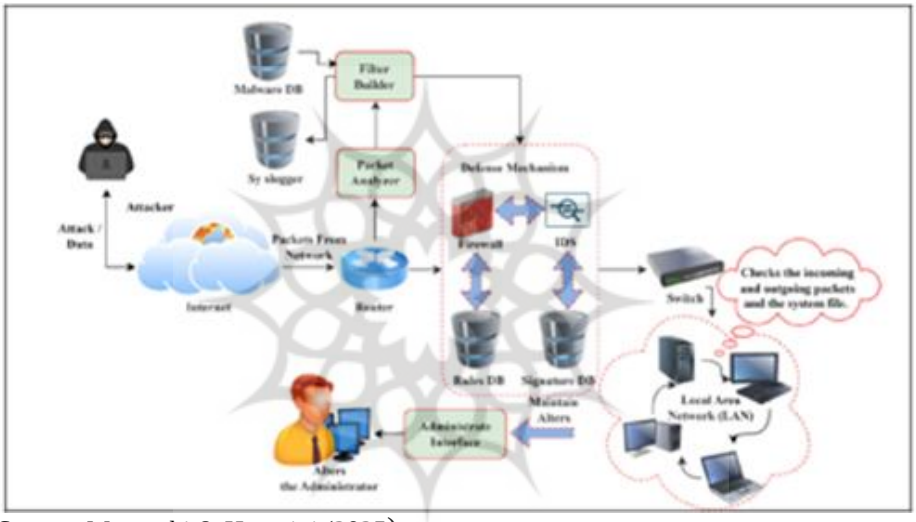
Background: Network security and penetration pose a significant challenge in the extensive IoT research of recent years. System security and user privacy demand security solutions that are carefully planned and diligently maintained.
Aims: This paper introduces a novel three-stage hybrid IDS, IoT-APOSCA, leveraging machine learning and meta-heuristics for attack detection; stages include pre-processing, feature selection, and attack detection. The pre-processing steps are: cleaning, visualization, feature engineering, and vectorization.
Methodology: Networks use Intrusion Detection Systems (IDSs) to monitor and detect malicious activities as a key security feature. The Arctic Puffin Optimization (APO) and Sine-Cosine Algorithm (SCA) are used in the feature selection stage, while a changed Random Neural Network (RNN) is employed in the attack detection stage.
Results: The proposed technique is assessed using the DS2OS dataset, and the outcomes show that the approach, integrating multiple learning models, led to an accuracy enhancement to 99.66%. Also, the values Recall and False Alarm Rate obtained are equal to 0.9926 and 0.003, respectively.
Conclusion: Intrusion detection system efficacy is directly tied to the quality of its classification method. Enhanced neural network performance is achievable through adjustments to parameters, such as network weights.

Cite this article: Aref M, Rahmani P, Shokrzadeh H. (2026). "An Arctic Puffin optimization with SCA approach, enhanced by a random neural network model for detecting attacks on the Internet of Things". *Cyberspace Studies*. 10(1): 61-80. doi: https://doi.org/10.22059/jcss.2025.395014.1146.

Website: https://jcss.ut.ac.ir/| Email: jcss@ut.ac.ir |
EISSN: 2588-5502
Publisher: University of Tehran

1. Introduction

A network of sensor-equipped devices, the Internet of Things (IoT), monitors areas and sends the collected data to users. Various applications make use of the IoT, such as COVID-19, healthcare, environmental monitoring, smart buildings, and beyond. In 2020, Gartner estimated that 6 billion devices were in the global IoT market. McKinsey forecasts that IoT will have a value of about 11 trillion dollars by 2025. Furthermore, the emergence of 6G wireless communication and mobile edge computing can aid in the growth of IoT devices (Chen et al., 2022). The workings of the IDS detection process are shown in Figure 1. IoT expansion results from merging sensors with wireless communication. Wireless sensor nodes are essential to the structure of IoT networks (Maazaahi & Hosseini, 2025).

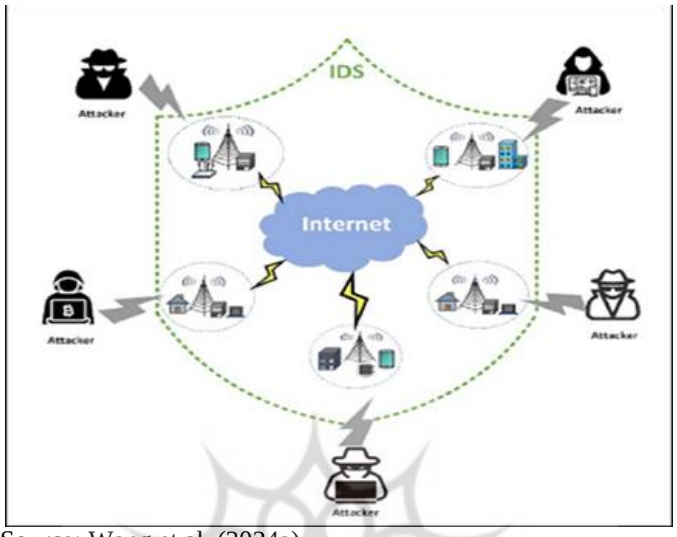


Source: Maazaahi & Hosseini (2025)

Figure 1. The process of detecting attacks in IDS

IoT network security and performance improvements are vital. These attacks can harm or weaken transmitted packets, thus compromising the system's overall efficiency (Khan et al., 2021; Rajabi et al., 2024). Ensuring complete IoT network security today requires Intrusion Detection Systems (IDS). These systems are vital for detecting and responding to intrusions that bypass preventative measures, requiring highly scalable solutions to address this challenge (Latif et al., 2020). By analyzing network traffic, Intrusion Detection Systems in cybersecurity help prevent and detect malicious attacks. Figure 2 shows a cyberattack compromising the Internet of Things. Consequently, a significant focus has become effective cyber threat management (Wang et al., 2024a). An efficient network intrusion detection model is critical for maintaining data security. Therefore, creating an effective network intrusion detection model requires an

understanding of the architecture of IoT systems, which comprise diverse components such as IoT devices, computational servers, internet devices, and wireless devices.



Source: Wang et al. (2024a)

Figure 2. Illustrates various scenarios of Internet of Things Cyber Attacks

By continuously scanning network and host traffic for suspicious activity, IDS technology safeguards against security policy breaches and protects data confidentiality and integrity. For Machine Learning (ML)-based IDSs (Anwar et al., 2024), relevant network information extraction hinges on practical feature engineering. This paper proposes Arctic Puffin Optimization (APO) (Wang et al., 2024b), a meta-heuristic algorithm, combined with a Random Neural Network (RNN) attack detection method for IoT networks (IoT-APOSCA) to solve the stated problems. To improve a random neural network's architecture and hyperparameters, the proposed model employs Arctic Puffin Optimization. The paper's main innovations can be described as follows:

- A novel hybrid algorithm is proposed in this paper to improve algorithm search, speed, and accuracy;
- The random neural network was enhanced using Arctic Puffin Optimization and SCA;
- This paper employs Arctic puffin optimization alongside SCA in training standard random neural networks; This approach helps avoid premature convergence toward local minima of the squared error function.

The rest of the paper is organized as follows: Section 2 provides a summary of relevant research. The optimization of Arctic puffin and SCA is discussed in Section 3. Section 4 describes the proposed

algorithm and methodology. Section 5 shows the implementation and evaluation of the proposed model. Finally, in Section 6, we present the conclusion and future work.

2. Related works

Research on IoT attack detection systems, employing both traditional machine learning and deep learning/meta-heuristic approaches, is the focus of this section. This study compares the core principles, advantages, and disadvantages of the three approaches to offer a complete perspective to researchers and practitioners. Attack detection frequently uses Machine Learning (ML), which has shown effectiveness. Attack detection systems frequently use Support Vector Machines (SVMs), Naïve Bayes, Decision Trees (DTs), Random Forests (RFs), and Artificial Neural Networks (ANNs).

2.1. Machine learning based attack detection methods

Wang et al. (2024c) presented ET-DCANET, a highly efficient hierarchical intrusion detection model. This model employed the extreme random tree algorithm to pick the best feature subset painstakingly. Zhong et al. (2024) introduced a novel intrusion detection system that uses blockchain, federated learning, and an ANN for key exchange. Kumar et al. (2023) proposed a homomorphic encryption scheme using a privacy chain, evidence weight, and the information value from a statistical transformation method. This system ensures user privacy and security.

2.2. Deep learning based attack detection methods

The rapid progress of deep learning technologies in recent years has led to the widespread use of advanced deep learning algorithms in attack detection. Including: DNNs, CNNs, RNNs, and Deep Convolutional GANs, GAN. Wang and Yang (2024) introduced a network intrusion detection model based on Deep Learning (DL). This model sought to boost detection accuracy through feature extraction from spatial and temporal aspects of network traffic data. The model aimed to amplify the minority class samples, handle data imbalance, and enhance the accuracy of network intrusion detection. Kumar et al. (2024) introduced a statistical differential privacy-deep neural network SDP algorithm to protect sensitive personal data.

2.3. Meta-Heuristic based attack detection methods

Due to reducing computational overhead and increasing classification accuracy, meta-heuristic algorithms have received more attention in recent years and have also been used in attack detection systems. Subramaniam et al. (2024) Intrusion Detection System using Hybrid Evolutionary Lion and Balancing Composite Motion Optimization Algorithm espoused feature selection with Ensemble Classifier (IDS-

IOT Hybrid ELOA-BCMOA-Ensemble-DT-LSVM-RF-XGBOOST) proposal for Securing IOT Network.

3. Preliminary concept

The main goal of this paper is to find optimal values for the weights of a Random Neural Network using the APO meta-heuristic algorithm and SCA. For this reason, basic concepts, such as the APO optimization algorithm and SCA, are discussed in this section. Swarm intelligence algorithms make up a subset of meta-heuristic algorithms. Swarm intelligence algorithms are AI methods mimicking decentralized, self-organizing group behaviors. Typically, these systems are comprised of simple agents that interact locally with each other and in their environment. Algorithms are an artificial intelligence method that relies on group behaviors in decentralized, self-organizing systems. Such systems are commonly made up of many simple agents that interact locally among themselves and their surroundings.

3.1. Arctic puffin optimization

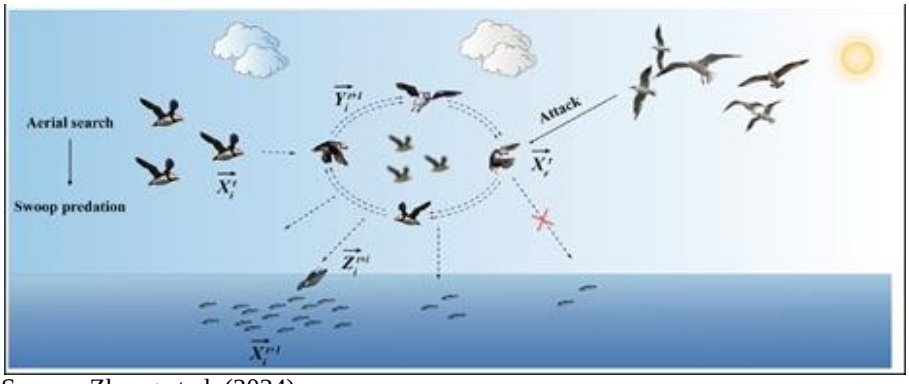
Small and rare, Arctic puffins are birds of the Arctic. Famous for their fishing skills, these ocean birds usually live together in flocks. Arctic puffins, mighty hunters, catch at least ten small fish per dive. Collaborative foraging techniques, employed pre-dive, enhance their hunting success. This section presents the APO algorithm, which is inspired by Arctic Puffin survival strategies and SCA. The APO model's framework consists of three phases: population initialization, aerial survey, and underwater resource extraction. Moreover, the puffin's behavioral transformation factor B accounts for the change between these two methods.

Initial population. Each Arctic Puffin represents a potential solution for participating in the optimization. The following equation describes the generation process of initializing the population:

$$\overrightarrow{X}_i^t = rand * (ub - lb) + lb \quad i = 1, 2, 3...N \tag{1}$$

where $\overrightarrow{X}_i^t$ represents the position of the *i*th Arctic Puffin; *rand* generates a random number between 0 and 1; *ub* and *ul* represent the upper and lower bounds, *N* is the number of individuals in the population.

Aerial flight stage (Exploration). Arctic puffins utilize specialized flight and foraging techniques to overcome the difficulties of their environment. Their daily lives require flexible adaptation between the ocean and air. The strategy above is shown in Figure 3.



Source: Zhong et al. (2024)

Figure 3. Flight stage of Arctic puffins

Arctic puffins often fly together in groups, improving their hunting and flight efficiency. When conditions are good, with few predators and lots of fish, they expertly rise to the surface to catch prey. The equations below update the position for this strategy.

$$\overrightarrow{Y_i^{t+1}} = \overrightarrow{X_i^t} + (\overrightarrow{X_i^t} - \overrightarrow{X_r^t}) * L(D) + R \quad (2)$$

$$R = round(0.5 * (0.05 + rand)) * a \quad (3)$$

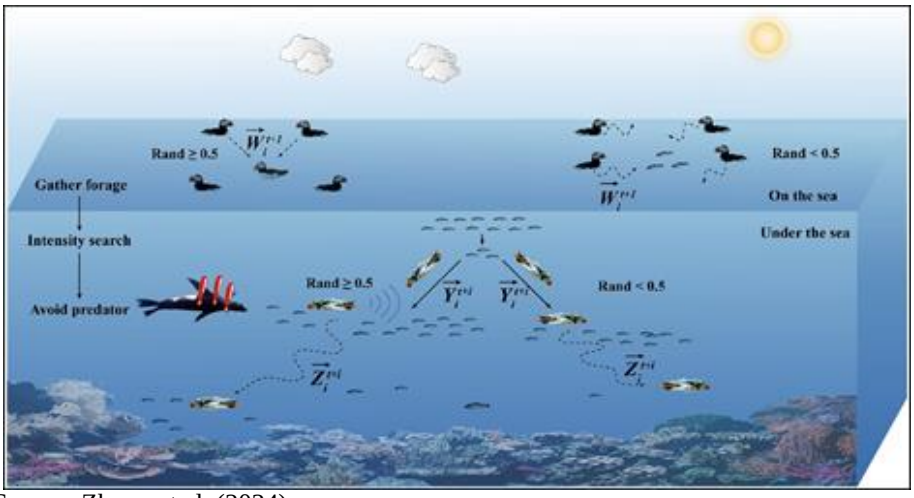
$$a \sim Normal(0,1) \quad (4)$$

where r is a random integer between 1 and *N-1*, excluding i; $\overrightarrow{X_i^t}$ represents the current ith candidate solution in the population; $\overrightarrow{X_r^t}$ is a candidate solution randomly selected from the current population, with $\overrightarrow{X_i^t} \neq \overrightarrow{X_r^t}$; *L(D)* denotes a random number D is the dimension; α is a random number which follows the standard normal distribution.

Underwater foraging stage (Exploitation). The survival strategy in the Arctic during the underwater foraging stage involves two key elements: aerial hunting and underwater exploration. Figure 4 illustrates how Arctic puffins, while on the surface, learn where to dive for food by watching others during the underwater search phase. Equation (5) describes the position update.

$$\overrightarrow{W_i^{t+1}} = \begin{cases} \overrightarrow{X_{r1}^t} + F * L(D) * (\overrightarrow{X_{r2}^t} - \overrightarrow{X_{r3}^t}) rand \geq 0.5 \\ \overrightarrow{X_{r1}^t} + F * (\overrightarrow{X_{r2}^t} - \overrightarrow{X_{r3}^t}) rand \prec 0.5 \end{cases} \quad (5)$$

where F represents the cooperative factor, adjusting the predation behavior of Arctic puffins. We consider *F=0.5* in this paper. variables r_1, r_2, r_3 are random numbers between 1 and *N-1*. $\overrightarrow{X_{r1}^t}$, $\overrightarrow{X_{r2}^t}$, $\overrightarrow{X_{r3}^t}$ are candidate solutions randomly selected from the current population, and $r_1 \neq r_2 \neq r_3$, $\overrightarrow{X_{r2}^t} \neq \overrightarrow{X_{r3}^t}$. In the APO algorithm, the parameter F is designed as a synergistic factor, inspired by the collaboration and team predation exhibited by Arctic puffins in their foraging behavior.



Source: Zhong et al. (2024)

Figure 4. Underwater foraging stage of Arctic puffins

As predation progresses, Arctic puffins may sense a depletion or exhaustion of food resources in their current foraging area after a specific period. To continue meeting their nutritional needs, they must alter their underwater positions to search for more fish or other underwater food sources. The position update equation for this stage is as follows:

$$\overrightarrow{Y_i^{t+1}} = \overrightarrow{W_i^{t+1}} * (1 + f) \tag{6}$$

$$f = 0.1 * (rand - 1) * \frac{(T - t)}{T} \tag{7}$$

where T represents the total number of iterations, and t denotes the current iteration count. $rand$ is a random number that introduces some randomness to f.

3.2. Sine-Cosine algorithm

In the following, we introduce a population-based algorithm called the Sine-Cosine Algorithm (SCA). SCA generates several initial random solutions that oscillate towards the best solution using a mathematical model based on sine and cosine functions (Mirjalili, 2016). The Sine Cosine Optimization Algorithm (SCA) has showed high extraction power and precise convergence, leading to achievement of the exact optimal point even in high-dimensional functions, not only SCA addresses both exploration and exploitation in optimization and aims to find the global optimum of the problem but also the Arctic puffins' position and behavior are improved using the parameters of the sine-cosine algorithm; Also, the Arctic puffin optimization is changed using the sine-cosine optimization algorithm, resulting in Conceptualize this robust new algorithm to achieve a better balance between the early

stages of global search and the late stages of global convergence. The parameter C of the Arctic puffin optimization was changed using the sine-cosine algorithm to solve the mentioned problem. In SCA, the best solution always represents the destination for search waves. Thus, search waves do not deviate from the primary optimum of the problem. Further, the fluctuating behavior in this algorithm allows it to search the search space around the optimum of the problem well, regardless of the difference between algorithms in random population-based optimization, with the optimization process being divided into two typical phases: exploration versus exploitation. The following position update equations for both phases are specified using Equation.

$$X_i^{t+1} = X_i^t + r_1 \times \sin(r_2) \times \left| r_3 p_i^t - X_i^t \right|$$
$$X_i^{t+1} = X_i^t + r_1 \times \cos(r_2) \times \left| r_3 p_i^t - X_{i_i}^t \right| \quad (8)$$

where X_i^t is the position of current solution in i_{th} dimension at t_{th} iteration, r_1, r_2, r_3 represent random numbers, p_i position of the destination point in i_{th} dimension, and $||$ indicates the absolute value. These two equations are combined to be used as follows.

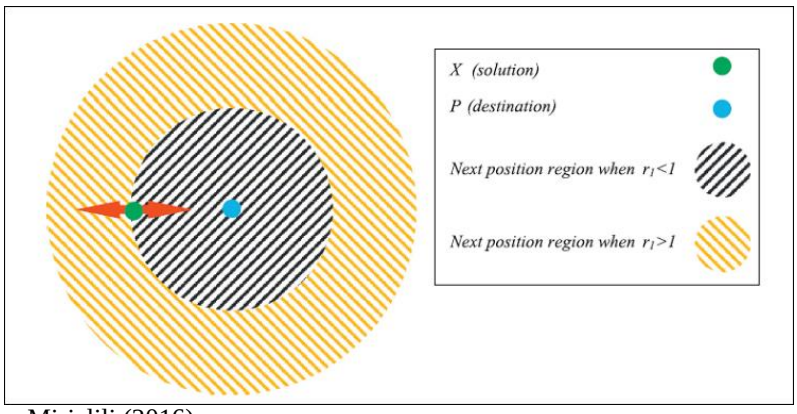
$$X_i^{t+1} = \begin{cases} X_i^t + r_1 \times \sin(r_2) \times \left| r_3 p_i^t - X_i^t \right|, r_4 < 0.5 \\ X_i^t + r_1 \times \cos(r_2) \times \left| r_3 p_i^t - X_{i_i}^t \right|, r_4 \geq 0.5 \end{cases} \quad (9)$$

where r_4 is random number in [0,1]; there are four main parameters in SCA: r_1, r_2, r_3, X_i^{t+1} denotes the new coordinates of the search wave and X_i^t represents the previous coordinates of the search wave and P_i^t shows the coordinates of the best solution found, which is considered as the destination. If r_1 less than 1, move towards the destination point, and if it is greater than 1, move away from the destination point. Figure 5 displays the effect of the r_1 parameter on the movement of waves in the sine-cosine algorithm. Equation (10) shows the effect of the r_1 parameter on the discovery and extraction of the sine-cosine algorithm. Also, the effect of the resolution parameter r_1 is shown in Figure 6.

$$r_1 = a - t\frac{a}{T} \quad (10)$$

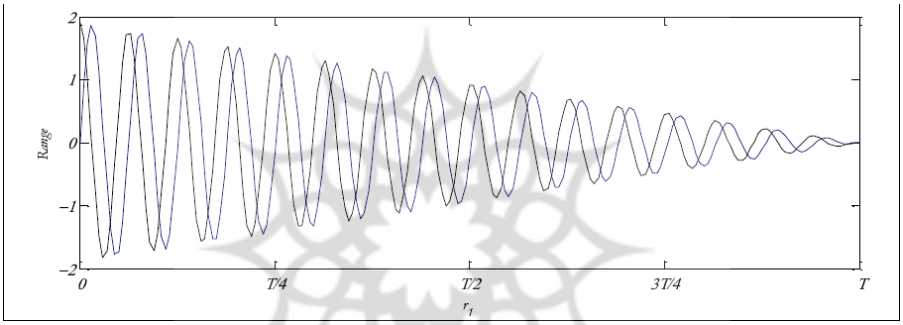
where t is the current iteration, T denotes the maximum of iterations, and a is constant.

Parameter r_2 is used to model oscillatory movement which varies within $[0,2\pi]$. Figure 7 illustrates the impact of parameter r_2 on wave movement in the sine cosine optimization algorithm. You can see the effect of the parameter r_2 clearly in Figure 7.



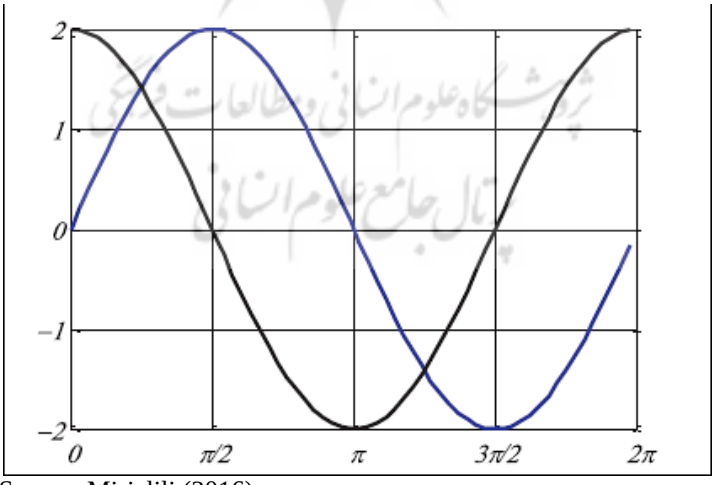
Source: Mirjalili (2016)

Figure 5. Impact of parameter $r_1$on wave movement in the sine cosine optimization algorithm



Source: Mirjalili (2016)

Figure 6. Impact of parameter r_1 on exploration and exploitation in the sine cosine optimization algorithm



Source: Mirjalili (2016)

Figure 7. Impact of parameter r_2 on wave movement in the sine cosine optimization algorithm

r_3 serves as a weight for the destination; if it is considered greater than 1, a larger step towards the destination is taken, and if it is less than 1, a smaller step is taken towards the destination. r_4 is a random variable between 0 and 1 The optimal solution to the problem is obtained by completing the final number of iterations of the algorithm and selecting the best-found solution, which is determined by a random variable between 0 and 1 that switches between sine and cosine movements. You can see the pseudocode of the sine-cosine algorithm in Figure 8.

```
Initialize a set of search agents (solutions) (X)
Do
  Evaluate each of search agents by the objective function
  Update the best solution obtained so far (P=X*)
   Update r1, r2, r3, r4
  Update the position of search agents using Equation (8)
While( t ≺ maximum number of iterations)
Return the best solution obtained so far as the global optimum
```

Source: Mirjalili (2016)

Figure 8. Pseudo codes of the sine cosine algorithm

4. Proposed method for detecting attacks in IoT

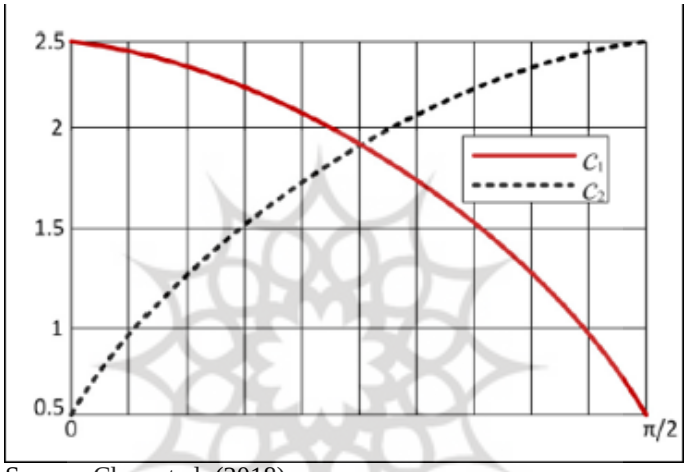
Hybrid models significantly improve network intrusion detection performance. The single feature extraction dimension of CNNs and RNNs results in suboptimal classification performance. Its applicability to diverse tasks accounts for the model's popularity. However, we find Random Neural Networks suitable for detecting attacks on IoT networks. Meanwhile, adjusting the weights in random neural networks has a direct effect on how accurately they classify. Neural networks achieve high classification accuracy through internal parameter adjustments. Our proposed framework uses meta-innovative algorithms to optimize a random neural network's architecture and internal parameters, thereby improving its classification accuracy. To improve classification accuracy, we adjust the neural network's architecture and weights accordingly. The APO algorithm designs a behavioral transition coefficient B to achieve a smooth transition from global search to local exploitation. It is defined as Equation (11).

$$B = 2 * \log(1/rand) * (1 - \frac{t}{T}) \tag{11}$$

where *rand* is a random number in (0, 1), t and T are the current iteration number and the maximum iteration number, respectively. In the Arctic puffin optimizer algorithm, a parameter C is introduced, where B is adjusted using C to determine the search strategy to be executed in the current iteration step. In this paper, the parameter C is determined by Equation (12).

$$C_1 = \theta \times \sin\left((1 - \frac{iter}{maxiter}) \times \frac{\pi}{2}\right) + \sigma$$
$$C_2 = \theta \times \cos\left((1 - \frac{iter}{maxiter}) \times \frac{\pi}{2}\right) + \sigma \quad (12)$$

where θ and σ are constants ($\Theta=2$, $\sigma=0.5$). The theoretical model of (C_1, C_2) is shown in Figure 9. The theoretical model of SCA (c_1 and c_2), The random numbers r_1 and r_2 between (0, 1) provide the movement with a direction (or region) of the current position either within the solution region and target point or outside of both. In this study, the C parameter is determined from the average of the (C_1, C_2) values.



Source: Chen et al. (2018)

Figure 9. Mathematical model display of (C_1, C_2)

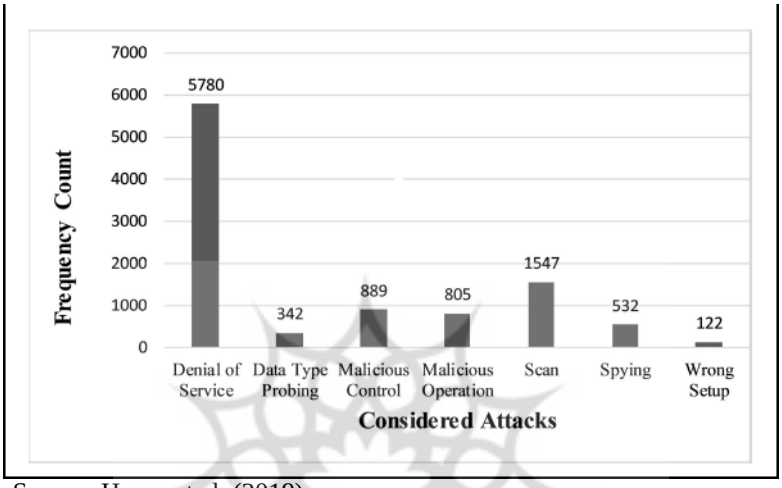
4.1. Dataset description

An open-source dataset named DS2OS was obtained from KAGGLE (Pahl & Aubet, 2018a). This dataset is one of the new generations of IoT and IIoT datasets for evaluating the fidelity and efficiency of different cybersecurity applications based on machine/deep learning algorithms. This dataset was provided by Pahl and Aubet (2018b). The dataset consists of 357952 samples and 13 features. It has 347935 normal data values and 10017 anomalous data values, with eight classes. Two features "Value" and "Accessed Node Type" have 2500 and 148 missing values, respectively. The detailed distribution of different attacks in a dataset is presented in Figure 10, and also Figure 11 shows the IoT network intrusion detection model of the metaheuristic optimization algorithm and RNN in a flow chart.

4.2. Random neural network architecture

In this section, we briefly explain the architecture of the random neural network. The technique for attack detection in IoT systems, such as the ANN and the Random Neural Network (RNN), is also inspired by the

human brain. RNN contains one input layer, eight hidden layers, and one output layer. The input layer assigns weights plus bias values and forwards these data to the hidden layers for further processing. Learning is essential in hidden layers as it plays a critical role in predicting the output from real features. Hidden layers transfer this information to the output layer for suitable output generation. After learning, the trained model is used to predict attacks by using a test set.



Source: Hasan et al. (2019)

Figure 10. Statistics of considered attacks in the DS2OS dataset

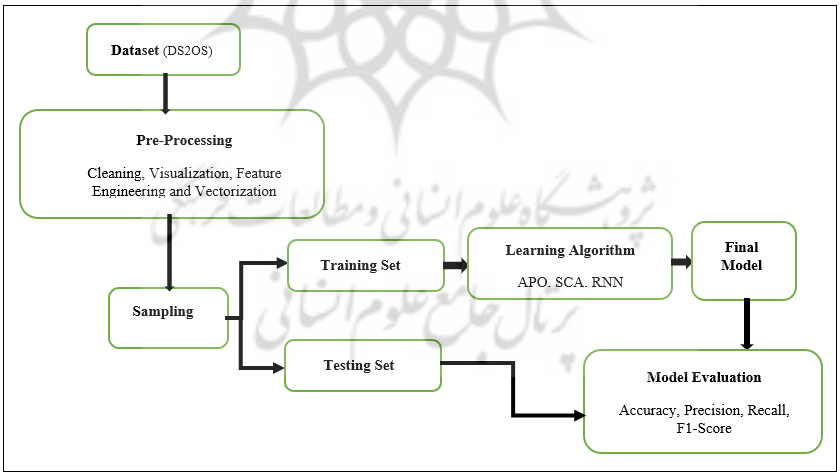


Figure 11. The IoT network intrusion detection model uses metaheuristic optimization algorithms and RNN in its structural flow chart

4.3. Adjusting the architecture of the Lightweight random neural network by using the Arctic Puffin Optimization and SCA

We know that a neural network is a black box. Still, by using the optimized parameters and architecture of the random neural network

modeled in this paper, we can increase the speed and accuracy of executing the attack detection process in the network. An important factor that affects the learning process of a neural network is the number of neurons in the hidden layers. If the number of neurons is too high, it can cause overfitting, and if the number of neurons is too low, it can cause underfitting. The correct determination of the number of neurons is essential for designing neural networks. In problems where we do not know the search space completely, using metaheuristic algorithms is efficient and valuable. Stochastic optimization algorithms are often population-based, such as the Arctic Puffin Optimization. The steps for implementing the algorithm are as follows.

Step 1. A search agent is a vector with a specified number of cells (the number of non-zero cells in the array is equivalent to the hidden layers of the network), where the number inside each cell indicates the number of neurons in that hidden layer. As shown in Figure 6, the Arctic Puffins' positions are updated using Equation (5). The goal of optimization in this phase is to minimize the number of layers of the neural network as well as the number of neurons corresponding to each layer, in order to reduce the complexity of calculations and the amount of memory consumed (Sveleba et al., 2021).

The number of neurons in the first hidden layer	The number of neurons in the second hidden layer	...	The number of neurons in the n hidden layer

Figure 12. Vector of a search agent

Step 2. Using the Arctic Puffin Optimization, the best search agent is obtained.

Step 3. The fitness function is the classification accuracy; using Equation (13), the fitness function of each search agent is determined.

$$Accuracy = \frac{T\,Pos + T\,Neg}{T\,Pos + T\,Neg + F\,Pos + F\,Neg} \quad (13)$$

where T_{Neg} is the number of records whose actual class is negative and the classification algorithm correctly identifies them as negative. T_{Pos} denotes the number of records whose actual class is positive and the classification algorithm correctly identifies them as positive. F_{Pos} shows the number of records whose actual class is negative but the classification algorithm incorrectly identifies them as positive. Finally, F_{Neg} reflects the number of records whose actual class is positive but the classification algorithm incorrectly identifies them as negative.

Step 4. A search agent is a vector with a specified number of cells, where the number inside each cell represents the weight of each. Figure 13 shows the Structure of a search agent.

Number of weight number 1	Number of weight number 2	…	Number of weight number n_{th}

Figure 13. Structure of a search agent

Step 5. Using the Arctic Puffin Optimization, the best search agent is obtained.

Step 6. Each search agent can estimate the weights of the network according to the fitness function. As mentioned, the fitness function is the classification accuracy, which is obtained using Equation (13).

5. Results of evaluations

In this section, the software and hardware implementation of the proposed scheme are described in detail. We make a comprehensive evaluation of our proposed scheme in comparison with the new typical schemes through simulation. After detailing the simulation settings, protocol comparisons, and metric evaluations, the simulation results are presented along with their analysis. The performance evaluation was conducted using the MATLAB R2024b software. These tests were performed on a 12-core central processing unit with 16 GB of RAM (ASUS TUF GAMING F15).

5.1. Metrics for evaluation

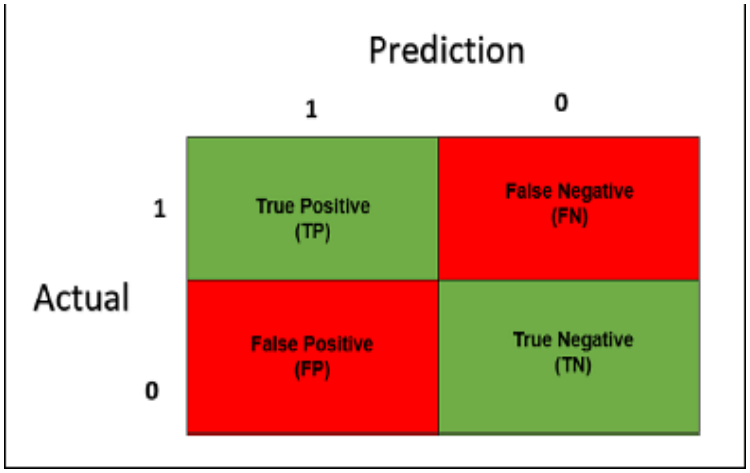
The Confusion Matrix (CM) (Zhong et al., 2024) was used to assess, analyze, and confirm the proposed detection technique. Various factors are taken into account when evaluating the proposed model. In the following, the performance parameters that are used to evaluate the proposed algorithm are briefly explained. We evaluated the suggested model by considering various performance metrics like accuracy, precision, recall, F1–score, and False Alarm Rate (FAR). The details of the Confusion matrix are clearly shown in Figure 14.

TN: The number of records whose actual class is negative and the classification algorithm correctly identifies them as negative.

TP: The number of records whose actual class is positive and the classification algorithm correctly identifies them as positive.

FP: The number of records whose actual class is negative, but the classification algorithm incorrectly identifies them as positive.

FN: The number of records whose actual class is positive but the classification algorithm incorrectly identifies them as negative.



Source: Zhong et al. (2024)

Figure 14. Confusion matrix

Accuracy is mathematically described as the ratio between accurate positive and negative results to complete the results of the machine learning model.

$$Accuracy = \frac{T\,Pos + T\,Neg}{T\,Pos + T\,Neg + F\,Pos + F\,Neg} \quad (14)$$

It is a ratio between truly predicted positive results to true and false-positive results and is mathematically described.

$$Precision = \frac{T_{Pos}}{T_{Pos} + F_{Pos}} \quad (15)$$

It describes the relationship between true positive predictions to true positive and false negative predictions.

$$Recall = \frac{T_{Pos}}{T_{Pos} + F_{Neg}} \quad (16)$$

F1 SCORE is a weighted average of precision and recall. The F1 score maintains the balance between precision and recall by considering positive and negative results.

$$F1 - Sccore = \frac{2 \times (Precision + Recall)}{Precision + Recall} \quad (17)$$

False alarm rate (FAR) means the false rate of the detection system. Indeed, malicious behaviors are detected as normal behaviors. Thus, a lower false alarm rate is more desirable.

$$FAR = \frac{F_{Pos}}{T_{Pos} + F_{Pos}} \quad (18)$$

Due to the constant advancements in network intrusion detection technology, there is now a larger selection of publicly accessible network intrusion detection datasets that are diverse and tailored to specific contexts, serving as valuable resources for research (Wang et al., 2024d). To achieve better detection results, further processing is required. The calculation of the dataset's imbalance ratio is as Equation (19).

$$imbalance-ratio=\frac{N_{\min}}{N_{\max}} \tag{19}$$

where N_{min} is the number of the minority class samples, and N_{max} denotes the number of the majority class samples, a scalar object has been created using the MinMax class Then, the minimum and maximum values have been found for each feature, and then all data has been converted based on the minimum and maximum values in the range of 0 and 1. This method is obtained using Equation (20).

$$Z=\frac{X-\min(x)}{\max(x)-\min(x)} \tag{20}$$

Here Z is normalized data, X is original data, min(X) is the minimum data of the feature, max(x) is the maximum data of the feature. The ratio of training data to test data in this paper is 80:20. That is, 80% of the data from the dataset is used for training the model, and 20% of the data from the dataset is used for testing and validating the proposed model. Figure 15 clearly reveal the comparison between the new methods and the proposed model (IoT-APOSCA) on DS2OS dataset in terms of accuracy rate.

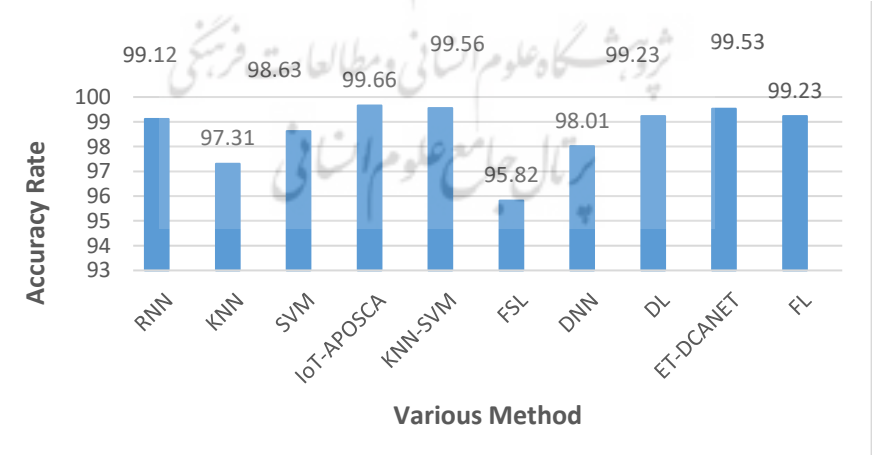


Figure 15. Comparison of the tested methods and the proposed model on DS2OS dataset

Figure 16 graphically indicates the difference between various methods on DS2OS dataset based on the recall (detection rate).

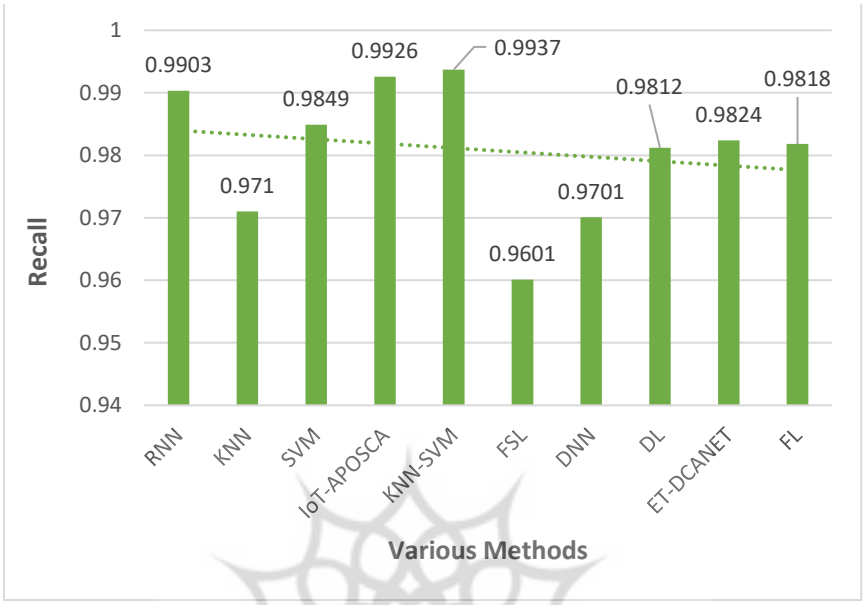


Figure 16. Graphically indicates the difference between various methods on DS2OS dataset based on the recall (detection rate)

Figure 17 demonstrates the variation between different methods and the proposed model on DS2OS.Considering that the lower the FAR, the better the performance of the model.

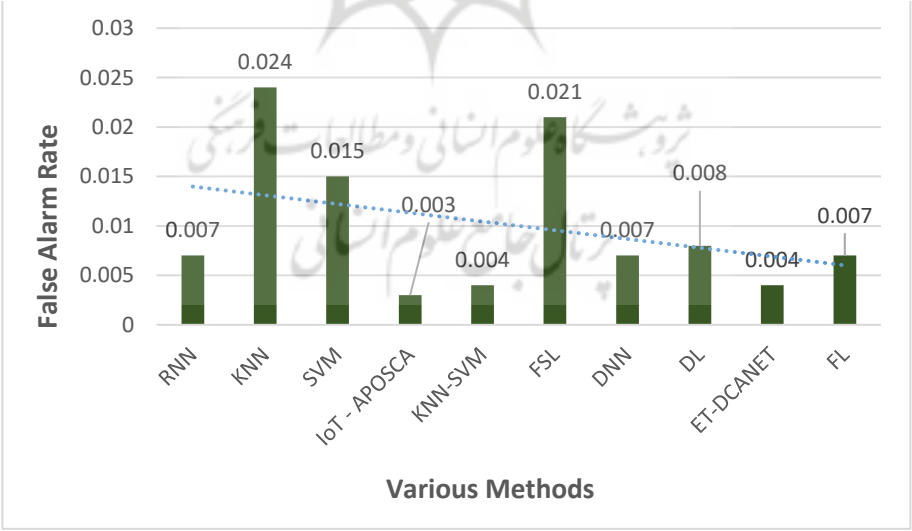


Figure 17. Comparison of the tested methods and the proposed model on DS2OS dataset

6. Conclusion and Future works

Intrusion detection system efficacy is directly tied to the quality of its classification method. Enhanced neural network performance is achievable through adjustments to parameters, such as network weights. This paper presents an approach to IoT network attack detection that leverages evolutionary intelligence and random neural networks. The proposed model used Arctic puffin optimization with SCA to improve the random neural network's architecture and internal parameters. With 99.66% accuracy, 0.9926 recall, and a 0.003 false alarm rate on DS2OS, the proposed model proved effective. Future efforts will concentrate on model optimization via various metaheuristic algorithms in vehicular networks, while Edge computing offers IoT services at the network edge. This approach enhances the efficiency and scalability of IoT devices. To solve latency and location awareness problems in cloud architectures, fog computing is used. Being decentralized, the Fog platform is perfect for many IoT applications. We propose using this model for future Internet of Things applications in fog domain security.

Conflict of interest

The authors declared no conflicts of interest.

Authorship contribution

Parisa Rahmani: Writing– review & editing, methodology, conceptualization, supervision, software.

Mohammad Arefi: Writing– original draft, software, writing– review & editing, methodology, formal analysis, data curation, resources.

Hamid Shokrzadeh: Data curation, validation, visualization, resources.

Ethical considerations

The authors have completely considered ethical issues, including informed consent, plagiarism, data fabrication, misconduct, and/or falsification, double publication and/or redundancy, submission, etc. This article was not authored by artificial intelligence.

Data availability

The dataset generated and analyzed during the current study is available from the author on reasonable request.

Funding

This research did not receive any grant from funding agencies in the public, commercial, or non-profit sectors.

References

Anwar SS, Asaduzzman, Sarker IH. (2024). “A differential privacy aided DEEPFED intrusion detection system for IOT applications”. *Security and Privacy*. 7(6).

https://doi.org/10.1002/spy2.445.

Chen C, Wang LC, Yu CM. (2022). **"**D2CRP: A novel distributed 2-hop cluster routing protocol for wireless sensor networks". *IEEE Internet of Things Journal.* 9(20). https://doi.org/10.1109/JIOT.2022.3148106.

Chen K, Fengyu Z, Lei Y, Shuqian W, Yugang W, Fang W. (2018). "A hybrid particle swarm optimizer with sine cosine acceleration coefficients". *Information Sciences.* 422: 218-241. http://dx.doi.org/10.1016/j.ins.2017.09.015.

Hasan M, Islam M, Zarif II, Hashem MMA. (2019). "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches". *Internet Things.* 7. https://doi.org/10.1016/j.iot.2019.100059.

Khan S, Alvi AN, Awais Javed M, Al-Otaibi YD, Kashif Bashir A. (2021). "An efficient medium access control protocol for RF energy harvesting based IoT devices". *Computer Communications.* 171: 28-38. https://doi.org/10.1016/j.comcom.2021.02.011.

Kumar GS, Premalatha K, Maheshwari GU, Kanna PR, Vijaya G, Nivaashini M. (2024). "Differential privacy scheme using Laplace mechanism and statistical method computation in deep neural network for privacy preservation". *Engineering Applications of Artificial Intelligence.* 128. https://doi.org/10.1016/j.engappai.2023.107399.

Kumar GS, Premalatha K, Maheshwari GU, Kanna PR. (2023). "No more privacy concern: A privacy-chain based homomorphic encryption scheme and statistical method for privacy preservation of user's private and sensitive data". *Expert Systems with Applications*. 234. https://doi.org/10.1016/j.eswa.2023.121071.

Latif S, Zou Z, Idrees Z, Ahmad J. (2020). "Novel attack detection scheme for the industrial internet of things using a lightweight random neural network". *IEEE Access*. 8. https://doi.org/10.1109/ACCESS.2020.2994079.

Maazaahi M, Hosseini S. (2025). "Machine learning and metaheuristic optimization algorithms for feature selection and botnet attack detection". *Knowledge and Information Systems*. 67: 3549-3597. https://doi.org/10.1007/s10115-024-02322-0.

Mirjalili S. (2016). "SCA: A Sine Cosine Algorithm for solving optimization problems". *Knowledge-Based Systems.* 96: 120-133. http://dx.doi.org/10.1016/j.knosys.2015.12.022.

Pahl MO, Aubet FX. (2018a). "Ds2Os Traffic Traces IOT Traffic Traces". https://www.kaggle.com/francoisxa/ds2ostraffictraces.

Pahl MO, Aubet FX (2018b). "All eyes on you: Distributed multidimensional IoT microservice anomaly detection". in *Proc. 14th Int. Conf. Netw. Service Manage (CNSM).* Nov. pp. 72-80.

Rajabi S, Asgari S, Jamali S, Fotohi R. (2024). "An intrusion detection system using the artificial neural network-based approach and firefly algorithm". *Wireless Personal Communications.* 137: 2409-2440. https://doi.org/10.1007/s11277-024-11505-5.

Subramaniam A, Chelladurai S, Ande SA, Srinivasan S. (2024). "Securing IOT network with hybrid evolutionary lion intrusion detection system: a composite motion optimization algorithm for feature selection and ensemble classification". *Journal of Experimental & Theoretical Artificial Intelligence.* https://doi.org/10.1080/0952813X.2024.2342858.

Sveleba S, Katerynchuk I, Kuno I, Sveleba N, Semotyjuk O. (2021). "Investigation of the transition mechanism to chaos in multilayer neural networks". *IEEE 4th International Conference on Advanced Information and Communication Technologies (AICT)*. https://doi.org/10.1109/aict52120.2021.9628919.

Wang LD, Yang G. (2024). "A network intrusion detection system based on deep learning in the IOT". *The Journal of Supercomputing*. 80: 24520-24558. https://doi.org/10.1007/s11227-024-06345-w.

Wang Z, Yang X, Zeng Z, He D, Chan S. (2024a). "A hierarchical hybrid intrusion detection model for industrial internet of things". *Peer-to-Peer Networking and*

Applications. 17: 3385-3407. https://doi.org/10.1007/s12083-024-01749-0.

Wang W, Tian W, Xu D, Zang H. (2024b), "Arctic puffin optimization: A bio-inspired metaheuristic algorithm for solving engineering design optimization". *Advances in Engineering Software*. 195. https://doi.org/10.1016/j.advengsoft.2024.103694.

Wang Z, Yang X, Zeng Z, He D, Chan S. (2024c). "A hierarchical hybrid intrusion detection model for industrial internet of things". *Peer-to-Peer Networking and Applications*. 17: 3385-3407. https://doi.org/10.1007/s12083-024-01749-0.

Wang X, Dai L, Yang G. (2024d). "A network intrusion detection system based on deep learning in the IOT". *The Journal of Supercomputing*. 80(16): 24520-24558. https://doi.org/10.1007/s11227-024-06345-w.

Zhong C, Sarkar A, Manna S, Khan MZ, Noorwali A, Das A, Chakraborty K. (2024). "Federated learning-guided intrusion detection and neural key exchange for safeguarding patient data on the internet of medical things". *International Journal of Machine Learning and Cybernetics*. 15: 5635-5665. https://doi.org/10.1007/s13042-024-02269-2.