Journal of Theoretical and Applied Information Technology
30th November 2024. Vol.102. No. 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

OPTIMIZED ANN HYPERPARAMETERS TO IDENTIFY MALICIOUS TRAFFIC IN NETWORKS

HIND KHOULIMI[1], OTHMAN BENAMMAR[2]

[1,2]Applied Mathematics and Computing Laboratory, Higher Normal School, Hassan II University, Casablanca, Morocco

E-mail: [1]hind-khoulimi-etu@etu.univh2c.ma, [2]Othman.benammmar@univh2c.ma

ABSTRACT

Intrusion detection systems have been a critical research area for over three decades. With the growth of internet traffic, the number of attacks that violate the confidentiality, integrity, and authenticity of important data has increased significantly. The advent of Artificial Intelligence (AI) especially Deep Learning (DL) creates models automatically to detect malicious traffic without human intervention. In this context, we have proposed an Intelligent Security Management System (ISMS) based on detection, analysis, and action engines. To guarantee efficiency and accuracy, we have used Artificial Neural Network (ANN) as a classification model, and to achieve better accuracy, different optimization algorithms are applied to select the best hyperparameters (weights and biases) for our model. In this present paper, we have based our study on KDD CuP 99, NSL KDD, and UNSW-NB 15 dataset to evaluate the best combination of ANN and optimization algorithms, using metrics: accuracy, loss, training time, precision, recall, and F1-score.

Keywords: *Artificial Neural Network, Harris Hawks Optimization, Particle Swarm Optimization, Spider Monkey Optimization; Cat Swarm Optimization.*

1. INTRODUCTION

In today's digital era, cybersecurity plays a critical role in safeguarding computer systems, networks, programs, and data from unauthorized access and malicious activities. It encompasses the use of various technologies and practices to ensure the confidentiality, integrity, and availability of data throughout its storage, processing, and transmission. However, as cybersecurity expert Gene Spafford pointed out, "The only truly secure system is switched off and unplugged, locked in a titanium-lined safe, buried in a concrete bunker, and surrounded by nerve gas and highly paid armed guards. Even then, I wouldn't stake my life on it." This highlights the growing challenge of securing systems against increasingly sophisticated cyber threats.

According to Cisco's annual internet report, the number of Distributed Denial of Service (DDoS) attacks has surged to 15.4 million in recent years, nearly doubling the 7.9 million attacks recorded in 2018. This surge underscores the urgent need for robust security measures to address vulnerabilities and protect networks from both known and unknown cyberattacks. While various security mechanisms such as encryption, user authentication, antivirus software, and firewalls have been developed, these solutions fall short in analyzing network packets to detect sophisticated attacks and prevent data breaches [1]. As a result, Intrusion Detection Systems (IDS) have become an essential component of cybersecurity architectures, deployed after firewalls to monitor and secure network traffic and protect against adversarial activities [2].

IDS systems typically rely on two main detection techniques: Misuse Detection (MD), which identifies known attack patterns, and Anomaly Detection (AD), which detects deviations from normal behavior. A more advanced approach, Hybrid Detection (HD), combines both methods to improve accuracy. However, traditional IDS systems face challenges in processing large volumes of data and detecting increasingly complex cyberattacks. The limitations of manual processing in the face of growing data complexity have made the integration

of Artificial Intelligence (AI) crucial in enhancing IDS capabilities.

AI has demonstrated its potential across various industries, including healthcare, education, manufacturing, and transportation, making it a promising solution for improving IDS performance [3]. By leveraging AI techniques, particularly Machine Learning (ML) and Deep Learning (DL), IDS systems can achieve automated, accurate, and scalable attack detection with minimal human intervention [4].

The selection of the research problem stems from the growing need to improve Intrusion Detection Systems (IDS) in the face of increasingly complex cyber threats that current security measures struggle to detect. In response to this need, we propose an Intelligent Security Management System (ISMS) [5] that manages IDS using three key engines: detection, analysis, and action as shown in Figure 1. The focus of this paper is on developing a high-performing model based on Artificial Neural Networks (ANNs) for classifying network traffic as benign or malicious. However, traditional ANNs, when trained using gradient descent methods, often encounter challenges such as getting trapped in local minima, which limits their accuracy.

To address these challenges, this paper introduces a hybrid approach that combines ANN with optimization algorithms such as Spider Monkey Optimization (SMO), Harris Hawks Optimization (HHO), Cat Swarm Optimization (CSO), and Particle Swarm Optimization (PSO). These algorithms are used to optimize ANN parameters (weights and biases) to enhance classification accuracy and ensure rapid, intelligent attack detection. The findings of this study extend beyond previous work by demonstrating how combining ANNs with optimization techniques—such as Spider Monkey Optimization, Harris Hawks Optimization, Cat Swarm Optimization, and Particle Swarm Optimization leads to higher classification accuracy and faster detection of cyber threats. By comparing these techniques across multiple benchmark datasets (KDD Cup 99, NSL KDD, and UNSW-NB 15), this study identifies an optimal model that significantly improves IDS capabilities over traditional models. Our findings thus provide new insights into the effectiveness of optimization algorithms for enhancing IDS, contributing both theoretically and practically to the field of cybersecurity.

The main contributions of this paper are as follows:

- Enhance Intrusion Detection System (IDS) performance by integrating Deep Learning techniques with optimization algorithms, aiming to improve detection accuracy and response times.
- Evaluation and comparison of the proposed models using three benchmark datasets: KDD Cup 99, NSL KDD, and UNSW-NB 15.
- Identify an optimal model for enhancing IDS accuracy.

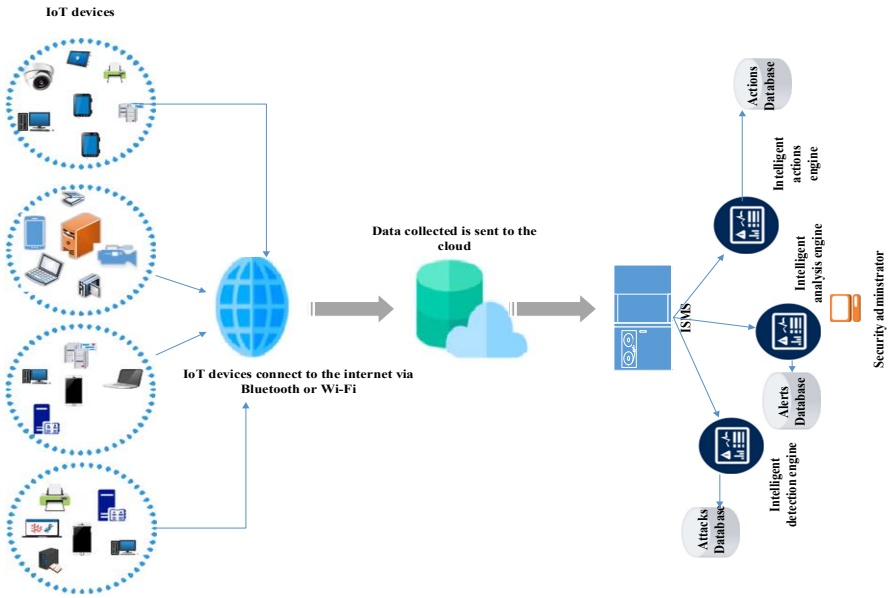


Figure 1. The architecture of Intelligent Security Management System

The remaining sections are organized as follows: Section I presents an introduction. Section II presents an overview and related work. Section III discuss the methodology of this paper. Section IV describes the results obtained with discussion. Conclusion and future work are presented in section V.

2. OVERVIEW AND RELATED WORK

In [6] Authors used the Neural Network and Particle Swarm Optimization algorithm to detect intrusion and attacks in cloud computing. The methods were tested for NSL KDD, and KDD CUP datasets. The results showed improved accuracy in detecting attacks and intrusions by unauthorized users.

In [7], the Authors create a new intrusion detection model to classify binary, triple, and multi-class attacks. The model is based on a Whale Optimization algorithm (WOA) to adjust the weight vector of the ANN to achieve the minimum mean square error. They used the Mississippi State University and Oak Ridge National Laboratory databases of power-system attacks to demonstrate the proposed model and show the experimental results. The comparison results show the superiority of the proposed WOA-ANN model.

In [8] Authors tries to build an Intrusion Detection System (IDS) for detecting security breaches in computer and network systems. For this reason, a new natural evolutionary algorithm (EA) called Multiverse Optimizer (MVO) is investigated and combined with ANN to develop advanced detection approaches for an IDS. This MVO-ANN model is applied to NSL-KDD and the new benchmark dataset called UNSWNB15. The results using UNSW-NB15 are better than those that were obtained using NSL-KDD.

In [9] Authors presented an advanced IDS based on a combination of particle swarm optimization and neural network algorithms applied on KDDCUP99, NSL-KDD, and CIDD datasets. A preprocessing is previously performed on these datasets to choose a subset of the features, reduce dimension, and normalize data. The proposed model classifies the attacks, reduces the number of false alarms, and provides a higher accuracy compared to other algorithms.

In [10] Authors try to improve the accuracy and efficiency of distinguishing normal traffic from abnormal one. The Authors carried out on a new approach for intrusion detection based on Artificial Bee Colony algorithm (ABC) and Monarch Butterfly Optimization (MBO) for preselecting the suitable biases ad weights of an ANN in order to increase the precision degree of classification for malicious and non-malicious traffic. Then, the ANN is retrained based on weights and biases obtained from the algorithm. The hybrid model was evaluated on three datasets namely KDD Cup 99, ISCX 2012, and UNSW-NB15. The results obtained were compared to nine other algorithms, demonstrating that the proposed model provides significant enhancement and efficiency to network intrusion detection.

In the research reported by [11] Authors introduced an intrusion detection system based on data mining and ML to detect network intrusion patterns. The model is based on ANN as a learning technique. The Grasshopper Optimization Algorithm (GOA) is used to minimize the intrusion detection error in the neural network by selecting suitable weight and bias. The models were evaluated by using KDD and UNSW datasets. The model outperforms and is more accurate than existing state-of-the-art techniques such as RF, XGBoost and embedded learning of ANN with BOA, HHO, and BWO algorithms in network detection.

In [12] Authors tested a wide range of ANN topologies, by changing both the number of hidden layers and the number of neurons on those layers. However, the choice of hyperparameters like the activation function, the optimizer, the batch size, and the number of epochs can impact the accuracy of the model. The tests were performed with the use of two IDS benchmark datasets, NSL-KDD and CICIDS2017.

In [13] Authors presented an efficient hybrid IDS, model which is built using MapReduce based Black Widow Optimized Convolutional-Long Short-Term Memory (BWO-CONV-LSTM) network. The first stage of this IDS model is the feature selection by the Artificial Bee Colony (ABC) algorithm. The second stage is the hybrid deep learning classifier model of BWO-COV-LSTM on a MapReduce framework for intrusion detection from the system traffic data. The proposed model is the combination of Convolutional and LSTM neural networks whose hyper-parameters are optimized by BWO to obtain the ideal architecture. Evaluation of the BWO-CONV-LSTM is performed on the NSL-KDD, ISCX-IDS, UNSW-NB15, and CSE-CIC-ISD2018 datasets. The results indicate that the proposed model BWO-CONV-LSTM model has high intrusion detection performance with 98.67%, 97.003%, 98.667% and 98.25% of accuracy for NSL-KDD, ISCX-IDS, UNSW-NB15, and CSE-CIC-IDS2018 datasets respectively, with fewer false values, less computation time and better classification coefficients.

In [14] presented a model detecting accurately an attack based on Whale with Cuckoo search

optimization (WCSO) based quantum neural network (QNN) and elliptical curve cryptography (ECC. Whale optimization algorithm (WOA) is used to choose the features in the network data that aid in precisely detecting intrusions. To identify attacks, the optimized quantum network which combines the WOA approach with the feedforward and backpropagation algorithms, is used. Sensitive data retrieving requires an encryption procedure that is enabled by the ECC algorithm, which could safely save the data files in the server, in order to secure the documentation with security measures The QNN with WOA-based IDS framework is a solid option for real-time intrusion detection analysis with high accuracy of 98.5%. Thus, the study has demonstrated that the suggested model will also provide better secure data storage, resolving security concerns.

In the research reported by [15], authors propose a novel IoT network intrusion detection approach based on Adaptive Particle Swarm Optimization Convolutional Neural Network (APSO-CNN). The PSO algorithm with change of inertia weight is used to adaptively optimize the structure parameters of one-dimensional CNN. The loss function value obtained from the first training CNN, is taken as the fitness value of PSO. Meanwhile, the comprehensive performance of proposed APSO-CNN shows that is the suitable tools for multi-type IoT network intrusion attack detection task.

3. METHODOLOGY

3.1 Model Classifier

The hybridization between ANN and optimization algorithms (HHO, PSO, SMO, and CSO) is similar. Each one is used to search over a hyperparameter space to find the combination of hyperparameters that results in the best performance on the validation set. The choice of hyperparameters can significantly affect the performance of the ANN.

During the optimization process HHO, SMO, PSO, and CSO creates a population of candidate solutions, which are represented as a vector of hyperparameters. These candidate solutions are then updated iteratively based on the fitness function, which is the validation accuracy of the ANN for a given set of hyperparameters. The fitness function is evaluated for each candidate solution, and the best solutions are selected for the next generation.

Each HHO, SMO, PSO, and CSO tries to balance exploration and exploitation during the optimization process. For example, HHO uses a mechanism called 'leadership hierarchy' to encourage exploration and exploitation simultaneously. This mechanism involves assigning a leadership position to a subset of candidate solutions based on their fitness values. The leaders then guide the search process by influencing the movement of the other candidate solutions toward promising areas of the hyperparameter space.

We come first to the building of the Fitness Function. In our case, for each optimizer (HHO, SMO, PSO, and CSO) we defined a function objf that implements (HHO, SMO, PSO, and CSO) algorithms for optimizing a given objective function that is used in the optimization process to evaluate the performance of ANN model. The fitness function takes a set of hyperparameters as input and returns a tuple of the model's validation accuracy and loss.

Here is how the hybridization happens between the optimization algorithm and the ANN model:

- The optimization algorithm (HHO, SMO, PSO, or CSO) generates a set of hyperprameters to be evaluated by the fitness function.
- The fitness function takes the generated hyperparameters and uses them to train an ANN model on the training data (Xtrain and Ytrain) for a fixed number of epochs (3 in our case) with a batch size of 16. It then evaluates the trained model on the validation data (Xvalid and Yvalid) and saves the model with the best validation accuracy using the ModelCheckpoint callback.
- The fitness function returns a tuple of the model's validation accuracy and loss to the optimization algorithm, which uses this information to update its search for better hyperparameters.

The four hybrid models follow the following steps as illustrated in Figure 2: The weight and biases of ANN are optimized through SMO, HHO, PSO and CSO. By SMO-ANN, HHO-ANN, PSO-ANN, and CSO-ANN weights and biases are modified at each step of the algorithm to reach to their optimal values as shown in Figure 3. Then, the weight and biases are fed into ANN again to help the ANN to minimize the attack detection error. This procedure is repeated until the required accuracy is achieved. Each optimization algorithm (SMO, HHO, PSO, and CSO) investigate the problem state to find the optimal solution. Each hybrid method (SMO-ANN, HHO-ANN, PSO-ANN, and CSO-ANN) is examined under three dataset KDD Cup 99, NSL KDD, and UNSW-NB 15 dataset. A train set and a test set are randomly selected from each dataset.

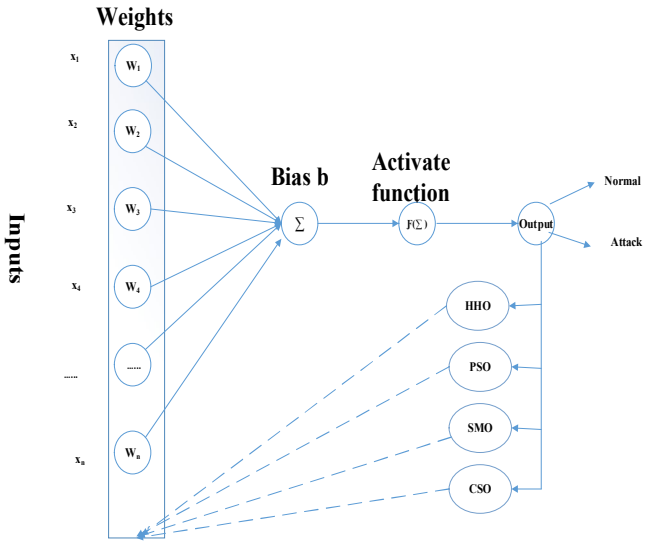


Figure 2: Hybrid classifier for ISMS-detection engine

3.2 Data and Metrics

In this section, all experiments have been conducted on a laptop kubuntu 20.04 operating system and Intel Core TM i3-8130U CPU@ 2.20 GHZ and 8 GB RAM. However, python version 2.7.18 has been used to implement the models using Jupyter Notebook which is a web application that allows to create an interactive environment that contains code, visualization and text.dimensionally.

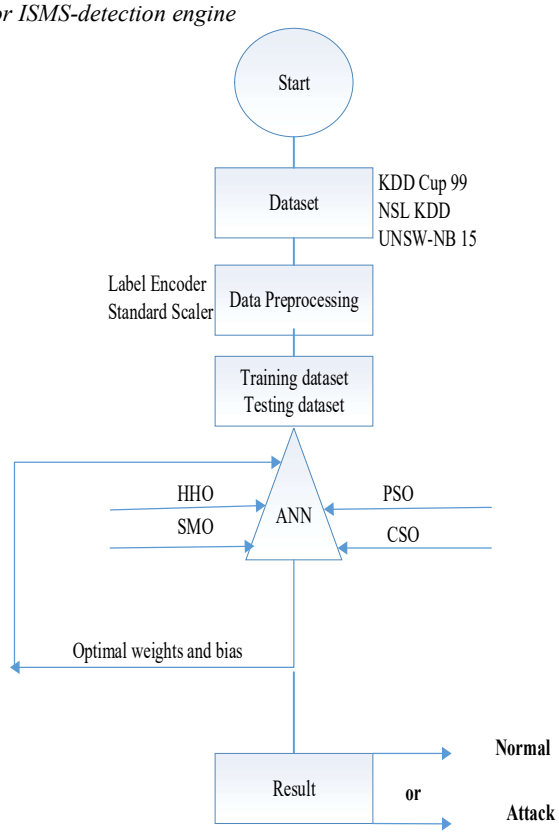


Figure 3: The usage of the optimization algorithm to select optimal weights and biases

3.2.1 Datasets

Datasets are critical to the development and testing of an IDS. However, DL models require a large amount of data to perform a better classification. Due to the unavailability of a recent dataset, we were forced to rely on existing datasets

that are commonly used by researchers for evaluating intrusion detection systems:

- KDD CuP 99 dataset: is Knowledge Discovery and Data Mining created in 1999. The dataset represents a simulated computer network environment. The KDD Cup 99. Data contains a large number of networks connection records, which are commonly referred to as network flows. Each network flow is characterized by a set of features that provide information about the connection, for example source and destination IP addresses, duration of the connection, protocol type, port numbers, and various statistical attributes. The dataset is divided into two parts: a training set and test set. The training set contains approximately 4.9 million records, while the test set contains around 311,029 records. The training set is further categorized into several classes, representing various attack types and legitimate links. The types of attacks included in the dataset are Denial of Service (DoS), User to Root (U2R), Remote to Local (R2L), Probing and the Normal case.
- NSL KDD: was created to solve the problems associated with the KDD Cup 99 dataset. These problems include redundant records and duplicate records. 41 attributes in the NSL KDD dataset are classified as either normal or attacked traffic. The NSL KDD dataset is divided into two sections: a test dataset and a training dataset. It contains four classes of attacks: DoS, U2R, R2L, and probing.
- UNSW-NB 15: was created by the Australian Center for Cyber Security (ACCS) which contains two million of records.

In the data analysis part that we followed, the main goal was to adjust the shape of the data so that it could be used with artificial intelligence algorithms. After the initial adjustment of the shape of the data and extracting the column that contain the categories of attacks that we need for the model to recognize them. Table 1 presents the number of samples in each of categories of attacks for KDD CuP 99, NSL KDD, and UNSW-NB 15 respectively. The data does not contain any missing data, then we come to the calculation of the correlation between the data columns and we notice that there is a strange formation between the columns stacked together and we also noted that there are some columns between them a very strong interdependence which is the problem must be solved. Therefore, before solving this problem, we applied the method of Interquartile Range (IQR) to identify outliers. The IQR is a measure of statistical dispersion and represents the range of the data from the first quartile (Q1) to the third quartile (Q3)

Table 1: Frequency of attack type values.

	Dos	Normal	Probe	R2L	U2R
KDD CuP 99	391458	97278	4107	1126	52
NSL KDD	67342	45927	11656	995	52
UNSW-NB 15	36171	145129	13160	39096	2520

The first step is to calculate the values for Q1 and Q3 using the quantile function. The quantile function divides the data into equal portions, and in this case, it calculates the values corresponding to the 25th percentile (Q1) and the 75th percentile (Q3) respectively. These percentiles help determine the range within which the majority of the once Q1 and Q3 are calculated, the next step is to calculate the IQR by subtracting Q1 from Q3. The IQR represents the spread of the middle 50% of the data.

To identify outliers, we set lower and upper bounds. The lower bound is calculated by subtracting 1.5 times the IQR from Q1, while the upper bound is obtained by adding 1.5 times the IQR to Q3. These bounds create a range beyond which data points are considered outliers.

Finally, we filter the dataFrame by selecting only those rows where the “duration” values fall within the calculated lower and upper bounds. This effectively removes any outliers from the “duration” column, ensuring that the subsequent analysis is not skewed by extreme values. Now, to solve the problem of the high correlation between some columns in the data. We extracted the columns with correlation greater than 0.8 (because data that have a very strong correlation in most cases contain negation of information and therefore this will cause a bias of the model to these columns and therefore the results will not be accurate although it may give correct results) and we got rid of them directly so that the correlation between those columns turns like this.

3.2.2 Evaluation Metrics

For proper evaluation of the effectiveness of an IDS, we are based on some important metrics:

- Confusion Matrix: represents the summary of the prediction in the form of a matrix, as shown in Table 2. It shows the number of correct and incorrect predictions per class.
- Accuracy: is the measure of the percentage of correct predictions and is expressed in the following way:

$$\frac{TP+TN}{TP+TN+FP+FN}$$

- Precision: determines the ratio of correct predictions to the total number of correct predictions.

$$\frac{TP}{TP+FP}$$

- Recall: is used to determine the ratio of the number of correct prediction results to the total number of results for a particular class and can be defined as:

$$\frac{TP}{TP+FN}$$

- F1 score: is a number between 0 and 1, which is the weighted average of the precision and recall. The F1 score, which is defined as follows, is seen to be a more accurate indicator of performance than accuracy.

$$\frac{2\times precision\times PR}{precision+TPR}$$

Table 2: Confusion matrix.

		Actual class	
		Normal	Attacks
Predicted class	Normal	TP	FP
	Attacks	FN	TN

3.3 Optimization Algorithms

3.3.1 Spider Monkey Optimization

SMO is a metaheuristic algorithm according to the spider monkey's foraging tactics. SMO is widely classified as an intelligent algorithm based on a fission-fusion social structure. The original version of the algorithm and its variants have been successfully used in several optimization issues to identify the optimal solution. SMO can effectively balance the trade-off between exploration and exploitation, which are two major components of any swarm intelligence-based algorithm [16]. The main steps of SMO's algorithm are:

- Initialization: The initiation phase generates an initial distributed spider monkey population, where SM_i represents the ith spider monkey (SM) in the population. Each SM_i is initialized as Equation (1):

$$SM_{ij} = SM_{minj} + U(0,1)\left(SM_{maxj} - SM_{minj}\right) \quad (1)$$

Where, SM_{minj} and SM_{maxj} are the lower and upper bounds of the search space in jth dimesion respectively, and U(0,1) is a randomly distributed number in the range of (0,1).

- The local leader phase is the exploration phase of the search region. During this stage, every group member updates their positions in the dimensions with a high perturbation as Equation (2):

$$SM_{newij} = SM_{ij} + U(0,1)x\left(LL_{kj} - SM_{ij}\right) + U(-1,1)\times\left(SM_{rj} - SM_{ij}\right) \quad (2)$$

Where, SM_{ij} is the jth dimension of ith SM, LL_{kj} represents the kth local leader of that group and SMrj is rth SM chosen arbitrarily within kth group in jth dimension such that r!=i and U(-1,1) is a distributed random number in the range (-1,1).

- Global leader phase: update their positions according to the probability based on the fitness of function as Equation (3):

$$Prob_i = 0.9\times\frac{\text{fit}_i}{\max\_\text{fit}} + 0.1 \quad (3)$$

The SM uses the global leader's knowledge, the neighboring SM's experience, and its persistence to update the position as Equation (4):

$$SM_{newij} = SM_{ij} + U(0,1)\times\left(GL_j - SM_{ij}\right) + U(-1,1)\times\left(SM_{rj} - SM_{ij}\right) \quad (4)$$

Where, GL_j is the position of group leader in jth dimension and j ∈ 1, 2, 3. . . D

- Global leader phase: update their positions according to the probability based on the fitness of function. The local Leader Learning phase (LLL) and Global Leader Learning phase (GLL), are employed to ensure the search procedure is not stagnating by updating the position of local leader and global leader by applying greedy selection, if there is no update, then local limit

count (LLL) and global limit count (GLL) are incremented by 1, the SM with best fitness is selected as local leader in (LLL phase) and as global leader in (GLL phase).

- The Local Leader Decision Phase is used to prevent local solutions from stagnating or converging prematurely. If a local leader doesn't reorganize within a certain limit, called LocalLeaderLimit, everybody in this group updates their positions using the global leader's experience, perturbation rate, or by random initialization as Equation (5):

$$SM_{newij} = SM_{ij} + U(0,1) \times (GL_j - SM_{ij}) + U(0,1) \times (SM_{rj} - LL_{kj}) \quad (5)$$

- Additionally, the final phase, the Global Leader Decision Phase, is designed to prevent the global best solutions from becoming stagnant. During this phase, the global leader's position is monitored and the global leader divides the population into smaller groups if it is not updated.

3.3.2 Harris Hawks Optimization

Harris Hawks Optimization (HHO) is a new nature-inspired population-based metaheuristic algorithm. The main inspiration of HHO algorithm that uses the Harris Hawk's hunting and pursuit patterns to capture prey in the wild. This algorithm mimics the strategies of exploration, exploitation, and attack of the Harris Hawk. The Harris Hawk, also called the Dusky Hawk, HH hunts in cooperative groups. HHO is used for the application of various optimization tasks such as satellite image segmentation, air pollution prediction, slope stability prediction, color image multilevel thresholding segmentation, and many more. HHO can be one of the smartest birds in the natural world. The main step's of HHO algorithm are:

- Exploration phase (Searching / Investigation for target)

HH searches for prey using random perches and waits for prey using two strategies, assuming equal chance q for each perch strategy. The first one, which is generated based on random location and other hawks, is modeled as Equation (6) for the condition of (q < 0.5). The second one is a random position of the group members and the random scaling of the components which is modeled as Equation (6) for the condition of (q ≥ 0.5). Where X(t+1) is the position vector, Xrand is a randomly selected Hawk from the current population, X(t) is the Hawk's current position, Xtarget is the target position, Xm(t) is the current Hawks' average position, r1, r2, r3, r4 are the random number between [0,1], and t is the current iteration.

$$X(t+1) = \begin{cases} X_{rand(t)} - r_1 \,|\, X_{rand(t)} - 2r2X(t) \,| \\ X_{target(t)} - Xm(t) - r_3(LowerBound + \\ r_4(UpperBound - LowerBound)) \end{cases} \quad (6)$$

The current population's average position might be calculated as Equation (7).

$$\mathrm{Xm}(\mathrm{t}) = \frac{1}{N}\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{X}_{\mathrm{i}}(\mathrm{t}) \quad (7)$$

Where Xm(t) is the average position of current hawks population, t is the current iteration, N is the total population size and Xi (t) is the hawk current position. After that, the Harris Hawk can take advantage of the Rabbit following this:

- When prey energy is low.
- Prey energy decrease while escaping from the hawk.

Depending on the prey's escape energy, the HHO algorithm can switch between various exploitation behaviors and exploration. The energy of the rabbit decreases during its escape activity. We model the prey energy as Equation (8):

$$E = 2E0(1 - \frac{t}{MaxT}) \quad (8)$$

Where, E is the energy of the prey's escape, E0 is the initial state of its energy inside the interval [-1,1]. However, MaxT is the maximum number of iterations and t is the current iteration.

- Exploitation phase

Once a prey has been detected, HH can launch a surprise attack. In this algorithm, four strategies are proposed for the representation of a hawk's attack on a rabbit:

✓ Case 01: Soft round up (E≥ 0.5 and r ≥ 0.5)
Hawk will perform Soft round up and Hard round up, HH (attackers) will surround the prey from different directions, softly or hardly (depends on the prey's energy level).
So, HH encircles the rabbit softly and make it more tired them the perform surprise attack. This

behavior is modeled as Equation (9) and Equation (10):

$$X(t+1) = \Delta X(t) - E\,|\,JX_{rabbit} - X(t)\,| \quad (9)$$

$$\Delta X(t) = X_{rabbit(t)} - X(t) \quad (10)$$

ΔX(t) is the difference between the rabbit's location and the current position t, r_5 is a random number between (0,1), and J = 2(1 − r_5) represents the strength of the random jumps of the rabbit during the entire escape procedure. The J value will change randomly in each iteration to imitate the characteristics of the rabbit's movements.

✓ Case 02: Hard round up (E < 0.5 and r ≥ 0.5):
In this case, Harris performs sudden attack. This behavior is modeled as Equation (11):

$$X(t+1) = Xraabit(t) - E\,|\,\Delta X(t)\,| \quad (11)$$

✓ Case 03: Soft round up with progressive rapid dives (E ≥ 0.5 and r < 0.5):
HH softly encircle the rabbit, making it tired, performing surprise attack. This behavior is mathematically modeled as Equation (12):

$$Y + Xrabbit(t) - E\,|\,JXrabbit(t) - X(t)\,| \quad (12)$$

Then, to determine whether it will be a good dive or not, they compare the possible results of such a move with the previous dive. In case of inappropriateness (when he sees that his prey makes more deceptive movements), he starts to dive irregularly, suddenly, and quickly. We assumed that they would dive based on LF-based patterns using the following Equation (13):

$$Z = Y + S \times LF(D) \quad (13)$$

Where S is a random vector of size 1 × D, D is the dimension of the problem, and LF is the delivery function calculated by the following Equation (14):

$$LF(x) = 0.01 \times \frac{(u \times \sigma)}{|v|^{\left(\frac{1}{\beta}\right)}}, \sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right) \quad (14)$$

Where u, v are random values within the range [0,1], β is a default constant that is set to 1.5. Therefore, the final strategy to update the Hawks' locations in this strategy can be performed as Equation (15):

$$X(t+1) = \begin{cases} Y \\ Z \end{cases} if \frac{F(Y) < F(X(t))}{F(Z) < F(X(t))} \quad (15)$$

Where Y and Z are obtained using new rules in equation (12) and equation (13). Each step selects only the better position or Z as next position. All search agents receive this energy.

✓ Case 04: Hard round up with progressive rapid dives (E < 0.5 and r < 0.5):
The hawks will try to reduce the distance between their average position and the position of the escaping prey. Therefore, in this strategy, under the following conditions, the following Equation (16) is carried out:

$$X(t+1) = \begin{cases} Y \\ Z \end{cases} if \frac{F(Y) < F(X(t))}{F(Z) < F(X(t))} \quad (16)$$

Where Y and Z are obtained using new rules in Equation (17) and Equation (18)

$$Y = X_{rabbit(t)} - E\,|\,JX_{rabbit(t)} - Xm(t)\,| \quad (17)$$

$$Z = Y + S + LF(D) \quad (18)$$

Where Xm(t) is calculated using Equation (8). Y or Z will be the next place where the new iteration will be released.

3.3.3 Particle Swarm Optimization

PSO is another optimization approach based on animal/bird behavioral studies. It was developed by Eberhat and Kennedy [17]. This algorithm is suitable for feature selection problems because of its simplicity in coding features, its global search capability, its computational efficiency, its fewer parameters, and its easier implementation. In PSO, particles form a population called a swarm and

represent candidate solutions in the search space. Swarms of particles are generated by a random distribution of 0 and 1. For each particle, if the principal component has a value of 1, it will be selected, and the principal component with a value of 0 will be ignored. This randomly initializes the particle swarm and then moves the swarm in the search space or principal space to update its position and velocity to determine which feature subset is optimal.

The current position of the particle position i and its velocity are shown in Equation (19) and Equation (20):

$$X_i = \{X_{i1}, X_{i2}, X_{i3}, \ldots, X_{iD}\} \quad (19)$$

$$V_i = \{V_{i1}, V_{i2}, V_{i3}, \ldots, V_{iD}\} \quad (20)$$

The velocity and position of the particle I are calculated by Equation (21) [18]

$$v_{id}^{(t+1)} = w \times v_{id}^{t} + c_1 \times r_1 \times (p_{id} - x_{id}^{t}) + c_2 \times r_{2i} \times (p_{gd} - x_{id}^{t})$$
$$x_{id}^{t+1} = v_{id}^{t} + v_{id}^{t+1} \quad (21)$$

Where t is the tth iteration in the process, d is the dth dimension in the search space, w is the inertia weight, c1 and c2 are acceleration constants, r1i, and r2i are randomly distributed in [0,1], pid and pqd represent the elements of pbest and gbest in the dth dimension.

To discover the optimal feature set, the position and velocity values of each particle are updated continually until a stopping criterion which could be either maximum iterations or adequate fitness.On each iteration, two new values, pbest and gbest, are added to the particles. The greatest solution to date is pbest (yij), whereas the second-highest score obtained from any particle in the population is gbest (ŷj).

3.3.4 Cat Swarm Optimization

Cat Swarm Optimization (CSO) is a swarm intelligence algorithm, initially developed by Chu et al in 2006 [19]. This approach inspired by the behavior of cats has proven their effectiveness in solving a variety of scientific and engineering optimization problems. It is inspired by resting and foraging behaviors. However, when cats are resting, they have a very high level of consciousness and they have a keen awareness of their surroundings. So, they are constantly monitoring the environment intelligently and deliberately, and when they see a target, they will start to move towards it very quickly. Each cat represents a solution set, which has its own location, a fitness value, and a flag. It consists of M dimensions in the search space, each with its own speed. The CSO algorithm consists of two modes: the Tracing Mode and the Seeking Mode. Consequently, we should run the cats through the algorithm after setting the number of cats to be included in the iteration. Every iteration saves the best cat, and the last one is the final solution.

✓ Seeking mode:

This mode mimics the resting behavior of cats, where four basic parameters play an important role: seeking range of selected dimension (SRD), search memory pool (SMP), self-position consideration (SPC), and number of dimensions to change (CDC). The seeking mode steps are as follows:

- From the current position of Cat, make as many copies as SMP.
- For each copy, randomly choose as many CDC dimensions as required to mutate.

Additionally, randomly add or subtract SRD values from the present ones. These values will replace the previous positions as shown in the following Equation (22):

$$Xjd_{new} = (1 + rand\,SRD) \times Xjd_{old} \quad (22)$$

Where Xjd_{old} is the current position, Xjd_{new} is the next position, j is the number of a cat and d is the dimensions, and rand is a random number in the range of [0, 1].

- Evaluation of the fitness score (FS) for all candidate positions.
- On the basis of the probability, one of the candidate points is selected as the cat's next position, and the candidate point with higher FS is more likely to be selected, as shown in Equation (23).

However, set the selection probability of each candidate point to 1 if all fitness values are equal.

$$Pi = \frac{|FS_i - FS_b|}{FS_{\max} - FS_{\min}} \quad \text{Where } <i<j \quad (23)$$

If the target is minimized, then F Sb = F Smax; otherwise, F Sb = F Smin.

✓ Tracing mode:

The cats' tracing behavior is mimicked in this mode. All dimensions of the cats' locations will have random velocity values supplied to them in the first iteration. For the next stage, the velocity values must be modified. In this setting, the cats will be as follows:

- Update the velocities ($V_{k,d}$) for each dimension by using equation (24).
- If a velocity value exceeds the maximum, then, it equals the maximum velocity.
- Update the Catk's location using the Equation (25)

$$V_{k,d} = V_{k,d} + r_1 c_1 \left(X_{best,d} - X_{k,d} \right) \quad (24)$$

$$V_{k,d} = V_{k,d} + V_{k,d} \quad (25)$$

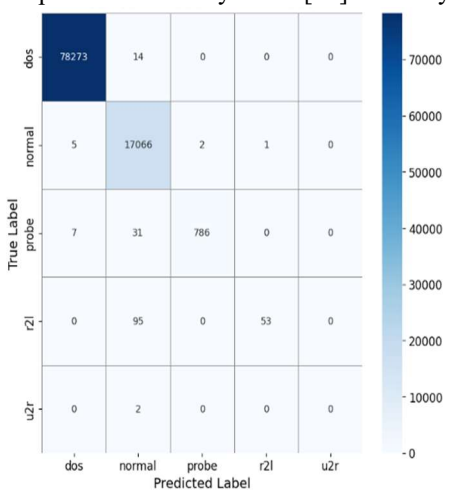
CSO has demonstrated its ability to solve diverse and complex problems in many fields [12]. One key feature of this method is that the two modes (tracing and seeking) are autonomous and distinct from one another. This allows researchers to balance the exploration and exploitation phases by easily modifying or improving these modes.

4. RESULTS AND DISCUSSION

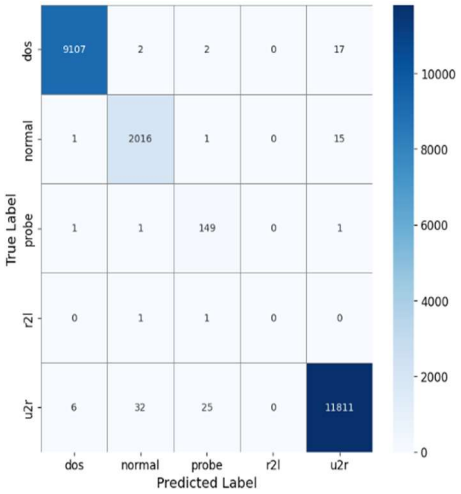
The performance of the proposed system in term of detection is evaluated under all metrics presented above. Furthermore, it was tested without any selection feature.

After evaluating the models, the results are shown in Table 3 for KDD CuP 99, NSL KDD, and UNSW NB-15 respectively in terms of loss scores and accuracy for training, validating, and testing dataset to benchmark our changes and see if the models improve.

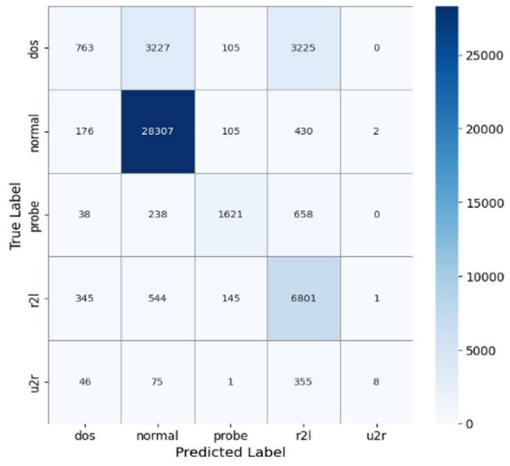
The confusion matrix for KDD CuP 99, NSL KDD, and UNSW NB-15 respectively as shown in Figure 4, Figure 5, Figure 6, Figure 7, and Figure 8. Based on this, we can calculate various metrics as shown in "Tab. 4" present the models performance, regarding to accuracy, precision, recall, F1-score, sensitivity, specificity, TP, and TN using the KDD CuP 99 and NSL KDD.



(a)

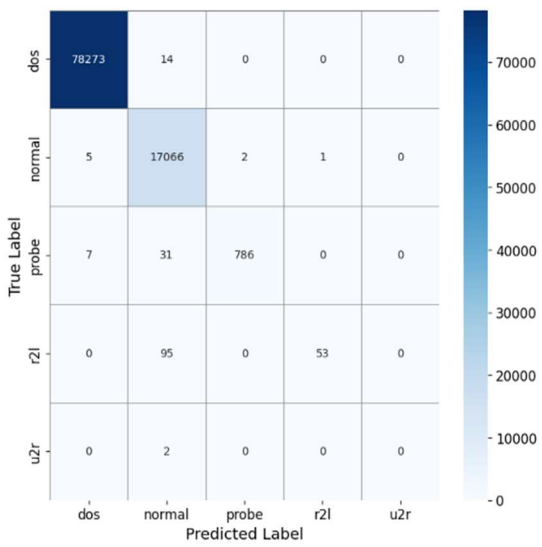


(b)

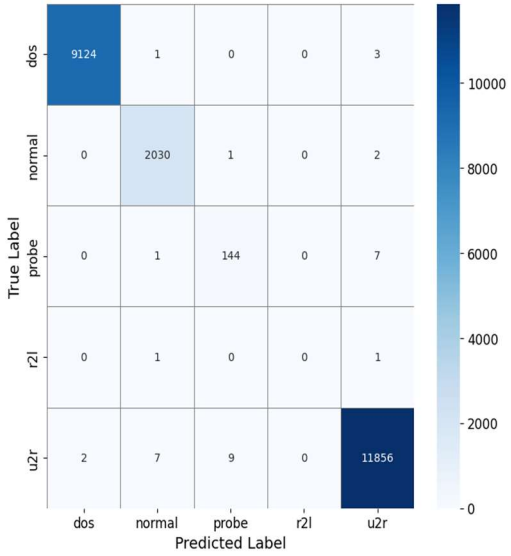


(c)

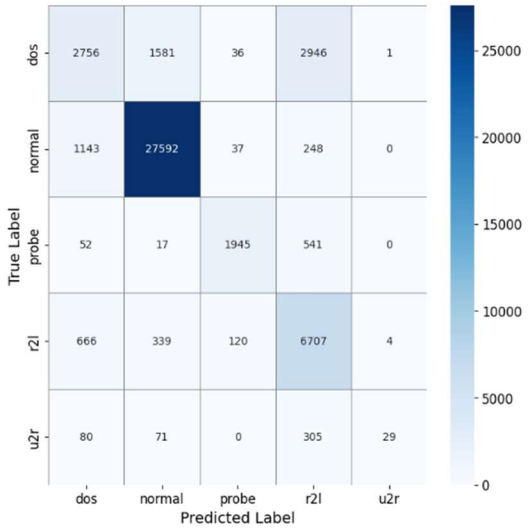
Figure 4: Confusion Matrix for ANN model for: (a) KDD CuP 99; (b) NSL KDD; (c) UNSW-NB 15.



(a)

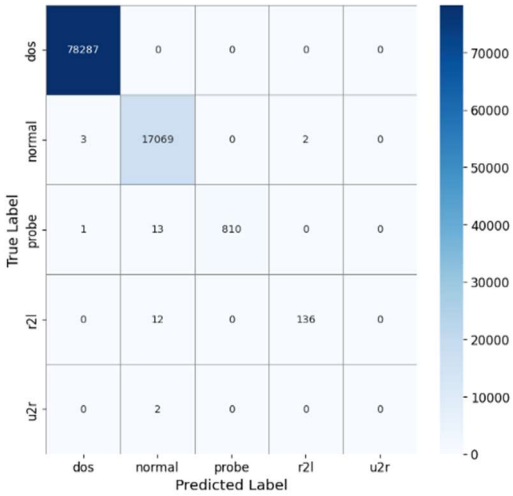


(b)

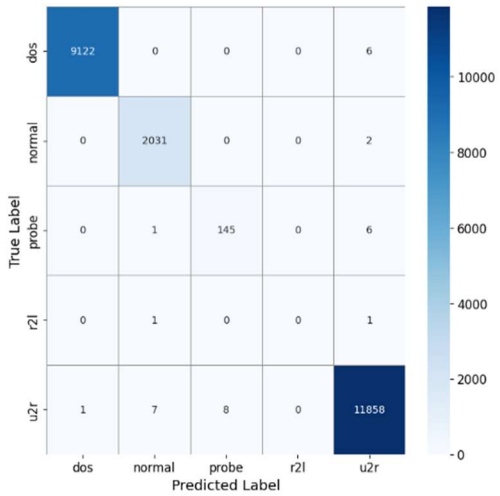


(c)

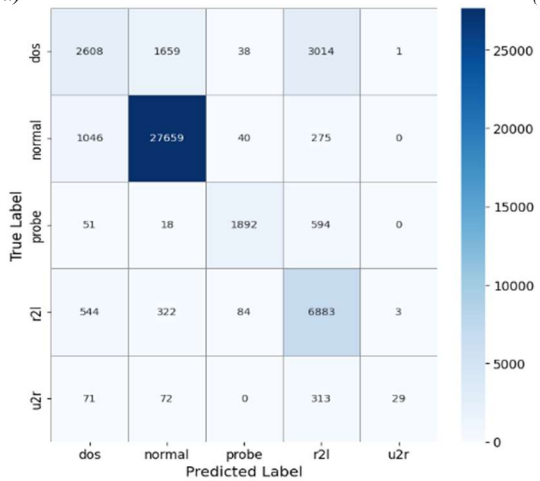
Figure 5: Confusion Matrix for HHO-ANN model for: (a) KDD CuP 99; (b) NSL KDD; (c) UNSW-NB 15.



(a)

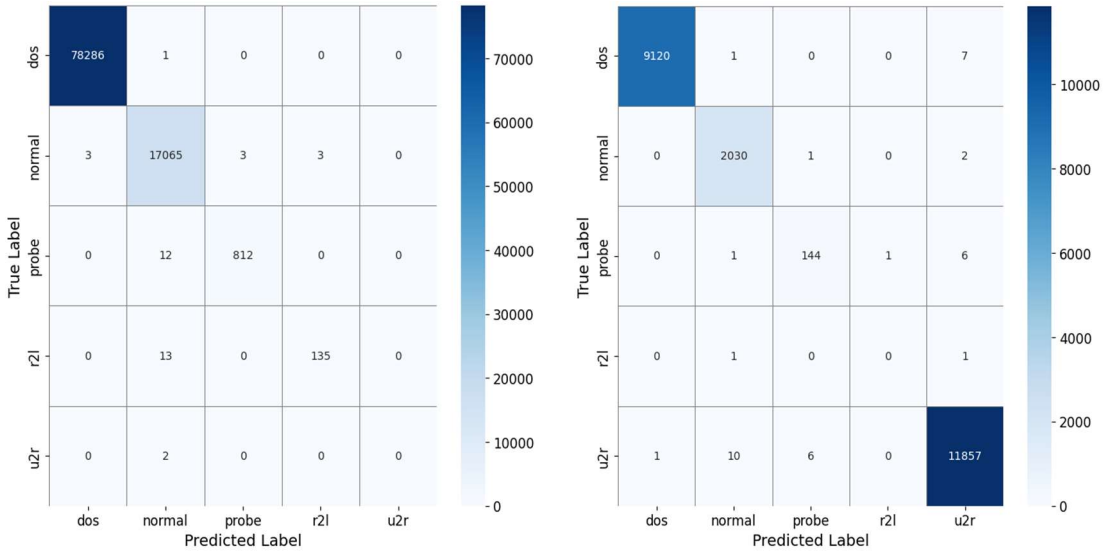


(b)

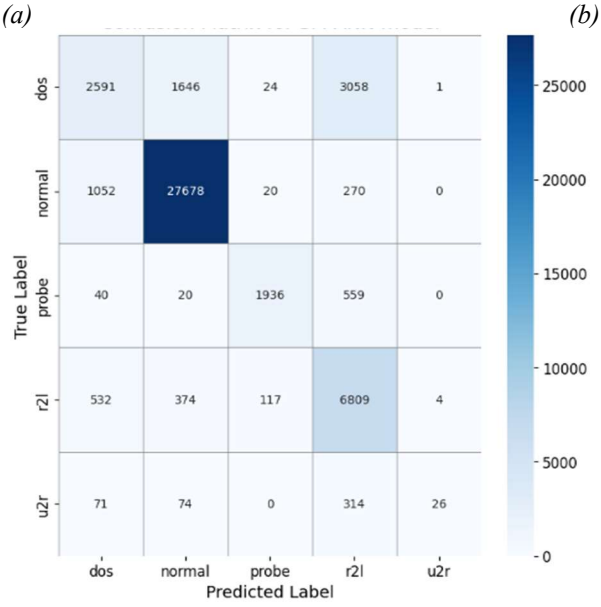


(c)

Figure 6: Confusion Matrix for PSO-ANN model for: (a) KDD CuP 99; (b) NSL KDD; (c) UNSW-NB 15.



(a) *(b)*



(c)

Figure 7: Confusion Matrix for SMO-ANN model for: (a) KDD CuP 99; (b) NSL KDD; (c) UNSW-NB 15.

Journal of Theoretical and Applied Information Technology
30th November 2024. Vol.102. No. 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

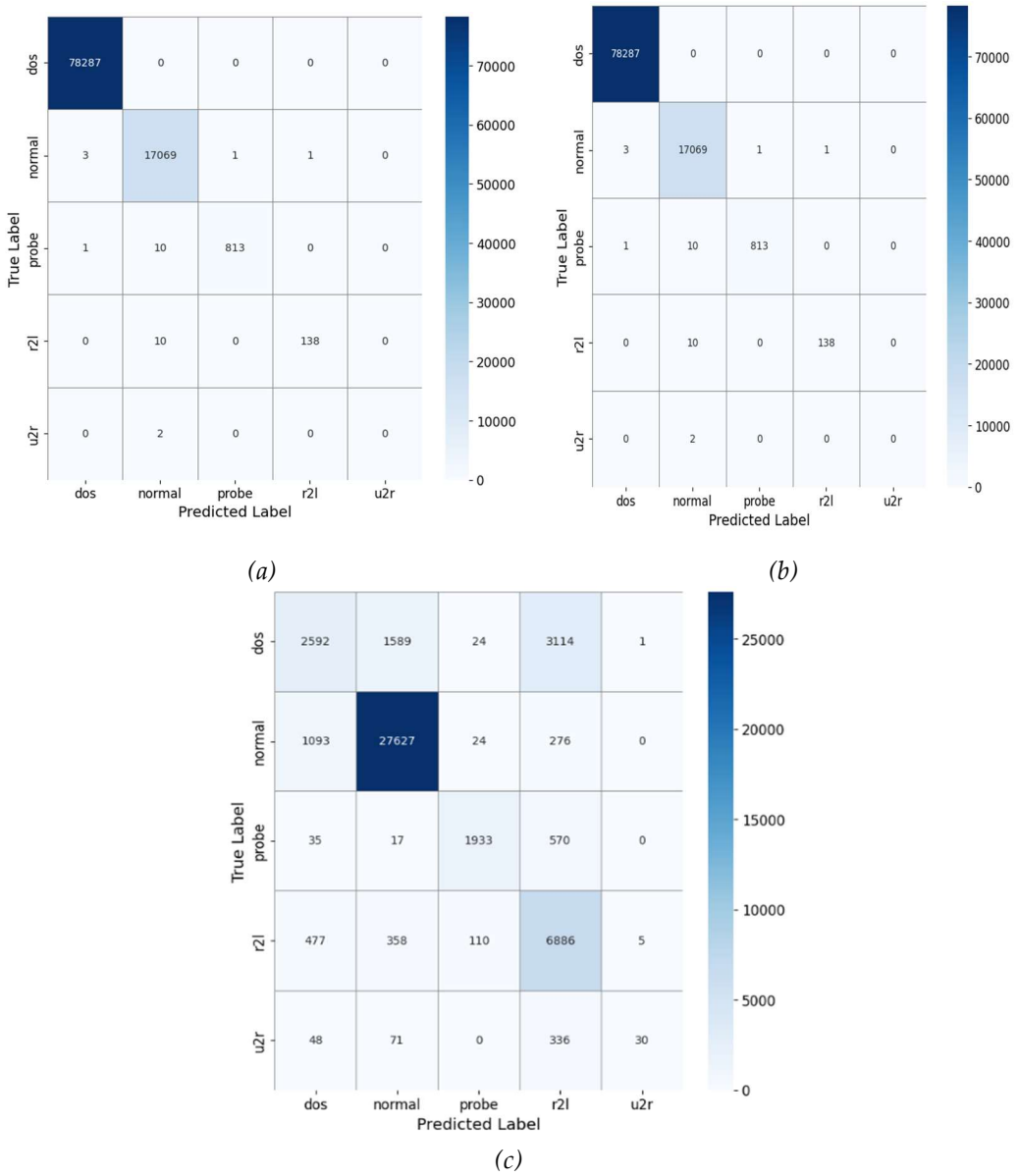


Figure 8: Confusion Matrix for CSO-ANN model for: (a) KDD CuP 99; (b) NSL KDD; (c) UNSW-NB 15.

Table 3: The accuracy and Loss.

		Accuracy			Loss		
		Training data	Validation data	Testing data	Training data	Validation data	Testing data
KDD CuP 99	ANN	99.81%	99.96%	99.97%	0.0058	0.0069	0.0059
	HHO+ANN	99.98%	99.96%	99.97%	0.0007	0.0415	0.1577
	PSO+ANN	99.98%	99.96%	99.97%	0.00082	0.0416	0.1655
	SMO+ANN	99.98%	99.96%	99.96%	0.00096	0.0465	1.2071
	CSO+ANN	99.98%	99.96%	99.97%	0.00063	0.0824	0.3076
NSL KDD	ANN	99.64%	99.48%	99.54%	0.0108	0.0182	0.014
	HHO+ANN	99.97%	99.87%	99.85%	0.00069	0.3539	0.1467

	PSO+ANN	99.97%	99.88%	99.86%	0.00081	0.316	0.1242
	SMO+ANN	99.98%	99.88%	99.84%	0.00053	0.3669	0.2854
	CSO+ANN	99.98%	99.89%	99.85%	0.00047	0.2789	0.2789
UNSW-NB 15	ANN	79.6%	80.12%	79.42%	0.4811	0.484	0.4945
	HHO+ANN	83.86%	83.16%	82.66%	0.342	0.4321	0.4846
	PSO+ANN	84.05%	83.13%	82.75%	0.3407	0.4033	0.3968
	SMO+ANN	83.05%	83.02%	82.68%	0.3403	0.4253	0.4377
	CSO+ANN	83.9%	83.08%	82.74%	0.3411	0.511	0.4395

Table 4: Detailed performance evaluation.

		ANN	HHO+ANN	PSO+ANN	SMO+ANN	CSO+ANN
KDD CuP 99	Accuracy	99.81%	99.97%	99.97%	99.96%	99.97%
	Precision	99.84%	99.96%	99.96%	99.95%	99.96%
	Recall	99.80%	99.96%	99.96%	99.96%	99.97%
	F1-score	99.82%	99.96%	99.96%	99.96%	99.96%
	Sensitivity	99.97%	100%	100%	99.99%	100%
	Specificity	99.99%	99.99%	99.99%	99.99%	99.99%
NSL KDD	Accuracy	99.54%	99.85%	99.86%	99.84%	99.85%
	Precision	99.55%	99.84%	99.85%	99.80%	99.80%
	Recall	99.50%	99.80%	99.80%	99.80%	99.80%
	F1-score	99.54%	99.84%	99.85%	99.83%	99.85%
	Sensitivity	99.97%	99.98%	100%	99.98%	99.98%
	Specificity	99.99%	99.99%	99.99%	99.99%	99.99%
UNSW-NB 15	Accuracy	79.42%	82.66%	82.75%	82.68%	82.74%
	Precision	77.38%	82.50%	82.73%	82.59%	82.79%
	Recall	79.42%	82.60%	82.74%	82.68%	82.74%
	F1-score	74.98%	81.55%	81.45%	81.37%	81.46%
	Sensitivity	19.12%	63.54%	61.12%	61.15%	61.94%
	Specificity	99.61%	98.74%	98.89%	98.90%	98.92%

Table 5: Comparison between the four models and other hybrid models over the KDD CuP 99

Model	Dataset	Accuracy	Precision	Recall		F1-score	Sensitivity	Specificity
HHO-ANN	KDD CuP 99	99.97%	99.96%	99.96%		99.96%	100%	99.99%
PSO-ANN		99.97%	99.96%	99.96%		99.96%	100%	99.99%
SMO-ANN		99.96%	99.95%	99.96%		99.96%	99.99%	99.99%
CSO-ANN		99.97%	99.96%	99.97%		99.96%	100%	99.99%
ABC-MBO-ANN [10]		87.62%						
GOA-ANN [11]		95.15 %					89.25%	93.17%
HHO-MLP [20]		99.13 %						

Table 6: Comparison between the four models and other hybrid models over the NSL KDD

Model	Dataset	Accuracy	Precision	Recall	F1-score	Sensitivity	Specificity
HHO-ANN	NSL KDD	99.85%	99.84%	99.80%	99.84%	99.98%	99.99%
PSO-ANN		99.86%	99.85%	99.80%	99.85%	100%	99.99%
SMO-ANN		99.84%	99.80%	99.80%	99.83%	99.98%	99.99%

CSO-ANN		99.85%	99.80%	99.80%	99.85%	99.98%	99.99%
CSA-ANN [21]		99.8%					
SMO-RF [22]		99.67%	99.95%	99.94%			
SMO-DNN [23]		99.4%	99.5%	99.5%	99.6%	99.4%	99.6%
SMO-ANN [24]		99.52%	99.37%				

Table 7: Comparison between the four models and other hybrid models over the UNSW-NB 15

Model	Dataset	Accuracy	Precision	Recall	F1-score	Sensitivity	Specificity
HHO-ANN	UNSW-NB 15	82.66%	82.50%	82.60%	81.55%	63.54%	98.74%
PSO-ANN		82.75%	82.73%	82.74%	81.45%	61.12%	98.89%
SMO-ANN		82.68%	82.59%	82.68%	81.37%	61.15%	98.90%
CSO-ANN		82.74%	82.79%	82.74%	81.46%	61.94%	98.92%
ABC-MBO-ANN [10]		95.72%					
GOA-ANN [11]		98.88%				98.14%	98.09%
HHO-MLP [20]		99.23%					

However, these results are compared with each other. The experimental results indicate that:
For ANN model:

- Accuracy indicates that the model correctly classified approximately 99.81%, 99.54%, and 79.42% for KDD Cup 99, NSL KDD, and UNSW NB-15 respectively of the total case.
- Precision indicates that around 99.84%, 99.55%, and 77.38% for KDD Cup 99, NSL KDD, and UNSW NB-15 respectively of cases predicted as positive by the model were indeed positive.
- Recall indicating that the model capture approximately 99.80%, 99.50%, and 79.42% for KDD Cup 99, NSL KDD, and UNSW NB-15 respectively of the actual positive cases.
- F1 score (99.82%, 99.54%, and 79.98% for KDD Cup 99, NSL KDD, and UNSW NB-15 respectively) which represents the harmonic mean of precision and recall.
- The sensitivity indicate that the model correctly identified around 99.97%, 99.97%, and 19.12% for KDD Cup 99, NSL KDD, and UNSW NB-15 respectively of the positive cases.
- The specificity suggest that the model achieved a high specificity for the different classes, ranging from 99.99% to 100%.
- The classifier achieved perfect precision, recall, and F1-score for the 'dos' and 'normal' class. The 'r2L' and 'u2r' class exhibits lower precision, recall, and F1-score because the were only two instances of 'r2L' class in the NSL KDD and 'u2r' class in KDD CuP 99. however, the 'r2L' class in KDD CuP 99 and 'u2r' class in NSL KDD achieved perfect precision, high recall, and F1-score.

For HHO-ANN model:

- The results indicate that the HHO+ANN model outperforms other models, demonstrating enhancements in accuracy, precision, recall, F1-score, sensitivity, specificity, as well as higher counts of true positives (TP) and true negatives (TN).
- We notice that the HHO-ANN model can classify 1 attack as 'u2r' class even if we have just two samples of this type of class which cannot be done by ANN model.
- The overall accuracy of the model 99.97%, 99.85%, and 82.66% for KDD Cup 99, NSL KDD, and UNSW NB-15 respectively indicating that it correctly predicted the majority of instances in the dataset.

For PSO-ANN:

- The model achieved perfect recall, precision, and F1-score for "dos" and "normal" classes, indicating that it correctly predicted all instances of these classes. The 'probe' class has a slightly lower recall, indicating that some instances were misclassified as other classes. The 'r2l' class has lower recall and F1-score, indicating that the model struggled to correctly identify instances of this class. The 'u2r' class has a precision, recall, and F1-score of 0, indicating that the model did not predict any instances for this class.

For SMO-ANN and CSO-ANN:

- The SMO-ANN exhibits a significantly elevated loss when evaluated on the testing data compared to the other models. Furthermore, the training duration for SMO-ANN and CSO-ANN exceeds that of the other models.

Based on comparison results for HHO-ANN, PSO-ANN, SMO-ANN, and CSO-ANN as shown in Table 4 and Figure 9:

For accuracy:

- We find that the accuracy of ANN model has increased with hybridization for HHO-ANN, PSO-ANN, SMO-ANN, and CSO-ANN with differences ranging: 0.16%, 0.16%, and 3.33% for KDD Cup 99, NSL KDD, and UNSW NB-15 respectively.

For loss:

- We can see that SMO-ANN produce higher loss for KDD Cup 99, NSL KDD, and UNSW-NB 15 respectively.

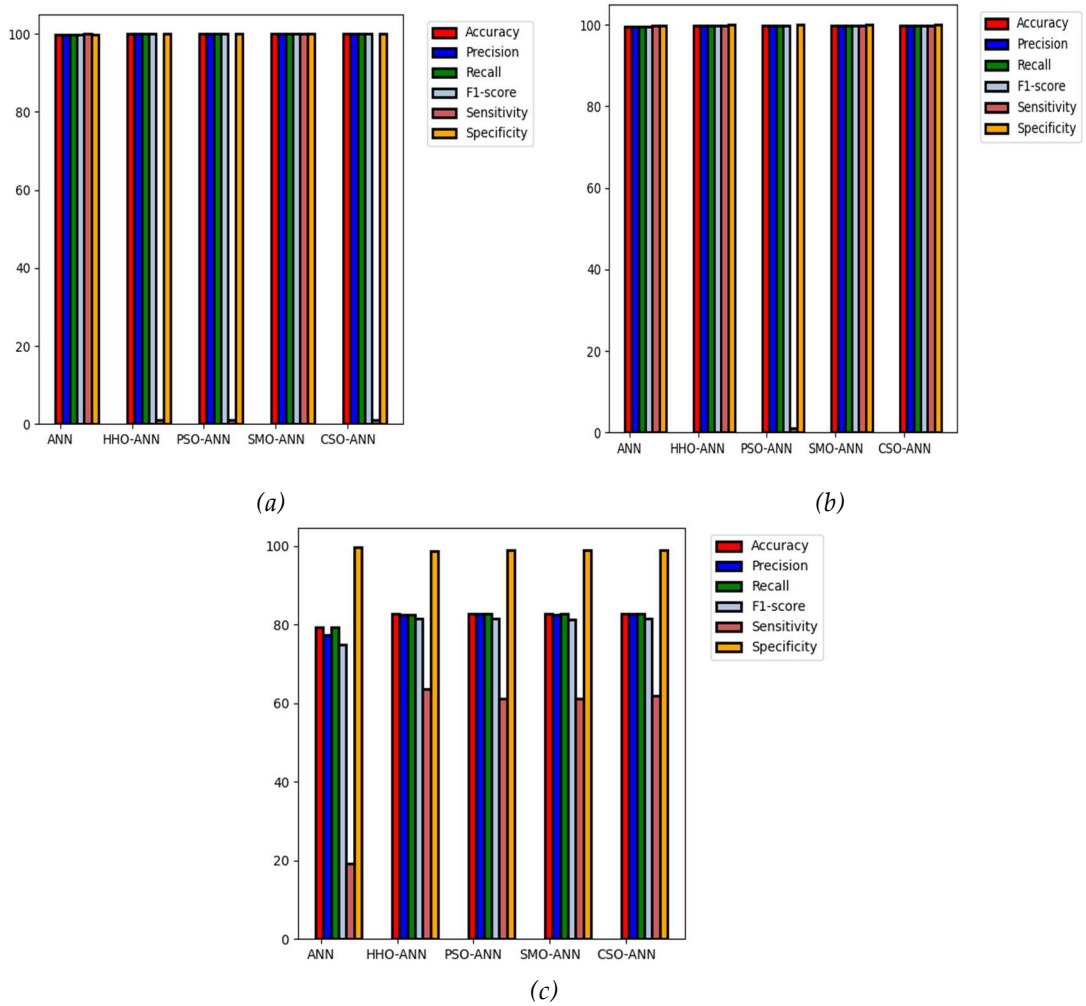


Figure 9: Comparison based on accuracy, precision, recall, F1-score, sensitivity, and specificity for: (a) KDD CuP 99; (b) NSL KDD; (c) UNSW-NB 15.

For precision:

- Indicate that the ANN model gains more precision when coupled with an optimized algorithm with differences ranging: from 0.12%, 0.30%, and 0.41% for KDD Cup 99, NSL KDD, and UNSW-NB 15 respectively.

For recall:

- A high recall score is 99.97% for CSO-ANN compared to ANN model of 99.80%. Similarly, for NSL KDD, the four hybrid models show the same high recall score (99.80%), while the ANN model achieved a slightly lower score of 99.50%.

However, UNSW NB-15 gained 6.75 % of the value of recall when coupled ANN with an optimization algorithm.

For F1-score:

- Reveals that HHO-ANN, PSO-ANN, SMO-ANN, and CSO-ANN all achieve the same F1-score of 99.96% for KDD Cup 99. However, for NSL KDD, PSO-ANN achieves a notably high F1-score of 99.85%. For UNSW-NB 15 a value of F1-score is passed 74.98% to 81.55% when coupled ANN with an optimization algorithm.

The hybridization of ANN with HHO, SMO, CSO, and PSO achieved higher accuracy, precision, recall, F1-score, sensitivity, and specificity, sources that met the following criteria: studies published in peer-reviewed journals, works that investigate IDS using AI techniques specifically ML and DL, and research that explores optimization algorithms in IDS contexts. We prioritized recent publications to capture advancements in IDS technologies and included benchmark studies that utilized datasets such as KDD Cup 99, NSL KDD, and UNSW-NB 15 to allow for meaningful comparison with our proposed methods as shown in Table 5 and Table 6. These datasets have simpler attack patterns that are well-understood and widely studied, allowing your optimization algorithms to effectively tune the parameters of the ANN for better classification of intrusion types. The structured nature of these datasets favors optimization algorithms like HHO and PSO, which are excellent for fine-tuning neural networks in well-structured environments with predictable patterns. As a result, your work outperforms previous approaches in terms of accuracy and precision on these datasets.

However, when applied to the UNSW-NB15 dataset, as show in Table 7 the hybrid model did not outperform other works in terms of accuracy and precision. The UNSW-NB15 dataset is more complex and includes a broader variety of attack patterns, which presents a greater challenge for conventional optimization methods. The diversity of attacks and more complex feature relationships in this dataset make it harder for optimization techniques like HHO, SMO, and PSO to navigate effectively, limiting their ability to optimize the ANN for higher accuracy and precision. In contrast, other authors may have used more advanced or specialized methods such as deep learning or ensemble models, which are better suited to handle the intricacy of UNSW-NB15, explaining why they achieved better accuracy and precision.

While this study provides meaningful insights into enhancing Intrusion Detection Systems (IDS) through Artificial Neural Networks (ANNs) optimized with advanced algorithms, it has certain limitations that should be acknowledged. Firstly, although the proposed model performs well on benchmark datasets (KDD Cup 99, NSL KDD, and UNSW-NB 15), these datasets may not fully capture the evolving nature and complexity of real-world cyber threats. Therefore, testing the model in more dynamic and unpredictable network environments would be beneficial to confirm its robustness.

Finally, the use of multiple optimization algorithms, such as Spider Monkey Optimization, Harris Hawks Optimization, Cat Swarm Optimization, and Particle Swarm Optimization, while beneficial for comparative analysis, adds computational complexity. The time and resource requirements for training and tuning may limit the model's practical applicability in resource-constrained settings. Future research could explore ways to streamline the optimization process, potentially by developing hybrid algorithms that retain high accuracy with lower computational demands.

5. CONCLUSION

This paper investigated the effectiveness of hybrid optimization algorithms combined with Artificial Neural Networks (ANNs) to enhance the detection accuracy of Intrusion Detection Systems (IDS) across three widely used datasets: KDD Cup 99, NSL KDD, and UNSW-NB 15. Our primary objective was to determine whether optimized ANNs could reliably differentiate between benign and malicious network traffic with high accuracy across diverse data environments.

The results demonstrate that different optimization algorithms yield varying degrees of success depending on the dataset characteristics. Specifically, HHO-ANN was found to perform best on KDD Cup 99, largely due to its ability to handle this dataset's inherent structure and frequency of attack patterns. This suggests that HHO-ANN's optimization approach aligns well with the simpler, more predictable data distributions typical of KDD Cup 99. In contrast, PSO-ANN excelled on NSL KDD, which has a more balanced data structure with fewer redundant records, indicating that PSO-ANN's feature extraction and optimization techniques may be more suitable for datasets with reduced bias. Lastly, CSO-ANN showed superior

performance on UNSW-NB 15, demonstrating adaptability to datasets with more complex and modern attack scenarios. These findings indicate that each model has strengths that align with specific dataset characteristics, emphasizing that no single model is universally optimal across all types of network data.

While these results meet our objectives, they also reveal limitations. The computational cost associated with training hybrid models remains a challenge, suggesting that future research should explore optimization methods that balance accuracy with efficiency. Moreover, although the models achieved high accuracy, their interpretability remains limited, which may affect real-world applicability in settings requiring human oversight. This insight calls for further development of interpretable AI methods within IDS frameworks to ensure the models are not only accurate but also transparent.

In summary, this study highlights the potential of tailored optimization techniques to enhance IDS performance. The results underscore the importance of selecting optimization approaches that align with dataset characteristics to achieve reliable outcomes. Future work will focus on expanding the model to additional datasets, refining its computational efficiency, and integrating it into our ISMS system to support real-time network security monitoring.

REFERENCES

[1] I. F. Kilincer, F. Ertam, et A. Sengur, « A comprehensive intrusion detection framework using boosting algorithms », *Computers and Electrical Engineering*, vol. 100, p. 107869, mai 2022, doi: 10.1016/j.compeleceng.2022.107869.

[2] M. H. Nasir, S. A. Khan, M. M. Khan, et M. Fatima, « Swarm Intelligence inspired Intrusion Detection Systems — A systematic literature review », *Computer Networks*, vol. 205, p. 108708, mars 2022, doi: 10.1016/j.comnet.2021.108708.

[3] C. Xie et F. Zhang, « A new sequence optimization algorithm based on particle swarm for machine learning », *J Ambient Intell Human Comput*, vol. 13, nº 5, p. 2601-2619, mai 2022, doi: 10.1007/s12652-021-03004-3.

[4] W. A. H. M. Ghanem, A. Jantan, S. A. A. Ghaleb, et A. B. Nasser, « An Efficient Intrusion Detection Model Based on Hybridization of Artificial Bee Colony and Dragonfly Algorithms for Training Multilayer Perceptrons », *IEEE Access*, vol. 8, p. 130452-130475, 2020, doi: 10.1109/ACCESS.2020.3009533.

[5] H. Khoulimi, M. Lahby, et O. Benammar, « Towards an intelligent system to manage IDS for IoT », in *2022 5th Conference on Cloud and Internet of Things (CIoT)*, mars 2022, p. 9-16. doi: 10.1109/CIoT53061.2022.9766759.

[6] A. S. Saljoughi, M. Mehrvarz, et H. Mirvaziri, « Attacks and Intrusion Detection in Cloud Computing Using Neural Networks and Particle Swarm Optimization Algorithms », *Emerging Science Journal*, vol. 1, nº 4, Art. nº 4, déc. 2017, doi: 10.28991/ijse-01120.

[7] L. Haghnegahdar et Y. Wang, « A whale optimization algorithm-trained artificial neural network for smart grid cyber intrusion detection », *Neural Comput & Applic*, vol. 32, nº 13, p. 9427-9441, juill. 2020, doi: 10.1007/s00521-019-04453-w.

[8] I. Benmessahel, K. Xie, et M. Chellal, « A new evolutionary neural networks based on intrusion detection systems using multiverse optimization », *Appl Intell*, vol. 48, nº 8, p. 2315-2327, août 2018, doi: 10.1007/s10489-017-1085-y.

[9] A. Shokoohsaljooghi et H. Mirvaziri, « Performance improvement of intrusion detection system using neural networks and particle swarm optimization algorithms », *Int. j. inf. tecnol.*, vol. 12, nº 3, p. 849-860, sept. 2020, doi: 10.1007/s41870-019-00315-9.

[10] W. A. H. M. Ghanem et A. Jantan, « Training a Neural Network for Cyberattack Classification Applications Using Hybridization of an Artificial Bee Colony and Monarch Butterfly Optimization », *Neural Process Lett*, vol. 51, nº 1, p. 905-946, févr. 2020, doi: 10.1007/s11063-019-10120-x.

[11] S. Moghanian, F. B. Saravi, G. Javidi, et E. O. Sheybani, « GOAMLP: Network Intrusion Detection With Multilayer Perceptron and Grasshopper Optimization Algorithm », *IEEE Access*, vol. 8, p. 215202-215213, 2020, doi: 10.1109/ACCESS.2020.3040740.

[12] M. Choraś et M. Pawlicki, « Intrusion detection approach based on optimised artificial neural network », *Neurocomputing*, vol. 452, p. 705-715, sept. 2021, doi: 10.1016/j.neucom.2020.07.138.

[13] P. R. Kanna et P. Santhi, « Hybrid Intrusion Detection using MapReduce based Black

Widow Optimized Convolutional Long Short-Term Memory Neural Networks », *Expert Systems with Applications*, vol. 194, p. 116545, mai 2022, doi: 10.1016/j.eswa.2022.116545.

[14] H. Kadry, A. Farouk, E. A. Zanaty, et O. Reyad, « Intrusion detection model using optimized quantum neural network and elliptical curve cryptography for data security », *Alexandria Engineering Journal*, vol. 71, p. 491-500, mai 2023, doi: 10.1016/j.aej.2023.03.072.

[15] X. Kan *et al.*, « A novel IoT network intrusion detection approach based on Adaptive Particle Swarm Optimization Convolutional Neural Network », *Information Sciences*, vol. 568, p. 147-162, août 2021, doi: 10.1016/j.ins.2021.03.060.

[16] X. Xia, W. Liao, Y. Zhang, et X. Peng, « A discrete spider monkey optimization for the vehicle routing problem with stochastic demands », *Applied Soft Computing*, vol. 111, p. 107676, nov. 2021, doi: 10.1016/j.asoc.2021.107676.

[17] I. Ahmad, « Feature Selection Using Particle Swarm Optimization in Intrusion Detection », *International Journal of Distributed Sensor Networks*, vol. 11, nº 10, p. 806954, oct. 2015, doi: 10.1155/2015/806954.

[18] M. Abdel-Basset, L. Abdel-Fatah, et A. K. Sangaiah, « Chapter 10 - Metaheuristic Algorithms: A Comprehensive Review », in *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, A. K. Sangaiah, M. Sheng, et Z. Zhang, Éd., in Intelligent Data-Centric Systems. , Academic Press, 2018, p. 185-231. doi: 10.1016/B978-0-12-813314-9.00010-4.

[19] A. M. Ahmed, T. A. Rashid, et S. Ab. M. Saeed, « Cat Swarm Optimization Algorithm: A Survey and Performance Evaluation », *Computational Intelligence and Neuroscience*, vol. 2020, nº 1, p. 4854895, 2020, doi: 10.1155/2020/4854895.

[20] M. Alazab, R. Abu Khurma, P. A. Castillo, B. Abu-Salih, A. Martín, et D. Camacho, « An effective networks intrusion detection approach based on hybrid Harris Hawks and multi-layer perceptron », *Egyptian Informatics Journal*, vol. 25, p. 100423, mars 2024, doi: 10.1016/j.eij.2023.100423.

[21] M. Imran, S. Khan, H. Hlavacs, F. A. Khan, et S. Anwar, « Intrusion detection in networks using cuckoo search optimization », *Soft Comput*, vol. 26, nº 20, p. 10651-10663, oct. 2022, doi: 10.1007/s00500-022-06798-2.

[22] E. Sandhya et Annapurani Kumarappan, « Enhancing the Performance of an Intrusion Detection System Using Spider Monkey Optimization in IoT », *IJIES*, vol. 14, nº 6, p. 30-39, déc. 2021.

[23] N. Khare *et al.*, « SMO-DNN: Spider Monkey Optimization and Deep Neural Network Hybrid Classifier Model for Intrusion Detection », *Electronics*, vol. 9, nº 4, Art. nº 4, avr. 2020, doi: 10.3390/electronics9040692.

[24] D. Kumari, A. Sinha, S. Dutta, et P. Pranav, « Optimizing neural networks using spider monkey optimization algorithm for intrusion detection system », *Sci Rep*, vol. 14, nº 1, p. 17196, juill. 2024, doi: 10.1038/s41598-024-68342-6.