Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.027581

ARTICLE

Improved Supervised and Unsupervised Metaheuristic-Based Approaches to Detect Intrusion in Various Datasets

Ouail Mjahed[1,*], Salah El Hadaj[1], El Mahdi El Guarmah[1,2] and Soukaina Mjahed[1]

[1]Faculty of Sciences and Technology, Department of Computer Sciences, Cadi Ayyad University, Marrakech, 40000, Morocco

[2]Mathematics and Informatics Department, Royal Air School, Marrakech, 40000, Morocco

*Corresponding Author: Ouail Mjahed. Email: mjahed.ouail97@gmail.com

Received: 04 November 2022 Accepted: 06 January 2023

ABSTRACT

Due to the increasing number of cyber-attacks, the necessity to develop efficient intrusion detection systems (IDS) is more imperative than ever. In IDS research, the most effectively used methodology is based on supervised Neural Networks (NN) and unsupervised clustering, but there are few works dedicated to their hybridization with metaheuristic algorithms. As intrusion detection data usually contains several features, it is essential to select the best ones appropriately. Linear Discriminant Analysis (LDA) and t-statistic are considered as efficient conventional techniques to select the best features, but they have been little exploited in IDS design. Thus, the research proposed in this paper can be summarized as follows. a) The proposed approach aims to use hybridized unsupervised and hybridized supervised detection processes of all the attack categories in the CICIDS2017 Dataset. Nevertheless, owing to the large size of the CICIDS2017 Dataset, only 25% of the data was used. b) As a feature selection method, the LDA performance measure is chosen and combined with the t-statistic. c) For intrusion detection, unsupervised Fuzzy C-means (FCM) clustering and supervised Back-propagation NN are adopted. d) In addition and in order to enhance the suggested classifiers, FCM and NN are hybridized with the seven most known metaheuristic algorithms, including Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Differential Evolution (DE), Cultural Algorithm (CA), Harmony Search (HS), Ant-Lion Optimizer (ALO) and Black Hole (BH) Algorithm. Performance metrics extracted from confusion matrices, such as accuracy, precision, sensitivity and F_1-score are exploited. The experimental result for the proposed intrusion detection, based on training and test CICIDS2017 datasets, indicated that PSO, GA and ALO-based NNs can achieve promising results. PSO-NN produces a tested accuracy, global sensitivity and F_1-score of 99.97%, 99.95% and 99.96%, respectively, outperforming performance concluded in several related works. Furthermore, the best-proposed approaches are valued in the most recent intrusion detection datasets: CSE-CICIDS2018 and LUFlow2020. The evaluation fallouts consolidate the previous results and confirm their correctness.

KEYWORDS

Classification; neural networks; Fuzzy C-means; metaheuristic algorithm; CICIDS2017; intrusion detection system

Abbreviations

AdaBoost	Adaptive Boosting
ALO	Ant-Lion Optimizer

BH	Black Hole algorithm
BPNN, CNN, NN	Back-propagation, Convolutional, Neural Network
CA	Cultural algorithm
CICIDS2017	IDS Dataset created by the Canadian Institute for Cyber-security in 2017
CTree	Conditional inference Tree
CSE-CICIDS2018	IDS Dataset created by the Communications Security Establishment (CSE) and the Canadian Institute for Cyber-security in 2018
DE	Differential Evolution algorithm
DL	Deep Learning
DT	Decision Tree
FCM	Fuzzy C-means
FKNN, K-NN	Fast, K-Nearest Neighbor Classifier
GA	Genetic Algorithm
HS	Harmony Search algorithm
IDS	Intrusion Detection System
IG	Information Gain
LDA, QDA	Linear, Quadratic Discriminant Analysis
LUFlow2020	IDS Dataset created by Lancaster University in 2020
ML	Machine Learning
MLP	Multi-Layer Perceptron
PCA	Principal Component Analysis
PSO	Particle Swarm Optimization
RBF	Radial Basis Function
RF	Random Forest
SVM	Support Vector Machines

Symbols

g_i	Centroid of class C_i
$A\left(A_{ij}\right)$	Confusion matrix
$W_{ij}^{(l)}$	Connection weight between the neurons i (in layer l) and j (in layer l-1)
V	Covariance matrix
X	Data instance
h_1	Discriminant function
d_{ij}	Euclidean distance between the i^{th} data and the j^{th} cluster center.
δ_i, δ	False decision rate for class C_i, Global false decision rate
f, f_i	Fitness function
m	Fuzziness index
F_{1i}, F_1	F_1-score for class C_i, Global F_1-score
c_i	i^{th} cluster center
$r_i^{(p)}$	i^{th} desired output value for data point p
$o_i^{(p)}$	i^{th} effective neuron output value for data point p
μ_{il}	Mean of feature x_i for class C_l
ω_{ij}	Membership of the i^{th} data to j^{th} cluster center
$N^{(l)}$	Number of neurons in the layer l
J	Objective function
$Y_j^{(l)}$	Output of neuron j in layer (l)

$P(t)$	Population at iteration t
β_i, β	Precision for class C_i, Global precision (Accuracy)
$E^{(p)}$	Quadratic error for data point p
γ_i	Sensitivity for class C_i, Global sensitivity
N_l	Size of class C_l
σ_{il}	Standard deviation of feature x_i for class C_l
$\theta_i^{(l)}$	Threshold of the i^{th} neuron in layer l
$t(x_i)$	t-statistic of feature x_i

1 Introduction

Owing to an increasing proliferation of cyber-attacks, the requirement to develop high-performance intrusion detection systems (IDS) is becoming more imperative than ever. In IDS research, the most used methodology is based on anomaly detection, where normal activity is compared with observed events in order to detect significant deviations. In recent works, most of the IDS are created on machine-learning technologies using the CICIDS2017 Dataset. As intrusion detection data usually contains numerous features, which strongly influence the learning phase, degradation of speed and performance due to overfitting can occur. Consequently, it is essential to select the best features appropriately through feature engineering.

The broad range of recently published works [1–33] has made it possible to develop IDS capable of recognizing intrusions with great efficiency. The working hypotheses are different, whether it concerns the datasets, the data reduction methods or the classification algorithms. The datasets used are diverse. Generally, those who have used CICIDS2017 have only used part of these datasets, but not always the same one. On the other hand, the attribute reduction techniques as well as the intrusion classification algorithms are very extensive and cover almost all existing ML tools. In some works, all the features were exploited, while in others, only a limited number of these attributes were used.

In addition, it can be noticed that NNs are among the most efficient methods, but there are very few works dedicated to the hybridization of these NNs with metaheuristic algorithms. Clustering methods are exploited in combination with other tools, but are little used as classification algorithms. On the other hand, Linear Discriminant Analysis (LDA) and t-statistic are considered as efficient conventional techniques for selecting the best features, but they have been little exploited. Thus, the proposed contribution through this paper, as well as its novelty, can be summarized as follows:

a) Regarding the data, CICIDS2017 Dataset was chosen to implement the proposed approach, because this dataset is realistic and exploits up-to-date attacks. Additionally, the proposed approach suggests a hybridized detection procedure of all the attack categories. However, due to the high dimension of the CICIDS2017 Dataset, only 25% of the data was used.

b) As feature selection method, LDA performance measure, as F_1-score, is used in combination with t-statistic.

c) For the detection of intrusions, unsupervised Fuzzy C-means (FCM) clustering and supervised NN techniques are chosen.

d) In addition, the classifiers (FCM and NN) are hybridized by seven most known metaheuristic methods.

e) The best approach developed will be confronted with two of the most recent datasets: CSE-CICIDS2018 and LUFlow2020.

Based on the CICIDS2017 Dataset, the goal was to implement intrusion detection solutions using FCM and Back-propagation Neural Network (BPNN) techniques that detect and classify with high accuracy each type of attack. Improved FCM and NNs by using hybrid metaheuristic algorithms are then designed. As metaheuristic methods, the seven most known algorithms, including Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Cultural Algorithm (CA), Differential Evolution (DE), Black Hole (BH) Algorithm, Harmony Search (HS), and Ant-Lion Optimizer (ALO), are chosen. To evaluate the proposed classifiers, performance metrics extracted from confusion matrices, such as precision, accuracy, sensitivity and F_1-score are employed.

The structure of this paper is as follows. Section 2 gives an overview of related work in the field of IDS. Section 3 introduces the available datasets used for the implementation and evaluation of the proposed approaches. In Section 4, a succinct description of the adopted methods and algorithms (as LDA, BPNN, K-means, FCM, PSO, GA, DE, CA, HS, BH and ALO) as well as performance measure parameters are provided. In Section 5, the experiment results on CICIDS2017 Dataset of the suggested algorithms are detailed; an evaluation of CICIDS2018 and LUFlow2020 Datasets and a comparison between state-of-the-art works and the proposed intrusion detection solutions are also outlined. Some discussions close Section 5. Finally, the main conclusions and perspectives end this paper in Section 6.

2 Related Works

Recently, Machine Learning (ML) methods have been frequently adopted in IDS, as detection or features selection tools. Ullah et al. [1] used a global approach to select significant features, oversampling and cleaning the CICIDS2017 Dataset, achieving values for specificity, precision, recall and F_1-score of 100%. Vijayanand et al. [2] proposed an IDS centered on genetic algorithm (GA) and Support Vector Machines (SVM) for feature selection and classification. To evaluate their approach, a small percentage of instances from the CICIDS2017 Dataset were used. Zhang et al. [3] performed an anomaly detection model, using Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM), which reached good classification outcomes with an accuracy of about 99%.

Binbusayyis et al. [4] suggested a feature selection method and established an IDS model by using Random Forest (RF) algorithm. The results achieve 99.88% accuracy. Ahmim et al. [5] proposed a novel IDS, conducted in CICIDS2017, using three different Machine Learning (ML) classifiers. The performance metrics showed that they achieved an accuracy rate of 96.66% and detection rate of 94.47%. Krishna et al. [6] introduced a Fast K-Nearest Neighbor Classifier (FKNN), reaching great precision with less computation time. Alrowaily et al. [7] experimented with seven ML algorithms using CICIDS2017 Dataset, showing that the K-Nearest Neighbor (K-NN) method performed better than other classifiers. Zhang et al. [8] introduced a real-time detection system, using a distributed Random Forest (RF) algorithm, implemented on the CICIDS2017 Dataset. The experimental results revealed that their suggested approach achieved higher accuracy with a shorter detection time. Using the synthetic minority oversampling technique (SMOTE), Adaptive Boosting (AdaBoost) and principal component analysis (PCA) on CICIDS2017, a F_1-score of 0.8183 was obtained [9]. In [10], a Random Forest Regressor is first used to select the best features, which is then exploited by K-NN, Multi-Layer Perceptron (MLP), Iterative Dichotomiser 3 (ID3), Adaboost, Naïve Bayes, RF and Quadratic Discriminant Analysis (QDA). The best accuracy value reached 0.98 with RF and ID3 [10].

In [11], the Fisher Score, SVM, K-NN and Decision Tree (DT) algorithms are proposed for features selection and classification tasks to design a denial of service based IDS. With KNN, DT and SVM, this IDS reached 99.7%, 57.76% and 99% success rates, respectively. Moreover, MLP and CNN classifiers are implemented on the Packet CAPture file (PCAP) of CICIDS2017 [12]. The results

attested the superiority of MLP, which distinguish network intrusion from benign traffic with an average true positive rate of 94.5%.

A Data Dimensionality Reduction method based IDS has been proposed [13], where SVM, CTree (Conditional inference Trees), XGBoost (Extreme Gradient Boosting) and Neural network (NN) classifiers using 36 selected features were evaluated. The highest accuracy achieved was 98.93% with XGboost, excluding Monday traffic (benign traffic) of the CICIDS2017 Dataset. In [14], the CICIDS2017 Dataset was classified using the DT, naïve Bayes, RF and SVM methods. During this experiment, the ten best features were extracted, allowing Naïve Bayes to provide the best result with an F_1-score of 0.92.

Studies using deep learning (DL) methods, such as deep neural network (DNN), recurrent neural network and deep reinforcement learning, have been developed [15–17]. The information Gain (IG) method is used to select appropriate and weighty features and ML algorithms as RF, Random Tree (RT) and J48 are applied in the investigation on CICIDS2017 Dataset. Specifically, RF and J48 achieve the highest accuracy values of 99.86% and 99.87%, respectively, using two different sets of selected relevant features [18].

Ferriyan et al. [19] emphasized applying GA for selecting optimal features and RF for labels classification. Vasan et al. [20] tried the convenience of PCA for intrusion detection. They found that classification accuracy is improved by PCA, by using the optimal number of principal components. Chabathula et al. [21] examined the effect of PCA, used to reduce the number of features, on many ML algorithms as Voting Features Interval, SVM, Random Forest Tree, Naïve Bayes probabilistic classifier, proving that tree algorithm yield the highest classification accuracy.

On the other hand, in IDS design, K-means and K-medoids clustering techniques have been considered in combination with other machine learning methods [22–25]. In [22], K-means clustering and KNN classifier were proposed to enhance the detection accuracy in the NSLKDD Dataset. A hybrid technique merging K-means (for data mining) and RBF kernel function of SVM (for classification) was applied to KDDCup99, achieving an accuracy of 88.70% [23]. An IDS using cluster centers and nearest neighbors proposed and applied to the KDDCup99 Dataset reached an accuracy of 99.9% [24]. In [25], anomalies were detected by using SVM classifier and k-Medoids clustering technique. Ariafar et al. [26] presented a structure for detecting network attacks by means of DT and K-means methods, where K-means parameters and DT confidence factor are adjusted by GA. Peng et al. [27] have offered a PCA and K-means-based IDS, where PCA is used to reduce the data size, and K-means are allowed to cluster the obtained data.

Thakkar et al. [28] considered, ML and DL algorithms, feature selection-based performance evaluation, on the basis of the CICIDS2017 Dataset. In [29], the performance of DNN-based IDS was enhanced by employing a novel fusion of statistical importance–based feature selection technique. A fusion of L1, L2 and elastic net regularization techniques are applied in [30] for increasing a DNN-based IDS performance, by using several datasets, including CICIDS2017.

Kanimozhi et al. [31] focused on detecting the botnet attacks in the CSE-CICIDS2018 Dataset by means of ANN, achieving an accuracy of 99.97% and a false positive rate of 0.001. In [32], Ferrag et al. showed that the DL models, based on Naïve Bayes, ANN, SVM and RF and applied to the CSE-CICIDS2018 Dataset, achieved a 95% of detection rate. In [33], SVM, KNN and DT performances were compared by using multiple datasets, including the CSE-CICIDS2018 Dataset, showing that the IDS-based accuracy ranged from 95% to 100%.

The previous state of the art allows to better situate the problem treated in this paper. The richness of works in the IDS field stimulates the attempt to improve NN and FCM by a metaheuristic-based

hybridization. CICIDS2017 is chosen as a database for implementing the proposed approach. In addition, CSE-CICIDS2018 and LUFlow2020 are adopted to assess the results obtained.

3 Datasets

This section provides an overview of the CICIDS2017, CSE-CICIDS2018 and LUFlow2020 Datasets used.

3.1 CICIDS2017 Dataset

CICIDS2017 produced by the Canadian Institute for Cyber-security IDS [34], includes up-to-date network intrusions. Additionally, CICIDS2017 meets all real-world attack criteria. This dataset, publicly available, shows the status of the network traffic, counting normal and attack traffic during five days. The dataset consists of 2830743 instances with 79 features, corresponding to 15 unbalanced traffic classes (one normal class and 14 attack traffic categories). As detailed in [35,36], this dataset is unbalanced because classes are present with an abundant number of occurrences as DDos, PortScan and Dos Hulk, and classes have few instances, such as Infiltration, Web Attack-Sql Injection and Heartbleed.

In the proposed research, only 25% of the MachineLearningCSV version of the CICIDS2017 Dataset was considered, with 707686 observations (Table 1).

Table 1: Characteristics of the used CICIDS2017 and CSE-CICIDS2018 Datasets

		CICIDS2017 Dataset			CSE-CICIDS2018 Dataset		
Class	Class label [10]	N_{Ci}	Training	Test	N_{Ci}	Training	Test
C_1	Benign	568019	397614	170405	120000	84000	36000
C_2	DoS hulk	57684	40379	17305	40000	28000	12000
C_3	PortScan	39550	27685	11865	–	–	–
C_4	DDoS	32005	22404	9601	40000	28000	12000
C_5	DoS GoldenEye	2670	1867	803	4000	2800	1200
C_6	FTP-patator	2017	1412	605	10000	7000	3000
C_7	SSH-patator	1525	1067	458	10000	7000	3000
C_8	DoS slowloris	1448	1014	434	1000	700	300
C_9	DoS slowhttptest	1336	935	401	5000	3500	1500
C_{10}	Bot	478	335	143	10000	7000	3000
C_{11}	Web attack–brute force	379	265	114	50	35	15
C_{12}	Web attack–XSS	17	11	6	25	17	8
C_{13}	Web attack–sql injection	7	5	2	7	5	2
C_{14}	Infiltration	7	4	3	1000	700	300
C_{15}	Heartbleed	4	3	1	–	–	–
All C_i		707146	495000	212146	241082	168757	72325

In the beginning, the dataset comprises 707686 observations on 79 features including category attributes. In the pre-processing step, some records containing missing values and non-regular values

are omitted. After cleaning up the initial dataset, the total number of remaining instances is 707146. The distribution of the data used is shown in Table 1. For experimental purposes, the 25% cleaned CICIDS2017 Dataset is divided with a lot of 70% for training data (with 495000 records) and 30% for testing data (212146 records), as depicted in Table 1.

The two sets of data (training and test) both consist of 15 intrusion classes (normal and attack) with high-class imbalance. The 79 features [34] will be the subject, in Subsection 5.1, of a search for the best characteristics that will be used by the different approaches proposed in this work.

3.2 CSE-CICIDS2018 Dataset

CSE-CICIDS2018 Dataset is produced as part of a collaborative project between the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC) [37]. This dataset, identical to the CICIDS2017, except for the PortScan and Heartbleed attacks, includes 13 different attack scenarios and using the same features. For evaluation purposes, a reduced lot of this dataset is used, after cleaning, with a 70%/30% ratio for the training/test sets, as shown in Table 1.

3.3 LUFlow2020 Dataset

The LUFlow Dataset [38] is one of the latest datasets for the domain of IDS, created by Lancaster University in 2020. This huge dataset consists of two classes (benign and malicious), with 16 features. Hence, only a reduced set of the samples, after cleaning, are randomly selected for this work, with training/test proportions of 70%/30%, as given in Table 2.

Table 2: Characteristics of the used LUFlow2020 Dataset

Class label	N_{Ci}	Training	Test
Benign	120000	84000	36000
Malicious	120000	84000	36000
All C_i	240000	168000	72000

4 Methodology

To train an ML algorithm, several techniques can be applied, such as supervised and unsupervised learning. A classification based on labeled data instances in the learning phase, thus defines supervised learning. LDA and NN represent two of the most well-known supervised learning methods. On the other hand, during an unsupervised learning unlabeled data instances can be grouped into classes. This clustering technique include K-means and FCM algorithms.

The use of an ML algorithm generally involves the minimization of a cost function. Metaheuristic algorithms are strategies that guide the process of finding minima. In addition, before being processed, the data must be reduced by first selecting the best features. For this purpose, LDA and t-statistic are adopted in this work. This section briefly outlines the tools used in this work as LDA and t-statistic–based selection method, FCM and NN-based algorithms, the metaheuristic algorithms and the performance measurement parameters.

4.1 LDA Principle

During a linear discriminant analysis (LDA), discriminant functions are sought [39,40]. In the simple case of 2 classes (C_1 and C_2), the expression of the unique discriminant function h_1, for a data example x, is as given in Eq. (1), where g_1 and g_2 are class centroids and V a covariance matrix

$$h_1(x) = (g_1 - g_2)^t \; V^{-1}x \tag{1}$$

This discriminant function is computed, using the training set. An intrusion instance x_0 is classified as C_1 or C_2, according to the value of $h_1(x_0)$, as proposed in System (2).

$$\begin{cases} if\ h_1(x_0) \geq 0 \quad then\ x_0 \in C_1 \\ else\ x_0 \in C_2 \end{cases} \tag{2}$$

4.2 Basics of Fuzzy C-Means (FCM) Algorithm

K-means algorithm [41] is a familiar clustering technique distributing instances into K clusters by updating the clusters centers iteratively. Similar instances are assigned to the same cluster and examples with lower similitude are arranged in separate sets. The main K-means steps are shown in Algorithm 1.

Algorithm 1: K-means algorithm

1: Set K data instances as first centroids.
2: **Repeat**
3: Arrange K clusters by allotting all data instances to the nearest center.
4: Recompute the cluster centers.
5: **Until** stabilization of centroids
6: Return K clusters positions.

With K-means, the results obtained usually depend on the initial centroids' positions. Few alternatives are proposed to answer this initialization question. The main idea of FCM is to express the membership of each instance to the different clusters by seeking to optimize the objective function [42].

At each iteration, the K cluster centroids $c_i (i \in \{1, \ldots, K\})$ are revised by using Eq. (3).

$$c_i = \frac{\sum_{j=1}^{N} {\omega_{ij}}^m x_j}{\sum_{j=1}^{N} {\omega_{ij}}^m} \tag{3}$$

where N is the number of data instances and m the fuzziness index ($m \in [1, \infty]$). ω_{ij} represents the membership of the i^{th} data to j^{th} cluster center, as given by the Eq. (4).

$$\omega_{ij} = \frac{1}{\sum_{l=1}^{K} \left(\frac{d_{ij}}{d_{il}}\right)^{\frac{2}{m-1}}} \tag{4}$$

d_{ij} is the Euclidean distance between the i^{th} data and the j^{th} cluster center.

The FCM main goal is to minimize the objective function J (Eq. (5)).

$$J(U, V) = \sum_{i=1}^{N} \sum_{j=1}^{k} {\omega_{ij}}^m {d_{ij}}^2 \tag{5}$$

The FCM steps are given in Algorithm 2.

Algorithm 2: FCM algorithm

1: Choose arbitrarily K cluster centers.
2: i = 1
3: **Repeat**
4: Calculate ω_{ij} using Eq. (4).
5: Compute c_i by Eq. (3).
6: i = i + 1
7: **Until** reaching a minimum of J or $\left\| U^{(i+1)} - U^{(i)} \right\| < u_0$, where: $u_0 \in [0,1]$ and $U = (\omega_{ij})_{N\times k}$ is the fuzzy membership matrix.
8: Return K clusters positions.

4.3 Basics of BPNN

A multilayered neural network is structured into layers of neurons: one input layer, one output layer and one or several hidden layers [39,43]. Each neuron i in a level l is directly connected to all the neurons of the previous layer $(l-1)$ and yields a response $Y_i^{(l)}$, result of a nonlinear processing through an activation function s (Eq. (6)).

$$Y_i^{(l)} = s\left(\sum_{j=1}^{N^{(l-1)}} W_{ij}^{(l)} Y_j^{(l-1)} - \theta_i^{(l)}\right) \tag{6}$$

where $\theta_i^{(l)}$ and $W_{ij}^{(l)}$ are the threshold of the neuron i in layer l and its connection weight with neurons j (in layer $l-1$), $N^{(l-1)}$ the number of neurons in the layer $l-1$.

Through supervised training, using the error back-propagation (BP) algorithm, the BPNN attempts to minimize, on the output layer, for each data p, a quadratic error $E^{(p)}$ (Eq. (7)) subsisting between the ith output effective value $o_i^{(p)}$ and the desired value $r_i^{(p)}$.

$$E = \sum_p E^{(p)} = \sum_p \sum_{i=1}^{q} \left(o_i^{(p)} - r_i^{(p)}\right)^2 \tag{7}$$

4.4 Metaheuristic Algorithms

This subsection will succinctly describe the metaheuristic algorithms adopted in this study as PSO, GA, DE, CA, HS, BH and ALO.

4.4.1 Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) is a metaheuristic algorithm founded on the swarm intelligence concept [44]. A PSO procedure starts with a random population of solutions. In the search space, the prospective solutions, called particles, have a fitness value to evaluate by the objective function, and a velocity that directs their flying [45].

At each iteration, PSO updates the velocity and position of each particle, using the two best attributes: p_{best} and g_{best}. The key points of PSO are shown in Algorithm 3.

Algorithm 3: PSO algorithm

1: Set initial population.
2: **Repeat**
3: Estimate fitness of each particle.
4: Revise p_{best}.

(Continued)

Algorithm 3: (Continued)

5: Revise g_{best}.
6: Create a new population.
7: Revise velocity.
8: Update position.
9: **Until** the end criterion is meet.

4.4.2 Genetic Algorithm (GA)

The genetic algorithm (GA), developed by Holland [46], is centered on three main operators: selection, crossover and mutation.

The GA process begins iterations with a generated random set of individuals and considered as potential solutions. That set of individuals is then tested against the objective function. The selection procedure retains then the best performing chromosomes, called parents, according to their fitness values. GA operates on a population, denoted $P(t)$, the tasks described by Algorithm 4.

Algorithm 4: GA algorithm

1: Randomly generate initial population $P(0)$.
2: $t = 1$
3: **Repeat**
4: Calculate the chromosomes fitness and remember the highest fitness value.
5: Choose parents according to the roulette wheel mechanism.
6: Crossover the parent's chromosomes.
7: Mutate the chromosomes.
8: $t = t + 1$
9: **Until** the end criterion is met.

4.4.3 Differential Evolution Algorithm (DE)

The differential evolution algorithm (DE) is a population-based metaheuristic search algorithm [47]. In DE, solutions are identified as genomes or chromosomes. DE algorithm takes on a population $P(t)$, the operations of mutation, crossover, selection and calculates the fitness value of each solution repeatedly, as illustrated by Algorithm 5.

Algorithm 5: DE algorithm

1: $t = 0$
2: Set initial $P(t)$.
3: Calculate $P(t)$ fitness.
4: **While** not termination-condition do
5: $t = t + 1$
6: For each parent $P(t)$, choose best solutions.
7: Create descendants using the DE operators.
8: Evaluate $P(t)$ fitness.
9: **End** while

4.4.4 Cultural Algorithm (CA)

CA is based on the human cultural evolution process. CA works on two spaces, including population and belief spaces [48]. The interaction between the population and belief spaces is provided through the communication protocol (defined by acceptance and influence functions). The population space (PS) of CA contains individual solutions, while the belief space (BS) represents the cultural information gained during the evolution process. The main Cultural Algorithm steps are described in Algorithm 6.

Algorithm 6: CA algorithm

1: $t = 0$
2: Initialize Population and Belief Spaces: $P(t)$, $B(t)$
3: **While** the stopping criterion is met, **do**
4: $t = t + 1$
5: Evaluate the fitness of individuals in $P(t)$
6: Accept some elite individuals from the $P(t)$ using *Accept function*
7: Update $B(t)$) using *Update function*
8: Produce new generation $P(t + 1)$ using *Influence function*
9: **End**

4.4.5 Harmony Search Algorithm (HS)

Harmony Search (HS) Algorithm is a metaheuristic search algorithm which attempts to reproduce the musicians creativity process in composing an agreeable harmony [49]. This algorithm considers harmony memory using a process similar to choosing the best-fit individuals in other metaheuristic techniques. In addition, a harmony memory-accepting factor (reaccept) is allocated. The main HS steps are given in Algorithm 7.

Algorithm 7: HS algorithm

1: Initialize randomly the first Harmony Memory (HM).
2: **Repeat**
3: Evaluate the HM.
4: Select the Best individuals from the HM.
5: Generate new Harmonics.
6: Evaluate the Harmonics.
7: Include the Best Harmonics in the HM.
8: **Until** the end criterion is met.

4.4.6 Black Hole Algorithm (BH)

The Black Hole algorithm (BH) is motivated by the phenomenon of the black hole and tries to reproduce its rule of attracting other stars in space [50]. Black holes symbolize solution with best fitness and other candidate solutions as star. All stars explore new best positions while moving towards black-hole. When a star reaches better fitness than black hole, its position is swapped. Star too close to black hole (pass event horizon) will be replaced by new random solution, as explained in Algorithm 8.

Algorithm 8: BH algorithm

1: Set randomly an initial population of stars in the search space.
2: $t = 0$
3: **While** ($t < MaxIteration$) or (stop criterion) do
4: Evaluate the fitness f_i of each star.
5: $f_{BH}>=$ Best f_i
6: **For** each i^{th} star do
7: $x_i = x_i + rand\ (x_{BH} - x_i)$
8: if $f_i > f_{BH}$ then *swap* (i^{th}, BH)
9: if *dist* (i^{th}, BH) $\leq R$ then *move* the i^{th} star to a random position, where $R = \frac{f_{BH}}{\sum_i f_i}$
10: **End** for
11: $t = t + 1$
12: **End** while

4.4.7 Ant Lion Optimization (ALO)

Ant lion optimization (ALO) algorithm uses ant lions hunting technique [51]. ALO comprises five main hunting steps: random walk, construction traps, entrapment of ants in the trap, catching prey and reconstruction traps. The main ALO steps are given in Algorithm 9.

Algorithm 9: ALO algorithm

1: Arbitrarily initialization of the first population of Ants and Ant lions.
2: Calculation of the fitness of Ants and Ant lions and determination of elite (best Ant-lion).
3: **While** ($t < MaxIteration$)
4: **For** each Ant
5: Select an Ant lion based on Roulette wheel.
6: Revise parameters.
7: Random walk creation and normalization.
8: Revise the ant position.
9: **End** for
10: Calculate the fitness of all Ants.
11: Substitute a fitter Ant lion as elite.
12: Update elite if Ant lion is better than elite.
13: $t = t + 1$
14: **End** while

4.5 Performance Metrics

Performance metrics regarding intrusion detection can be defined from a confusion matrix, such as accuracy, precisions, sensitivities and F_1-score. On the other hand, and before applying a classification method, it is essential to select the best features. The following subsections summarize how to compute these metrics.

4.5.1 Confusion Matrix Based Parameters

IDS effectiveness is estimated according to its ability to classify intrusion traffic into a correct type. Table 3 displays all possible cases of classification, where A_{ij} denote the number of examples of genuine *class* C_i categorized as *class* C_j and N_i is the size for class C_i.

Table 3: Confusion matrix

Actual class	Predicted class			
	C_1	C_2	...	C_{15}
C_1: N_1	A_{11}	A_{12}	...	$A_{1,15}$
C_2: N_2	A_{21}	A_{22}	...	$A_{2,15}$
...	...	...	...	...
C_{15}: N_{15}	$A_{15,1}$	$A_{15,2}$	...	$A_{15,15}$

To evaluate the proposed classifiers (FCM, BPNN, hybrid based FCM and hybrid based NN), two parameter sets are used. The first set includes specific metrics of each type of attack, as the Detection Rate or Precision, the True Positive Rate (recall or sensitivity), the False Decision Rate (or False Alarm Rate) and F_1-score. The second one consists of global metrics as global sensitivity, global false decision rate, global F_1-score and accuracy.

For each predicted class C_i, precision rate β_i, sensitivity rate γ_i, false decision rate δ_i and F_{1i} -score are computed according to Eq. (8).

$$\beta_i = \frac{A_{ii}}{N_i}, \quad \gamma_i = \frac{A_{ii}}{\sum_j A_{ji}}, \quad \delta_i = \frac{\sum_{j \neq i} A_{ji}}{\sum_j A_{ji}} = 1 - \gamma_i, \quad F_{1i} = \frac{2\beta_i \gamma_i}{\beta_i + \gamma_i} \tag{8}$$

Thus, the global rates as accuracy β, the global sensitivity (γ), the global false decision rate (δ) and the global F_1-score, become as specified in Eq. (9).

$$\beta = \frac{\sum_i N_i \beta_i}{\sum_i N_i}, \quad \gamma = \frac{\sum_i N_i \gamma_i}{\sum_i N_i}, \quad \delta = 1 - \gamma, \quad F_1 = \frac{\sum_i N_i F_{1i}}{\sum_i N_i} \tag{9}$$

It is also essential to note that for all the metaheuristic based approaches (hybridized FCM and NNs), and to maximize accuracy β, global sensitivity γ and F_1-score (or to minimize False Decision Rate δ), the identical fitness function f was considered (Eq. (10)).

$$f = 1 - F_1 \tag{10}$$

4.5.2 T-Statistic Based Features Ranking

As feature selection technique, the t-statistic method [52] is applied in this work. For each characteristic x_i, a univariate Discriminant Function is calculated on data belonging a priori to two classes C_j and C_l. t-statistic, $t(x_i)$ is obtained according to Eq. (11).

$$t(x_i) = \frac{|\mu_{ij} - \mu_{il}|}{\sqrt{\frac{\sigma_{ij}^2}{N_j} + \frac{\sigma_{il}^2}{N_l}}} \tag{11}$$

where μ_{ik}, σ_{ik} and N_k signify respectively the mean, the standard deviation of i^{th} feature x_i and the size for class C_k, for $k = \{j, l\}$. The best features are those with the highest t-statistic values.

5 Implementation of the Proposed Methodology

As earlier presented, the aim of this work is to improve the detection of intrusions in 25% of CICIDS2017 Dataset, by using a hybridized detection process of all the attack families. To this end, this work was carried out in three main steps:

i) To select the best features, a combination between F_1-score, extracted from LDA, and t-statistic method was adopted.

ii) For the detection of intrusions, unsupervised Fuzzy C-means (FCM) and hybrid FCM clustering using seven metaheuristic algorithms (GA, PSO, CA, DE, HS, BH and ALO) are performed after the designation of the finest number of clusters by Elbow technique.

iii) To improve performance parameters of intrusion detection, supervised NNs (a Back-propagation Neural Network and seven hybrid NNs, using the above-cited metaheuristic algorithms) are achieved.

To finalize this analysis, two further steps are achieved:

iv) The best approaches, proven during the two previous steps, are reassessed using data from CICDS2018 and LUFlow2020 Datasets.

v) Performance regarding the suggested approaches and state of the art works are displayed.

In this section, the different steps of the suggested approach are detailed. In addition, the overall results are discussed.

Notice that the various analyzes achieved in this work have been developed under the Matlab environment, R2017b, with an Intel Core i7 3.0 GHz processor, 16 GB of RAM.

5.1 Features Selection

As introduced above, the used CICIDS2017 Dataset contains observations on 79 features counting category attribute [34]. In this work, and to select the best features, two techniques are used, namely F_1-score extracted from LDA and t-statistic.

Indeed, a univariate discriminant analysis was developed, allowing to calculate the Discriminant Functions (DF) by considering one variable x_i at a time and a couple of classes at a time as well. This allowed to establish the confusion matrix for each DF (and for each feature), and to deduce the F_1-score value (F_{1i}).

In the same way, the t-statistic function $t(x_i)$ was calculated, for all the features x_i and for all class pairs, as given in Eq. (11). Thus, considering all classes, each feature x_i has an average value for F_{1i}, noted (F_{1mi}) and for $t(x_i)$, noted $t_m(x_i)$. The sum ($0.5\ F_{1m\,i} + 0.5\ t_m(x_i)$) is used to rank the features in decreasing order. Thus, the 24 best uncorrelated attributes, ranked by decreasing value of the sum ($0.5\ F_{1m\,i} + 0.5\ t_m(x_i) \geq 0.5$) on the basis of the previously obtained results and for all intrusion classes, are illustrated in Table 4.

To test the discriminating power of these 24 selected features to detect normal traffic or attacks, a complete LDA is performed. Thus, by exploiting the 24 variables listed in Table 4, and the first nine most frequent classes (C_1 to C_9) of Table 1, eight discriminant functions, DF, (h_1, h_2, …, h_8) were calculated for eight pairs of classes. To carry out the designs, we chose 1200 instances per intrusion class and a decision rule as given in System (12).

$$\begin{cases} if\ h_1(x) \geq 0 \quad then\ x \in C_1 \\ else\ if\ h_i(x) \geq 0 \quad then\ x \in C_i\ for\ i = \{2, 3, \ldots, 7\} \\ else\ if\ h_8(x) \geq 0 \quad then\ x \in C_8 \\ else\ x \in C_9 \end{cases} \tag{12}$$

Table 4: The 24 selected features for CICIDS2017 Dataset

Rank	1	2	3	4	5	6	7	8	9	10	11	12
Feature ID [10]	41	13	65	63	66	12	40	42	18	39	52	67
Rank	13	14	15	16	17	18	19	20	21	22	23	24
Feature ID [10]	54	20	14	8	55	9	22	26	25	24	36	21

The best DFs allow to establish the confusion matrix, from which the performance rates (precisions β_i, sensitivities γ_i and F_{1i}) are extracted and illustrated in Fig. 1. In the search for the best DFs, various combinations of the 24 features are operated. The precisions of these DFs as a function of the number of features they contain are shown in Fig. 1.

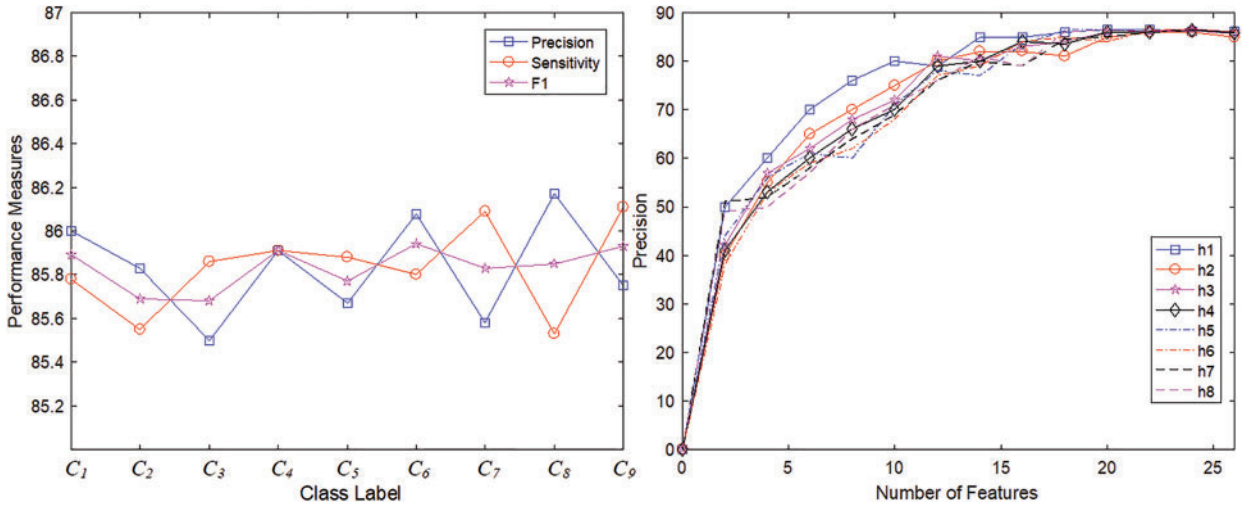

Figure 1: Performance measures obtained by LDA

These results show acceptable precisions, sensitivities as well as F_1-scores, with an average value of these three parameters around 85.83%, because the selected features are sufficient and relevant to characterize the attack patterns. In addition, the experiment results illustrate the fact that through LDA, identification of the different classes is possible with reasonable accuracy and global F_1-score (improvable by non-linear methods like those proposed in the following subsections).

5.2 FCM and Hybrid FCM Based Detection of Intrusion

The considered data (707146 intrusion instances of Table 1, split into a training set (70%) and test set (30%), with 24 features listed in Table 4) will be classified using FCM and hybridized FCM. Let us start first by determining the exact number of clusters existing in the used CICIDS2017 Dataset by the elbow method.

5.2.1 *Finest Number of Clusters (K) Using Elbow Method*

Before clustering the data using K-means or FCM methods, it is necessary to know the exact number of clusters (K) present in the dataset. The elbow method gives an interpretation to the evolution of the variance regarding the number of clusters [53]. In fact, Within-Cluster Sum of Square (WCSS) is calculated as a function of the number of clusters (K), WCSS is the sum of the squared distance between each instance and the cluster centroid (Fig. 2). According to Fig. 2, the curves (obtained by K-means and FCM clustering designs) decrease to the elbow point ($K = 13$–15), from which the curve decreases no more or very little. This observation brings us closer to the actual number of classes ($K = 15$), contained in the used CICIDS2017 Dataset. However, from clustering point of view, if the class size is negligible (as for classes C_{12}–C_{15} in Table 1), it is immersed by the other classes. Finding $K \sim 13$ is not surprising; this reflects one of the intrusion classes' realities.

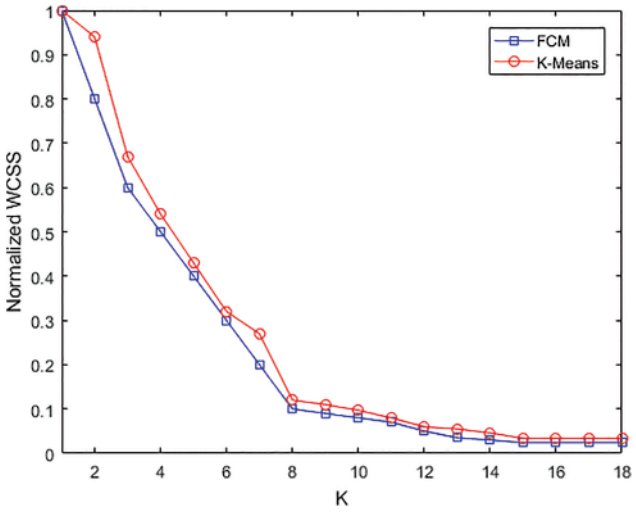

Figure 2: Within clusters sum of squared distances (WCSS) *vs. K*

5.2.2 *FCM Based Detection of Intrusion*

Each instance is labeled according to the type of intrusion to which it belongs. Since FCM is an unsupervised procedure, the instance label is not used during clustering, but as validation information and to calculate performance metrics. Intrusion data defined by the 24 most informative features, as defined in Table 4, are considered as input for the FCM algorithm. According to Algorithm 2, all data in the training set are grouped into 15 clusters ($K = 15$). After an initial random choice of 15 clusters centers, FCM assigns to each intrusion instance a membership score for each of the clusters. By minimizing the adopted objective function, FCM updates, at each iteration, clusters centers and membership levels for each instance. FCM ends by producing a list of cluster centers and multiple membership levels. This allows for establishing the confusion matrix, based on the knowledge of the real membership of each instance, and from which the performance parameters are deduced and reported in Table 5.

Table 5: Unsupervised detection performance (Accuracy β, global sensitivity γ, global F_1-score F_1 and execution time, in seconds) for FCM and Hybrid FCM

	Unsupervised detection performance		
Method	β (%)	γ (%)	F_1 (%)
FCM	86.95 ± 1.03	87.03 ± 0.89	86.91 ± 1.04
PSO-FCM	$\mathbf{90.86 \pm 1.01}$	$\mathbf{90.37 \pm 1.11}$	$\mathbf{90.76 \pm 0.91}$
GA-FCM	$\mathbf{90.19 \pm 0.94}$	$\mathbf{89.64 \pm 1.03}$	$\mathbf{89.88 \pm 1.01}$
DE-FCM	89.76 ± 0.96	89.05 ± 1.07	89.38 ± 1.04
CA-FCM	89.56 ± 1.09	89.11 ± 0.92	89.27 ± 1.01
HS-FCM	89.36 ± 0.97	88.80 ± 1.13	89.14 ± 0.99
BH-FCM	88.93 ± 1.06	88.83 ± 0.91	88.81 ± 1.05
ALO-FCM	$\mathbf{90.18 \pm 0.88}$	$\mathbf{90.01 \pm 0.96}$	$\mathbf{90.04 \pm 0.97}$

5.2.3 Hybrid FCM Based Detection of Intrusion

Metaheuristic algorithms as PSO, GA... are powerful and stochastic non-linear optimization tools. To find the optimum fuzzy partitions of intrusion types, a new metaheuristic based FCM clustering method has been proposed. Clustering using PSO-FCM, GA-FCM, DE-FCM, CA-FCM, HS-FCM, BH-FCM and ALO-FCM can be achieved using the Algorithm 10.

Algorithm 10: Metaheuristic based FCM algorithm

1: Setting the FCM clustering parameters (as the number of clusters K, fuzzy index m...)
Setting metaheuristic algorithms parameters (population size, number of runs, maximum of iterations, fitness function and other factors as given in Table 6).
2: Initialize the population $P(0)$, corresponding to $N \times K$ cluster centroids.
3: Computing Fitness function of each K clusters combination.
4: Best-fitted solutions are selected after having being developed according to the PSO, GA, DE, ... or ALO algorithm (Algorithms 3–9 described in Subsection 4.4).
5: Check the end condition, if satisfied then stop, otherwise go to Step 3.
6: The finest K clusters generated by the metaheuristic algorithm correspond to global optimized cluster centers.
7: These K clusters represent an initialization and will be used for final processing according to FCM Algorithm (Algorithm 2 outlined in Subsection 4.2).

This algorithm comprises two parts. In the first step, the K clusters ($K = 15$) are developed according to the metaheuristic algorithm concerned by using the tuning parameters exposed in Table 6. The best result obtained is then readapted, during the second stage, according to FCM clustering Algorithm.

Table 6: Algorithms parameters tuning

Algorithm	Parameters tuning
All algorithms	Number of features = 24, Population size = 10–100, Maximum number of iterations = 1000, Number of runs = 50, Fitness function: $f = 1 - F_1$ (Eq. (10))
PSO	Personal learning coefficient = 1.49, Global learning coefficient = 1.49, Inertia weight damping ratio = 0.99, Inertia weight = 0.72
GA	Crossover probability = 0.8, Mutation rate = 0.02, Mutation probability = 0.3, Selection pressure = 8
DE	Differential Mutation factor = 0.75, Crossover probability = 0.85,
CA	Constant parameter = 0.2, Acceptance rate = 0.35
HS	Pitch-adjusting rate = 0.3, Harmony memory rate = 0.95, Fret width or bandwidth = 0.2
BH	Translation probability = 0.75, Translation parameter = 0.25
ALO	Number of search agents = 1000

As observed for FCM design, the instance label is not used during clustering, but as validation information and to calculate confusion matrices and their corresponding performance metrics as achieved and illustrated in Table 5 and Fig. 3. FCM and Hybrid FCM being unsupervised approaches design, only the CICIDS2017 training Dataset was used, acting also as a test set. Thus, the results provided in Table 5 represent the test performance rates (Accuracy β, global sensitivity γ and global F_1-score).

Because the results of the proposed algorithms depend on random initializations, the FCM-based approaches were repeated 50 times. The results illustrated in Table 5 represent the mean values with the standard deviations of the performance parameters obtained. These results can be shortened as follows. The best accuracy rates are (in %) 87.98, 91.87, 91.13, 90.72, 90.53, 90.33, 89.99 and 91.06 for FCM, PSO-FCM, GA-FCM, DE-FCM, CA-FCM, HS-FCM, BH-FCM and ALO-FCM, respectively. The best global sensitivity rates are (in %) 87.92, 91.48, 90.67, 90.12, 90.03, 89.93, 89.74 and 90.97 for the previous eight FCM-based algorithms, respectively, while the best global F_1 values reach (in %) 87.95, 91.67, 90.89, 90.42, 90.28, 90.13, 89.86 and 91.01 for the eight FCM-based algorithms mentioned above, respectively.

The best performance rates obtained are summarized in Fig. 3. As proved by these last results, PSO-FCM, GA-FCM, ALO-FCM and DE-FCM outweigh the rest of the FCM-based algorithms. Fig. 3 presents also the best precisions, sensitivities and F_1-scores of these last 4 algorithms, with regard to the intrusion classes C_1 to C_{15} (as defined in Table 1).

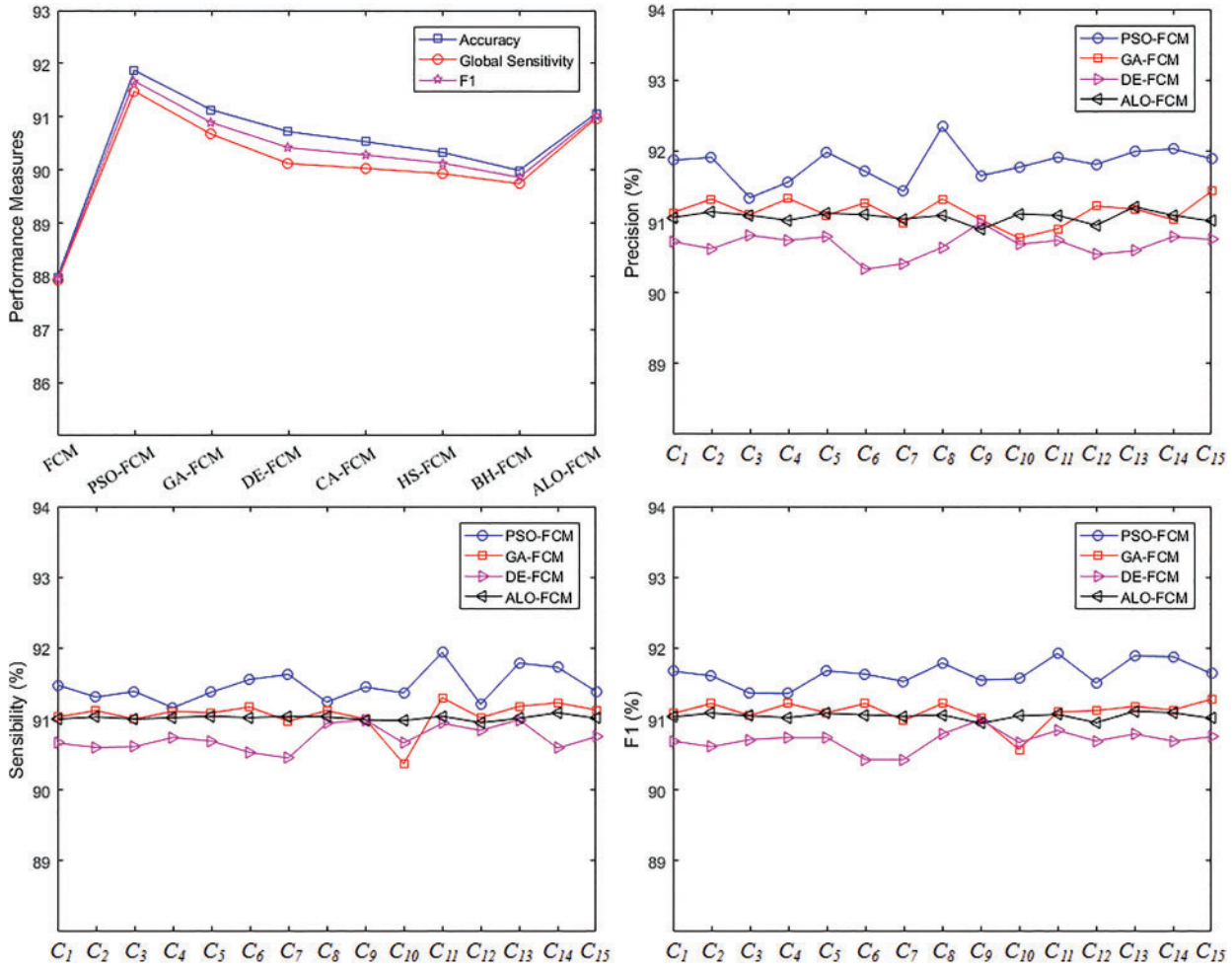

Figure 3: Best performance measures for FCM-based algorithms and intrusion categories (C_1 to C_{15} are attack classes defined in Table 1)

5.3 BPNN and Hybrid NN-Based Detection of Intrusion

As dealt with in the previous section, the considered CICIDS 2017 intrusions will be detected using BPNN, followed by processing through seven metaheuristic-based NN as expressed in Algorithm 11 and based on parameters shown in Table 6. However, before proceeding with the detection of these intrusions, it would be wise to seek the best NN architecture.

5.3.1 Optimal NN Search

It should be noted that regardless of the adopted architecture, the NN model consists of one input layer including 24 neurons, corresponding to selected features in Table 4. The input layer is followed by several hidden layers, whose exact number and sizes are to be determined. The output layer consists of 15 neurons, where each neuron is dedicated to a class. The desired response r_i of the i^{th} output neurons ($i = 1, \ldots, 15$), for each instance x, is coded according to Eq. (13).

$$\begin{cases} r_i(x) = 1 \quad for \quad x \in C_i \\ r_i(x) = -1 \quad for \quad x \notin C_i \end{cases} \tag{13}$$

For every intrusion instance x_0, a decision rule is proposed, for the 15 intrusion classes, depending on the ith neural output o_i ($i = 1, \ldots, 15$) as specified in System (14).

$$\begin{cases} if \quad o_i(x_0) \geq 0 \quad then \quad x_0 \in C_i \\ else \quad x_0 \notin C_i \end{cases} \tag{14}$$

In the search of the best NN, all structures with 1 to six hidden layers, containing between 10 and 100 neurons each, are considered.

During the BPNN classification, the NN weights and thresholds are computed using the formulation given in Subsection 4.3. In the case of hybrid-NN using GA, PSO, DE, BH, CA, HS and ALO, the weights and thresholds are the metaheuristic components to be handled until the end of the iterations. Additionally, runs with 10 to 1000 iterations and populations ranging from 10 to 100 individuals were explored.

Algorithm 11: BPNN and metaheuristic-NN implementation

1: Adoption of NN Architecture.
2: For the BPNN case, update weights and thresholds according to the formulation given in Subsection 4.3, using the training dataset.
3: Return the results of BPNN with weights, threshold and BPNN training performances (precision, sensitivity, F_1-score and accuracy of classifications).
4: Create the confusion matrix-based on the test set, compute test performances.
5: For the hybrid NN (GA-NN, PSO-NN, CA-NN, HS-NN, ALO-NN, BH-NN or DE-NN), improve thresholds and weights based on the corresponding metaheuristic procedure.
6: Repeat Steps 3 and 4 for each metaheuristic-based NN and return training and test performances (precision, sensitivity, F_1-score and accuracy).

The key points to note, in NN designs, are the reliance of accuracy rates with the NN size (regardless of the neurons spreading in the network), the training number of iterations and the population's size. The Accuracy-Number of neurons dependency is shown in Fig. 4, where accuracy rates are schemed regarding the number of hidden nodes for BPNN and PSO-NN. This first result is probed thanks to architectures with 5 hidden layers, using BPNN and PSO-NN (with 200 iterations and a population size of only 40 individuals). After a lot of experimentation, and as shown in Fig. 4, the finest architecture attained is (24, 40, 60, 40, 20, 30, 15).

Regarding the contribution of the Number of Iterations in improving accuracies, Fig. 4, produced on the same NN architecture (24, 40, 60, 40, 20, 30, 15), shows that accuracies can be improved by 3–4 points if the number of iterations is increased to 1000. It can be perceived that the PSO-NN, GA-NN and ALO-NN algorithms are more efficient.

Fig. 4 illustrates the best accuracy rates found for different population's size values and produced on the same NN architecture (24, 40, 60, 40, 20, 30, 15). The last result again explains the advantage of the PSO-NN approach. Moreover, satisfactory results are provided with an average population of 50 individuals.

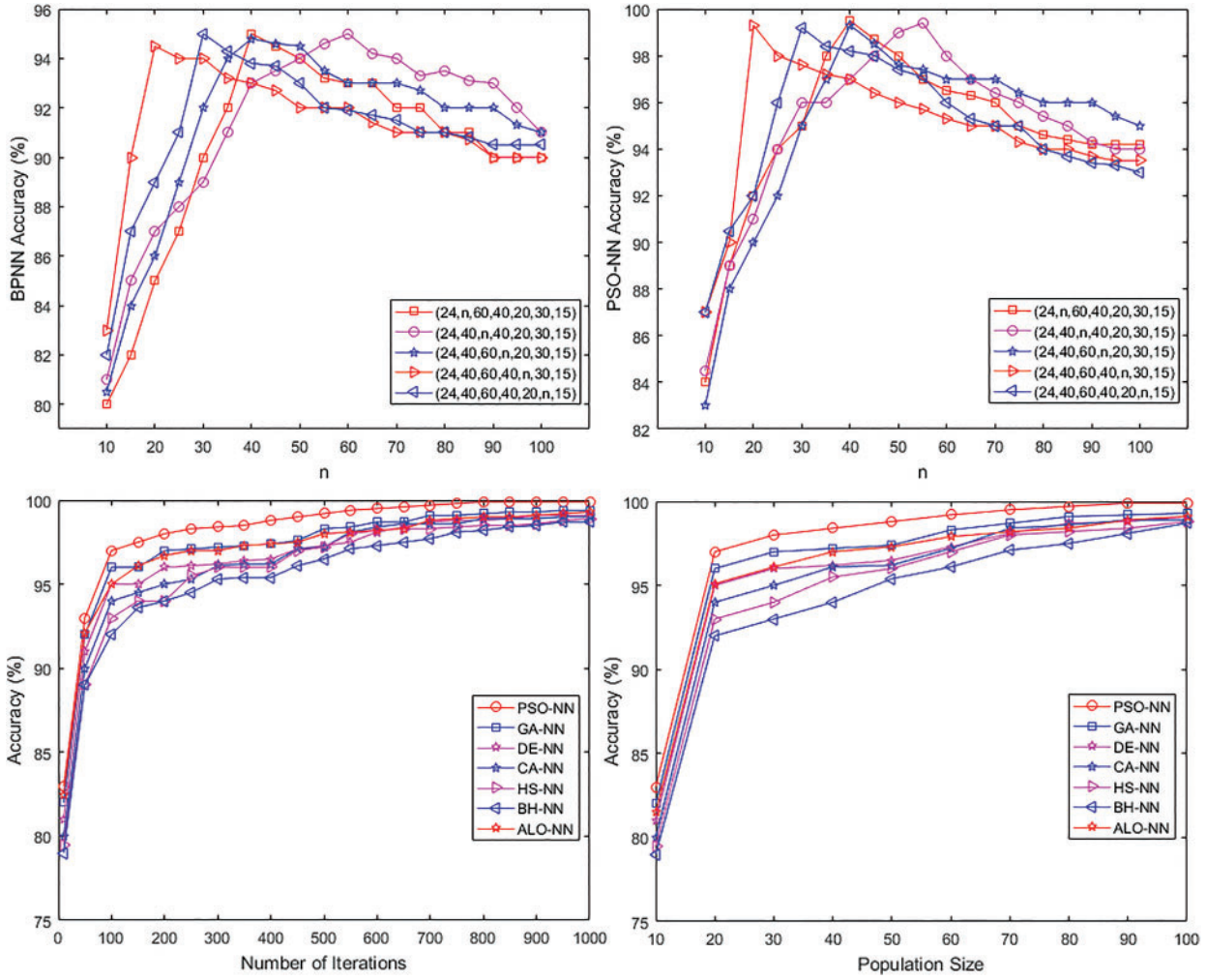

Figure 4: Accuracies *vs*. number of hidden neurons *n* for BPNN and PSO-NN and accuracy *vs*. number of iterations and accuracy *vs*. population's size

5.3.2 Intrusion Detection by Using Hybrid NN

Because the proposed metaheuristic algorithms are initialized in a random way, the hybrid NN-based approaches were re-executed 50 times. Notice that, for each run, a confusion matrix is established allowing to extract the global results. The best results on the 50 adopted runs, obtained with the eight NN-based algorithms for both training and testing datasets, are shown in Table 7.

The best training performance measures by the eight NN-based algorithms can be summarized as follows. The best training accuracy rates correspond (in %) to 96.88, 99.99, 99.56, 99.13, 98.87, 98.97, 99.12 and 99.25 for BPNN, PSO-NN, GA-NN, DE-NN, HS-NN, BH-NN and ALO-NN, respectively. The best global training sensitivity rates are (in %) 96.34, 99.97, 99.45, 99.09, 98.68, 99.13, 98.94 and 99.41 for the previous eight NN-based algorithms, respectively. The best training F_1-values cover the range (in %) of 96.11, 99.98, 99.50, 99.11, 98.77, 99.05, 99.03 and 99.33 for the eight NN-based algorithms mentioned above, respectively.

Table 7: Intrusion detection performance by using BPNN and metaheuristic-based NN on training and test datasets

	Training performance rates			Test performance rates		
Method	β (%)	γ (%)	F_1 (%)	β (%)	γ (%)	F_1 (%)
BPNN	95.76 ± 1.12	95.41 ± 0.93	95.13 ± 0.98	95.77 ± 0.97	95.74 ± 1.09	95.75 ± 1.03
PSO-NN	**99.10 ± 0.89**	**98.99 ± 0.98**	**99.04 ± 0.94**	**98.96 ± 1.01**	**99.02 ± 0.93**	**98.99 ± 0.97**
GA-NN	**98.49 ± 1.07**	**98.47 ± 0.98**	**98.44 ± 1.06**	**98.52 ± 0.99**	**98.19 ± 1.07**	**98.35 ± 1.03**
DE-NN	**98.22 ± 0.91**	**98.07 ± 1.02**	**98.14 ± 0.97**	**98.03 ± 0.98**	**97.82 ± 1.11**	**97.93 ± 1.04**
CA-NN	97.71 ± 1.16	97.70 ± 0.98	97.69 ± 1.08	97.64 ± 1.08	97.68 ± 1.01	97.66 ± 1.04
HS-NN	98.03 ± 0.94	98.14 ± 0.99	98.08 ± 0.97	98.00 ± 0.98	97.92 ± 0.99	97.96 ± 0.98
BH-NN	98.09 ± 1.03	97.93 ± 1.01	98.01 ± 1.02	98.01 ± 1.02	97.87 ± 1.05	97.94 ± 1.03
ALO-NN	**98.26 ± 1.09**	**98.49 ± 0.92**	**98.42 ± 1.91**	**98.24 ± 0.99**	**98.06 ± 1.05**	**98.15 ± 1.02**

Additionally and in conjunction with the four best NN-based algorithms, the greatest test accuracy rates are (in %) 99.97, 99.51, 99.01 and 99.23 for PSO-NN, GA-NN, DE-NN and ALO-NN, respectively. The highest global test sensitivity rates reach the following values (in %): 99.95, 99.26, 98.93 and 99.11 for the previous four best NN-based algorithms, respectively. The highest test F_1-values extend over the interval (in %) of 99.96, 99.38, 98.97 and 99.17 for the four best NN-based algorithms mentioned above, respectively.

Fig. 5 displays the best test global performance measures attained by the eight NN-based algorithms and examples of test precision, sensibility and F_1-score rates, obtained by the four best NN-based algorithms (PSO-NN, GA-NN, ALO-NN and DE-NN) with respect to intrusion category.

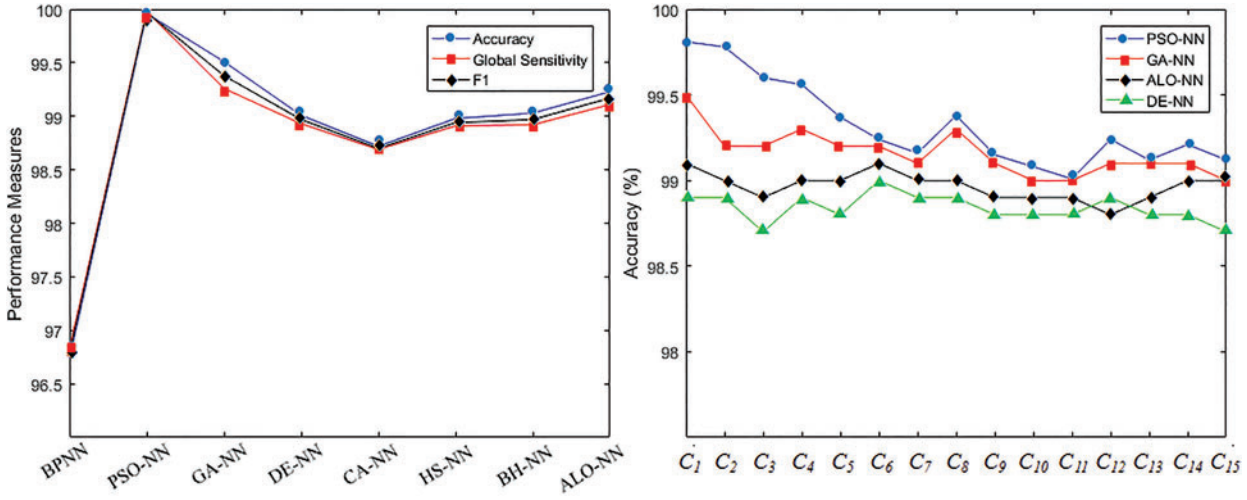

Figure 5: (Continued)

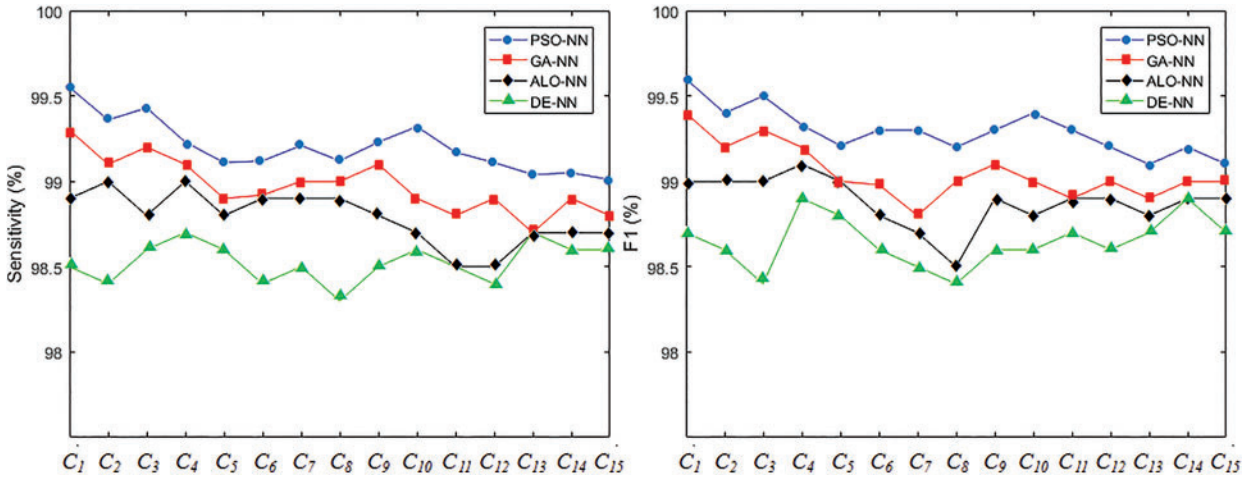

Figure 5: Test performance measures with respect to NN-based algorithms and intrusion detection performance comparing of NN-based algorithms for accuracy, sensitivity and F_1-score

As shown by the previous results, PSO-NN is the best algorithm among the eight metaheuristic-based NNs. Details about the distribution of PSO-NN performance rates with respect to the 15 intrusion classes, given in Table 8, for training and test datasets, show excellent performance, situating thus accuracies, sensitivities and F_1 rates, for all classes, between 99.32% and 99.99%.

Table 8: Detection performance rates for all CICIDS2017 classes by using PSO-NN

	Training performance rates			Testing performance rates		
Attack class	β (%)	γ (%)	F_1 (%)	β (%)	γ (%)	F_1 (%)
C_1	99.04 ± 0.95	98.94 ± 1.03	99.01 ± 0.97	99.00 ± 0.98	98.90 ± 1.05	99.01 ± 0.94
C_2	98.94 ± 1.04	99.05 ± 0.89	98.92 ± 1.04	99.04 ± 0.92	99.05 ± 0.90	99.01 ± 0.93
C_3	98.90 ± 1.07	98.98 ± 0.96	98.95 ± 0.99	99.09 ± 0.88	98.90 ± 1.04	98.95 ± 0.99
C_4	98.92 ± 1.05	98.99 ± 0.95	99.00 ± 0.93	98.93 ± 1.03	99.01 ± 0.94	99.02 ± 0.93
C_5	98.97 ± 0.92	99.05 ± 0.89	98.86 ± 1.02	98.96 ± 0.94	99.01 ± 0.95	98.90 ± 1.05
C_6	99.04 ± 0.92	98.84 ± 1.04	98.92 ± 0.88	98.89 ± 0.97	99.05 ± 0.88	99.05 ± 0.84
C_7	98.98 ± 0.90	99.02 ± 0.89	98.84 ± 1.04	98.91 ± 0.94	98.93 ± 1.04	98.98 ± 0.93
C_8	98.74 ± 1.04	98.94 ± 0.87	98.75 ± 1.05	98.61 ± 0.87	99.02 ± 0.93	98.81 ± 0.96
C_9	98.73 ± 1.04	98.90 ± 0.94	98.89 ± 0.89	98.74 ± 1.05	98.94 ± 0.98	98.68 ± 1.02
C_{10}	98.71 ± 1.02	98.75 ± 0.87	98.59 ± 1.04	98.83 ± 0.95	99.02 ± 0.89	98.83 ± 0.93
C_{11}	98.82 ± 0.94	98.93 ± 0.88	98.46 ± 1.05	98.61 ± 1.02	99.03 ± 0.89	98.76 ± 1.03
C_{12}	98.76 ± 1.03	98.73 ± 0.92	98.79 ± 0.94	98.51 ± 1.03	99.03 ± 0.90	98.79 ± 0.91
C_{13}	98.68 ± 0.89	98.58 ± 0.91	98.48 ± 1.02	98.37 ± 0.95	98.88 ± 1.04	98.75 ± 0.88
C_{14}	98.86 ± 0.92	98.60 ± 1.03	98.76 ± 0.93	98.73 ± 0.88	98.98 ± 0.91	98.85 ± 0.91
C_{15}	98.48 ± 1.04	98.62 ± 0.89	98.52 ± 0.97	98.60 ± 1.02	98.91 ± 0.89	98.68 ± 1.03

The execution times, achieved offline during the learning phase, for FCM-based algorithms and NN-based techniques with (24, 40, 60, 40, 20, 30, 15) architecture, using 1000 iterations, are collected in Table 9. From Table 9 and previous results, the PSO-NN approach provides the best efficiency-computational cost.

Table 9: Average run times in seconds for all FCM and NNs approaches (learning with 1000 iterations)

Method	FCM	PSO-FCM	GA-FCM	DE-FCM
CPU time (s)	4.31	201.34	264.67	297.35
Method	CA-FCM	HS-FCM	BH-FCM	ALO-FCM
CPU time (s)	362.54	432.76	356.82	1399.45
Method	BPNN	PSO-NN	GA-NN	DE-NN
CPU time (s)	254.14	645.15	592.75	579.77
Method	CA-NN	HS-NN	BH-NN	ALO-NN
CPU time (s)	756.34	732.11	472.45	1956.32

5.4 Approach Evaluation by Using CSE-CICIDS2018 and LUFlow2020 Datasets

After ensuring that the PSO, GA and ALO-based FCM and NN have achieved good accuracy, the models are tested using the CSE-CICIDS2018 [37] and LUFlow2020 Datasets [38].

Regarding the CSE-CICIDS2018 Dataset, the training and test data given in Table 1 were used. As this dataset has the same features as the CICIDS2017 Dataset, the same selected attributes (Table 4) are retained.

For the LUFlow2020 Dataset, the training and testing of the proposed approaches are performed on the data set in Table 2. The features of this dataset have been reduced from 16 to 10, based on the same features selection approach employed for the CICIDS2017 Dataset, described in Subsection 5.1. The features selected are shown in Table 10.

Table 10: Features selected for the LUFlow2020 Dataset

Rank	1	2	3	4	5	6	7	8	9	10
Feature ID [38]	4	7	11	2	8	15	12	10	6	9

For the hybridization of FCM or NN by the three metaheuristics chosen (PSO, GA and ALO), the same procedures defined in Algorithms 10 and 11 are used and keep the same tuning parameters of Table 6. The results obtained during the training phase, with 1000 iterations and after 50 runs, are summarized in Table 11. The best test performance rates are shortened in Table 12. Based on these results, PSO-NN, GA-NN and ALO-NN, as well as the FCM-based approaches, keep the same efficiencies as those reached for CICIDS2017. Moreover, PSO-NN and GA-NN show excellent test accuracies, reaching 99.77 % and 99.31%, respectively, for the CSE-CICIDS2018 Dataset, (99.97% and 99.72%, respectively for the LUFlow2020 Dataset).

Table 11: Training results using PSO-, GA- and ALO-based FCM and NN

	CSE-CICIDS2018 Dataset			LUFlow Datasets		
Method	β (%)	γ (%)	F_1 (%)	β (%)	γ (%)	F_1 (%)
PSO-FCM	90.52 ± 1.04	90.31 ± 1.01	90.46 ± 0.96	92.16 ±1.05	91.97 ± 1.03	92.04 ± 0.93
GA-FCM	90.21 ± 0.98	89.76 ± 1.02	90.11 ± 1.03	91.69 ± 0.96	91.13 ± 1.08	91.38 ± 1.01
ALO-FCM	90.29 ± 1.02	90.19 ± 0.97	90.23 ± 1.03	91.24 ± 0.99	90.99 ± 1.07	91.13 ± 1.04
	NN architecture			NN architecture		
	(24, 40, 60, 40, 20, 30, 13)			(10, 35, 45, 40, 20, 2)		
	β (%)	γ (%)	F_1 (%)	β (%)	γ (%)	F_1 (%)
PSO-NN	99.05 ± 0.91	98.92 ± 0.99	99.01 ± 0.95	98.97 ± 1.01	99.05 ± 0.94	99.01 ± 0.97
GA-NN	98.65 ± 1.01	98.39 ± 0.97	98.46 ± 1.01	98.56 ± 0.98	98.53 ± 1.05	98.45 ± 1.05
ALO-NN	98.53 ± 1.05	98.41 ± 0.98	98.47 ± 1.04	98.43 ± 0.97	98.28 ± 1.03	98.33 ± 1.04

These results are illustrated in another way, with the Receiver Operating Characteristic (ROC) curves, given in Fig. 6. The area under the curve (AUC) values are also provided in Table 12, once again proving the correctness of the proposed approach.

Table 12: Best test performance rates and AUC values using PSO-, GA- and ALO- based FCM and NN algorithms

	CSE-CICIDS2018 Dataset				LUFlow2020 Datasets			
Method	β (%)	γ (%)	F_1 (%)	AUC (%)	β (%)	γ (%)	F_1 (%)	AUC (%)
PSO-FCM	91.44	90.98	91.21	91.11	91.86	91.37	91.76	91.82
GA-FCM	91.11	90.71	90.97	90.87	91.19	90.64	90.97	90.67
ALO-FCM	90.97	90.41	90.63	90.34	90.86	90.79	90.83	90.54
PSO-NN	99.77	99.65	99.73	99.64	99.97	99.98	99.97	99.95
GA-NN	99.31	98.96	99.24	99.26	99.72	99.61	99.69	99.58
ALO-NN	99.11	98.91	99.02	99.03	99.52	99.32	99.43	99.37

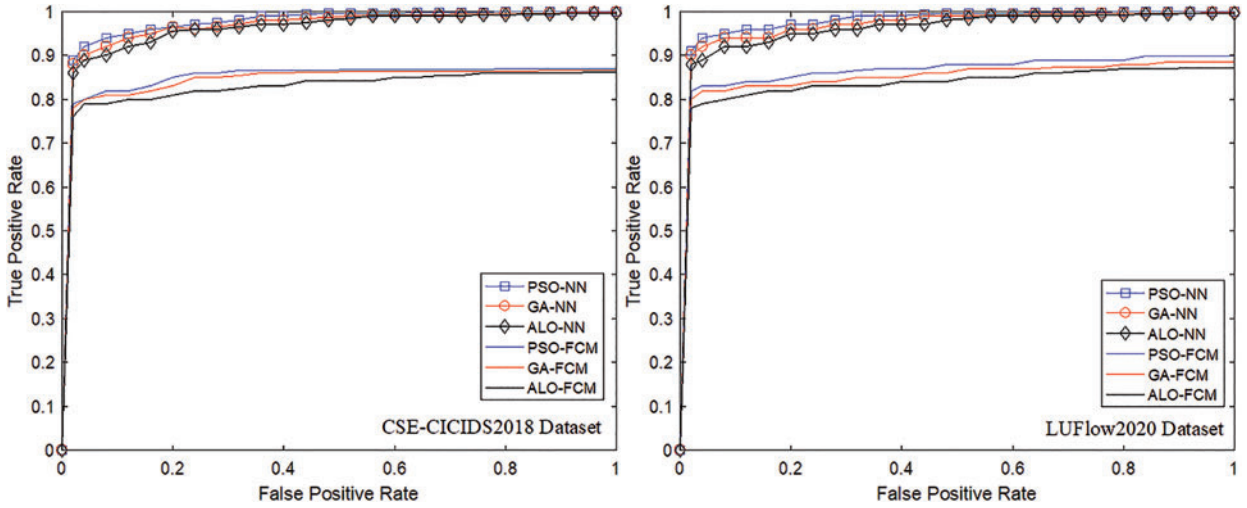

Figure 6: ROC curves for FCM- and NN-based approaches using CSE-CICIDS2018 and LUFlow2020 Datasets

5.5 Comparison with State-of-the-Art Works

Performance evaluation regarding the suggested approaches and published works [9,54–69] is displayed in Table 13. Despite the fact that the comparison with related work is not a straight forward process, the established results show that the three algorithms PSO-NN, GA-NN and ALO-NN are very efficient in the intrusion detection task. Moreover, PSO-NN algorithm provides improved performance than state-of-the-art studies concerning accuracy (β), sensibility (γ) and F_1-score.

Table 13: Comparison with previous works on CICIDS2017, CSE-CICIDS2018 and LUFlow2020 Datasets

Reference	Year	Method	Dataset	β (%)	γ (%)	F_1 (%)
[9]	2019	Adaboost		81.83	100	90.01
[54]	2019	AE and PCA		99.60	98.80	98.80
[55]	2020	IGAN-IDS		84.45	84.45	84.17
[56]	2021	NSA		–	97.00	97.00
[57]	2019	RF		96.20	–	–
[58]	2019	LSTM, CNN	CICIDS2017	98.00	–	–
[59]	2021	IG + RF (1)		99.83	99.80	99.90
[59]	2021	IG + RF (2)		99.79	99.90	99.90
[60]	2022	RF		99.92	99.92	99.91
[61]	2022	LDA		88.10	88.33	89.21
[61]	2022	MLP		99.10	99.79	99.11
Proposed approach	2022	**PSO-NN**	CICIDS2017	**99.97**	**99.95**	**99.96**
		GA-NN		99.51	99.26	99.38
		ALO-NN		99.13	99.11	99.12

(Continued)

Table 13 (continued)

Reference	Year	Method	Dataset	β (%)	γ (%)	F_1 (%)
[62]	2019	DBN	CSE- CICIDS2018	95.00	–	–
[63]	2020	DNN		90.25	–	–
[64]	2020	DL		95.00	–	–
[65]	2019	LSTM		96.20	–	–
[66]	2019	DL		96.00	–	–
[67]	2019	CNN		96.00	–	–
[68]	2021	HCRNN		97.75	–	–
[60]	2022	RF		99.04	99.04	98.83
Proposed approach	2022	**PSO-NN**	CSE- CICIDS2018	**99.77**	**99.65**	**99.73**
		GA-NN		99.31	98.96	99.24
		ALO-NN		99.11	98.91	99.02
[69]	2022	RF	LUFlow2020	99.94	–	–
		ANN		99.60	–	–
		DNN		99.82	–	–
Proposed approach	2022	PSO-NN	LUFlow2020	**99.97**	**99.98**	**90.98**
		GA-NN		99.72	99.61	99.69
		ALO-NN		99.52	99.32	99.43

5.6 Discussion

In this subsection, the results obtained are commented on by raising issues related to the interpretation of the False decision rate, the evaluation of the approaches proposed by recent datasets, the comparison with state-of-the-art works, and the implementation and deployment of real-time IDS.

5.6.1 General Comments

Firstly, it is important to remember that the results provided are articulated around the three parameters accuracy, sensitivity and F_1-score. The results of the fourth parameter (the false decision rate) can be deduced from the sensitivity (its complement). On the basis of the experimental outcomes (Table 5), by means of training the CICIDS2017 Dataset, with the 24 selected features (Table 4), FCM and Hybrid FCM algorithms can detect all attack categories with acceptable performance rates. Indeed, the detection accuracy of FCM-based algorithms increases from 88% for FCM to 91.87%, 91.13% and 91.06% for PSO-FCM, GA-FCM and ALO-FCM, respectively. The same observation is made for sensitivity measurements, where this parameter rises from 87.92% for FCM to 91.48%, 90.67% and 90.97% for ALO-FCM, GA-FCM and ALO-FCM, respectively.

The performance of BPNN and metaheuristic-based NNs in detecting intrusions, as presented in Table 7 (for the training and test used ICICIDS2017 data), greatly improved accuracy, sensitivity and F_1 rates. The three performance parameters range from 96% for BPNN to 99.99% for PSO-NN. The accuracy of PSO-NN approach in detecting intrusion families is excellent, reaching a value of 99.99% using the training Dataset and 99.97% using testing dataset. In the same way, the test-sensitivity and test-F_1 rates are worth 99.95% and 99.96%, respectively. Since the PSO-NN based overall sensitivity

is 99.95%, the false decision rate is 0.05%. In other words, out of a decision regarding 212146 test-instances (defined in Table 1), only 106 intrusions would be false alarms.

It should be noticed that the NN-based results are linked to the decision rules established in (System 14). Thus to reduce the False decision rate, these decision rules must be modified by exploiting the histograms of the 15 PSO-NN training responses. For example, by exploiting the histogram of o_4 (the output reserved to the C_4 class or the DDos intrusion category), shown in Fig. 7, and calculated for all the training instances, the DDos detection sensitivity obtained by PSO-NN can reach the value of 100% (and the false detection rate can be 0%), if the rule defined in (System 15) is applied.

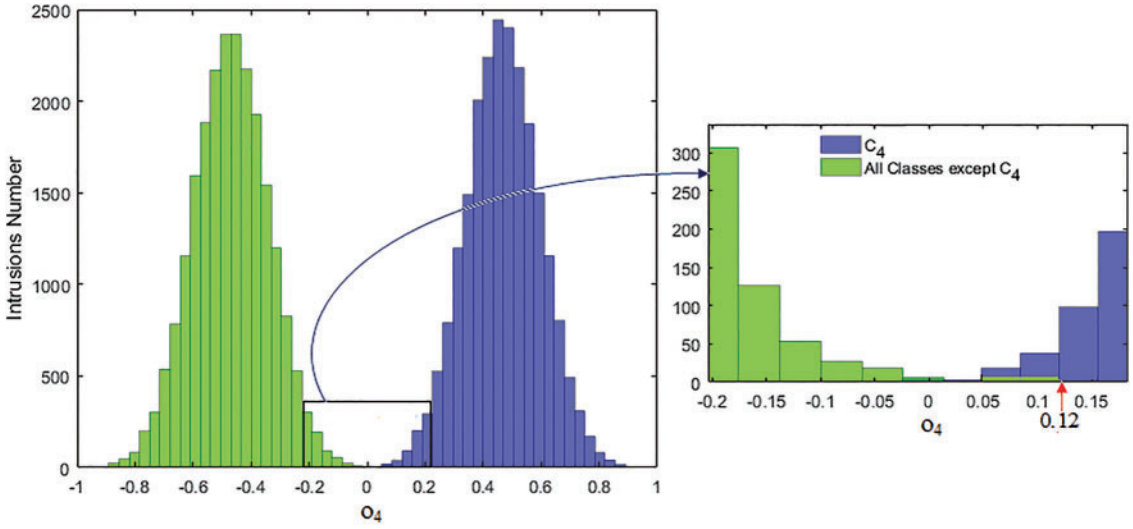

Figure 7: Distribution of intrusions with respect to the PSO-NN output o_4 (by using 22400 C_4-training instances and 22400 training instances belonging to *All* C_i, $(i \neq 4)$

$$\begin{cases} if \quad o_4(x_0) \geq 0.12 \quad then \quad x_0 \in C_4 \\ else \quad x_0 \notin C_4 \end{cases} \tag{15}$$

Thus by applying the rules (14 and 15) to 9600 test instances belonging to C_4 (DDos class) and 9600 test instances randomly combining all other classes C_i, $i \neq 4$ (all classes except DDos), detection rates were obtained as illustrated in Table 14. PSONN's results for the 15 intrusion classes, provide a sensitivity of 99.95% for DDos-intrusions. The Sensitivity (or False detection) rates concerning the detection of DDos-intrusions thus go from 99.95% (or 0.05%) with the initial rule given in System 14 to 100% (or 0%) with the modified rule (System 15).

Table 14: Test detection rates for C_4 or DDos intrusions obtained by PSO-NN algorithm

	Predicted class according to rule Eq. (14)		Predicted class according to rule Eq. (15)	
Actual class	C_4	*All* C_i, $i \neq 4$	C_4	*All* C_i, $i \neq 4$
C_4: 9600	9595	5	9595	5
All C_i, $i \neq 4$: 9600	5	9595	0	9600

The same technique can be generalized for all PSO-NN outputs o_i $(i = 1, \ldots, 15)$ to reduce, or even nullify the C_i False decision rates.

The evaluation of approaches based on PSO, GA and ALO on the CSE-CICIDS2018 and LUFlow2020 Datasets shows their efficiency. The results obtained with LUFlow2020 even exceed those obtained with CICIDS2018. This can be interpreted by the fact that in LUFFlow2020, the number of attack classes (equal to 2) allows PSO-NN algorithm to learn features better.

Regarding the comparison with the state-of-the-art works, Table 13 allows appreciating the efficiency of the proposed approach. The results obtained generally exceed those obtained in the recent works [9,54–69], and this is for the three datasets considered (CICIDS2017, CSE-CICIDS2018 and LUFlow2020).

The majority of the ML published methods are valid for a network-based IDS implementation. Existing datasets also provide a reference for the intrusions categories that may be encountered. Updating these datasets is more than imperative. Moreover, merging these datasets would be a great asset, at least those described by the same features. The interest in selecting the best features is well established. Today, a variety of features selection techniques exists, combining conventional and new approaches or introducing novel concepts [1,20,28–30,56,60]. This research axis is predestined for further development.

5.6.2 Real Time Deployment

IDS is not a laboratory matter only since most techniques proven by research teams end up in products delivered by service providers and commercials.

Despite the abundance of publications in the field of IDS, the treatment of certain key points related to the deployment and real-time implementation of network-based IDS remains rare. Indeed, few documents address this issue [70–74].

For the deployment, several solutions exist and must be adapted to the network environment. For efficient use of IDS, the combination of both a network- and host-based IDS is recommended. A network-based IDS secures the network by monitoring all traffic on specific segments, while a host-based IDS is deployed on devices.

From a professional point of view, the problem of detection time by an IDS does not arise, given the panoply of existing information technology solutions, regardless of the size of intrusion packets. The plethora of commercial IDS solutions can analyze and record network data packets in real-time, as reported for several IDS deployment [70–74].

As indicated in Table 9, training an IDS algorithm consumes a lot of computation time because for NN optimization requirements, for example, the number of iterations, the size of the NN, the parameters of the metaheuristics must be appropriate to achieve the expected performance. For an architecture of (24, 40, 60, 40, 20, 30, 15), PSO-NN requires more than 645 s for its optimization (with 1000 iterations, population size equal to 100 and 495000 instances). Running an IDS, like the test operation, consumes very little time. For the same PSONN architecture, on average, more than 76500 intrusions can be processed per second.

For a 10 Gbps link, PSO-NN-based IDS will give near real-time results as it can be combined with threads. Threads and asynchronized calculation are essential to maintain all the traffic passed to PSONN, where detections are instant and do not need a powerful CPU or a large memory.

6 Conclusion

This paper aims to improve FCM- and NN-based Intrusion Detection performance on CICIDS2017 Dataset by using the seven most known metaheuristic algorithms. To achieve this objective, the proposed approach focused on the following major directions:

a) First of all, 25% used in this dataset have been cleaned and the attributes reduced to 24 features only, as a result of a combination of LDA performance parameter with t-statistic. The interest in the selected features has also been demonstrated through the results obtained.

b) New hybridizations of FCM and NN are suggested and applied for the first time in intrusion detection research.

c) To confirm the truthfulness of the proposed approach, several performance indices as accuracy, sensibility, and F_1-score are exploited.

The unsupervised approach to detect intrusions via FCM, showed its validity, especially after hybridization with PSO, GA, and ALO. Through PSO-FCM, for example, the accuracy, global sensitivity, and F_1 have reached the values of 91.87%, 91.48%, and 91.67%, respectively.

On the other hand, NNs and their metaheuristic-based hybridizations have greatly improved the detection of intrusions in the CICDS2017 Dataset. PSO-NN offered tested accuracy, sensitivity and F_1-score of 99.97%, 99.95%, and 99.96%, respectively, improving hybrid FCM and other hybrid NN.

d) The best approaches based on PSO, GA, and ALO methods are reassessed using intrusion instances from CICDS2018 and LUFlow2020 Datasets. The results obtained further confirmed the correctness of the proposed approaches.

e) On the other hand, the comparison of the proposed approaches with the state-of-the-art works proved that PSO-NN outperforms some recently published results and this is for the three datasets considered.

Although the training of the IDS algorithms requires a long calculation time, the real-time processing of the attacks is instantaneous and does not generally require special hardware or additional memory.

Future studies should extend the work to the following key points:

i) Data mining: data may be reduced by assembling attack classes into similar groups or by merging several datasets, simplifying the analysis and further increasing performance rates. Moreover, a Dataset is better when it is described with a minimum of very relevant features. This can be achieved by sightseeing new tools.

ii) IDS design: other Artificial Intelligence-based methods, not yet sufficiently covered, can be combined or hybridized.

iii) In addition to improving the performance of IDS by ML methods, more interest should be given to the implementation and deployment aspects.

Funding Statement: The authors received no specific funding for this study.

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

References

1. Ullah, I., Mahmoud, Q. H. (2019). A two-level hybrid model for anomalous activity detection in IOT networks. *Proceedings of the 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pp. 1–6. Las Vegas, NV, USA.
2. Vijayanand, R., Devaraj, D., Kannapiran, B. (2018). Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection. *Computers & Security, 77(2),* 304–314. https://doi.org/10.1016/j.cose.2018.04.010
3. Zhang, Y., Chen, X., Jin, L., Wang, X., Guo, D. (2019). Network intrusion detection: Based on deep hierarchical network and original flow data. *IEEE Access, 7,* 37004–37016. https://doi.org/10.1109/ACCESS.2019.2905041
4. Binbusayyis, A., Vaiyapuri, T. (2019). Identifying and benchmarking key features for cyber intrusion detection: An ensemble approach. *IEEE Access, 7,* 106495–106513. https://doi.org/10.1109/ACCESS.2019.2929487
5. Ahmim, A., Maglaras, L., Ferrag, M. A., Derdour, M., Janicke, H. (2018). A novel hierarchical intrusion detection system based on decision tree and rules-based models. arXiv:1812.09059v1.
6. Krishna, K. V., Swathi, K., Rao, B. B. (2020). A novel framework for NIDS through fast kNN classifier on CICIDS2017 Dataset. *International Journal of Recent Technology and Engineering, 8(5),* 3669–3675.
7. Alrowaily, M., Alenezi, F., Lu, Z. (2019). Effectiveness of machine learning based intrusion detection systems. *Proceedings of the International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*, pp. 277–288. Atlanta, GA, USA.
8. Zhang, H., Dai, S., Li, Y., Zhang, W. (2018). Real-time distributed random-forest-based network intrusion detection system using apache spark. *Proceedings of the IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, pp. 1–7. Orlando, FL, USA.
9. Yulianto, A., Sukarno, P., Suwastika, N. A. (2019). Improving AdaBoost-based intrusion detection system (IDS) performance on CICIDS2017 Dataset. *Journal of Physics: Conference Series, 1192,* 12–18.
10. Sharafaldin, I., Lashkari, A. H., Ghorbani, A. A. (2018). Toward generating a new intrusion detection Dataset and intrusion traffic characterization. *Proceedings of the Fourth International Conference on Information Systems Security and Privacy*, pp. 108–116. Funchal, Madeira, Portugal.
11. Aksu, D., Üstebay, S., Aydin, M. A., Atmaca, T. (2018). Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm. *Proceedings of the International Symposium on Computer and Information Sciences*, pp. 141–149. Berlin, Germany, Springer.
12. Watson, G. (2018). *A comparison of header and deep packet features when detecting network intrusions (Technical Report)*. University of Maryland: College Park, MD, USA.
13. Bansal, A. (2018). *DDR scheme and LSTM RNN algorithm for building an efficient IDS (Master's Thesis)*. Thapar Institute of Engineering and Technology, Punjab, India.
14. Abdulrahman, A. A., Ibrahem, M. K. (2018). Evaluation of DDoS attacks detection in a CICIDS2017 Dataset based on classification algorithms. *Iraqi Journal of Information and Communication Technology, 1(3),* 49–55. https://doi.org/10.31987/ijict.1.3.40
15. Niyaz, Q., Sun, W., Javaid, A. Y., Alam, M. (2016). A deep learning approach for network intrusion detection system. *EAI Endorsed Transactions on Security and Safety, 3(9),* 21–26.
16. Yin, C., Zhu, Y., Fei, J., He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access, 5,* 21954–21961. https://doi.org/10.1109/ACCESS.2017.2762418
17. Tang, C., Luktarhan, N., Zhao, Y. (2020). SAAE-DNN: Deep learning method on intrusion detection. *Symmetry, 12(10),* 1695. https://doi.org/10.3390/sym12101695
18. Kurniabudi, K., Stiawan, D., Darmawijoyo, D., Bin Idris, M. Y., Bamhdi, A. M. et al. (2020). CICIDS-2017 Dataset feature analysis with information gain for anomaly detection. *IEEE Access, 8,* 132911–132921. https://doi.org/10.1109/ACCESS.2020.3009843

19. Ferriyan, A., Thamrin, A. H., Takeda, K., Murai, J. (2017). Feature selection using genetic algorithm to improve classification in network intrusion detection system. *Proceeding of the International Electronics Symposium on Knowledge Creation and Intelligent Computing*, pp. 46–49. Surabaya, Indonesia.
20. Vasan, K. K., Surendiran, B. (2016). Dimensionality reduction using principal component analysis for network intrusion detection. *Perspectives in Science, 8(1),* 510–512. https://doi.org/10.1016/j.pisc.2016.05.010
21. Chabathula, K. J., Jaidhar, C. D., Kumara, M. A. A. (2015). Comparative study of principal component analysis based intrusion detection approach using machine learning algorithms. *Proceedings of the 3rd International Conference on Signal Processing, Communication and Networking*, pp. 1–6. Chennai, India.
22. Shapoorifard, H., Shamsinejad, P. (2017). Intrusion detection using a novel hybrid method incorporating an improved KNN. *International Journal of Computer Applications, 173(1),* 5–9. https://doi.org/10.5120/ijca2017914340
23. Ravale, U., Marathe, N., Padiya, P. (2015). Feature selection based hybrid anomaly intrusion detection system using K means and RBF kernel function. *Procedia Computer Science, 45,* 428–435. https://doi.org/10.1016/j.procs.2015.03.174
24. Lin, W. C., Ke, S. W., Tsai, C. F. (2015). CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-Based Systems, 78(1),* 13–21. https://doi.org/10.1016/j.knosys.2015.01.009
25. Chitrakar, R., Chuanhe, H. (2012). Anomaly detection using support vector machine classification with k-medoids clustering. *Proceedings of the Third Asian Himalayas International Conference on Internet*, pp. 1–5. Kathmundu, Nepal.
26. Ariafar, E., Kiani, R. (2018). Intrusion detection system using an optimized framework based on datamining techniques. *Proceedings of the IEEE 4th International Conference in Knowledge-Based Engineering and Innovation KBEI*, pp. 785–791. Teheran, Iran.
27. Peng, K., Leung, V. C. M., Huang, Q. (2018). Clustering approach based on mini batch K-means for intrusion detection system over big data. *IEEE Access, 6,* 11897–11906. https://doi.org/10.1109/ACCESS.2018.2810267
28. Thakkar, A., Lohiya, R. (2022). A survey on intrusion detection system: Feature selection, model, performance measures, application perspective, challenges, and future research directions. *Artificial Intelligence Review, 55(1),* 453–563. https://doi.org/10.1007/s10462-021-10037-9
29. Thakkar, A., Lohiya, R. (2023). Fusion of statistical importance for feature selection in deep neural network-based intrusion detection system. *Information Fusion, 90(1),* 353–363. https://doi.org/10.1016/j.inffus.2022.09.026
30. Thakkar, A., Lohiya, R. (2021). Analyzing fusion of regularization techniques in the deep learning-based intrusion detection system. *International Journal of Intelligent Systems, 6(12),* 7340–7388. https://doi.org/10.1002/int.22590
31. Kanimozhi, V., Jacob, T. P. (2019). Artificial intelligence based network intrusion detection with hyperparameter optimization tuning on the realistic cyber Dataset CSE-CIC-IDS2018 using cloud computing. *Proceedings of the IEEE International Conference on Communication and Signal Processing (ICCSP'2019)*, pp. 33–36. Melmaruvathur, India.
32. Ferrag, M. A., Maglaras, L., Moschoyiannis, S., Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications, 50(1),* 102419. https://doi.org/10.1016/j.jisa.2019.102419
33. Kilincer, I. F., Ertam, F., Sengur, A. (2021). Machine learning methods for cyber security intrusion detection: Datasets and comparative study. *Computer Networks, 188,* 107840. https://doi.org/10.1016/j.comnet.2021.107840
34. CICIDS2017 Dataset. https://www.unb.ca/cic/datasets/ids-2017.html

35. Pelletier, Z., Abualkibash, M. (2020). Evaluating the CICIDS-2017 Dataset using machine learning methods and creating multiple predictive models in the statistical computing language R. *International Research Journal of Advanced Engineering and Science, 5(2),* 187–191.
36. Panigrahi, R., Borah, S. (2018). A detailed analysis of CICIDS2017 Dataset for designing intrusion detection systems. *International Journal of Engineering and Technology, 7,* 479–482.
37. CSE-CIC-IDS2018 Dataset. https://www.unb.ca/cic/datasets/ids-2018.html
38. LUFlow2020 Dataset. https://www.kaggle.com/datasets/mryanm/luflow-network-intrusion-detection-data-set
39. Mjahed, M. (2005). Higgs search at LHC by neural networks. *Nuclear Physics B, 140,* 799–801. https://doi.org/10.1016/j.nuclphysbps.2004.11.263
40. Li, C., Wang, B. (2014). *Fisher linear discriminant analysis, lectures notes*. Boston: Northeastern University.
41. Praveen, P., Rama, B. (2017). A K-means clustering algorithm on numeric data. *International Journal of Pure and Applied Mathematics, 117(7),* 157–164.
42. Balasko, B., Abonyi, J., Feil, B. (2005). Fuzzy clustering and data analysis toolbox: For use with MATLAB. *Math Works*, pp. 1–74. Veszprem, Hungary.
43. Haykin, S. (2009). *Neural networks and learning machines*, 3rd edition. Upper Saddle River, New Jersey: Pearson Education, Inc.
44. Kennedy, J., Eberhart, R. C., Shi, Y. (2001). *Swarm intelligence*. San Francisco, CA: Morgan Kaufmann Publishers Inc.
45. Rini, P., Shamsuddin, M., Yuhaniz, S. (2011). Particle swarm optimization: Technique, system and challenges. *International Journal of Computer Applications, 14(1),* 19–27. https://doi.org/10.5120/1810-2331
46. Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.
47. Price, K. V., Storn, R. M., Lampinen, J. A. (2005). *Differential evolution: A practical approach to global optimization*. Berlin: Springer.
48. Reynolds, R. (1994). An introduction to cultural algorithms. *Proceedings of the Third Annual Conference on Evolutionary Programming*, pp. 131–139. San Diego, California, USA.
49. Gao, X. Z., Govindasamy, V., Xu, H., Wang, X., Zenger, K. (2015). Harmony search method: Theory and applications. *Computational Intelligence and Neuroscience, 2(2),* 1–10. https://doi.org/10.1155/2015/258491
50. Hatamlou, A. (2013). Black hole: A new heuristic optimization approach for data clustering. *Information Sciences, 222,* 175–184. https://doi.org/10.1016/j.ins.2012.08.023
51. Mirjalili, S. (2015). The ant lion optimizer. *Advances in Engineering Software, 83,* 80–98. https://doi.org/10.1016/j.advengsoft.2015.01.010
52. Zhou, N., Wang, L. (2007). A modified T-test feature selection method and its application on the hap map genotype data. *Genomics, Proteomics & Bioinformatics, 5(3–4),* 242–249. https://doi.org/10.1016/S1672-0229(08)60011-X
53. Bholowalia, P., Kumar, A. (2014). EBK-means: A clustering technique based on elbow method and K-means in WSN. *International Journal of Computer Applications, 105(9),* 17–24.
54. Abdulhammed, R., Musafer, H., Alessa, A., Faezipour, M., Abuzneid, A. (2019). Features dimensionality reduction approaches for machine learning based network intrusion detection. *Electronics, 8(3),* 322. https://doi.org/10.3390/electronics8030322
55. Huang, S., Lei, K. (2020). IGAN-IDS: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks. *Ad Hoc Networks, 105(6),* 102177. https://doi.org/10.1016/j.adhoc.2020.102177
56. Hosseini, S., Seilani, H. (2021). Anomaly process detection using negative selection algorithm and classification techniques. *Evolving Systems, 12(3),* 769–778. https://doi.org/10.1007/s12530-019-09317-1

57. Bindra, N., Sood, M. (2019). Detecting DDoS attacks using machine learning techniques and contemporary intrusion detection dataset. *Automatic Control and Computer Science, 53(5),* 419–428. https://doi.org/10.3103/S0146411619050043
58. Lee, J., Kim, J., Kim, I., Han, K. (2019). Cyber threat detection based on artificial neural networks using event profiles. *IEEE Access, 7,* 165607–165626. https://doi.org/10.1109/ACCESS.2019.2953095
59. Kurniabudi, K., Stiawan, D., Darmawijoyo, D., Bin Idris, M. Y., Kerime, B. et al. (2021). Important features of CICIDS-2017 Dataset for anomaly detection in high dimension and imbalanced class dataset. *Indonesian Journal of Electrical Engineering and Informatics, 9(2),* 498–511. https://doi.org/10.52549/ijeei.v9i2.3028
60. Zhang, Y., Zhang, H., Zhang, B. (2022). An effective ensemble automatic feature selection method for network intrusion detection. *Information, 13(7),* 314. https://doi.org/10.3390/info13070314
61. Rosay, A., Cheval, E., Carlier, F., Leroux, P. (2022). Network intrusion detection: A comprehensive analysis of CIC-IDS2017. *Proceedings of the 8th International Conference on Information Systems Security and Privacy (ICISSP 2022)*, pp. 25–36. Vienna, Austria.
62. Wei, P., Li, Y., Zhang, Z., Hu, T., Li, Z. et al. (2019). An optimization method for intrusion detection classification model based on deep belief network. *IEEE Access, 7,* 87593–87605. https://doi.org/10.1109/ACCESS.2019.2925828
63. Farhan, R. I., Abeer, T. M., Nidaa, F. H. (2020). Optimized deep learning with binary PSO for intrusion detection on CSE-CIC-IDS2018 Dataset. *Journal of Al Qadisiyah for Computer Science and Mathematics, 12(3),* 16–27.
64. Farhan, R. I., Abeer, T. M., Nidaa, F. H. (2020). Performance analysis of flow-based attacks detection on CSE-CIC-IDS2018 Dataset using deep learning. *Indonesian Journal of Electrical Enginering and Computer Science, 20(3),* 16–27. https://doi.org/10.11591/ijeecs.v20.i3.pp1413-1418
65. Lin, P., Ye, K., Xu, C. Z. (2019). Dynamic network anomaly detection system by using deep learning techniques. In: *Lecture notes in computer science*, pp. 161–176. Switzerland: Springer Science and Business.
66. Zhou, Q., Pezaros, D. (2019). Evaluation of machine learning classifiers for zero-day intrusion detection—an analysis on CIC-AWS-2018 Dataset. arXiv:1905.03685v1.
67. Kim, J., Shin, Y., Choi, E. (2019). An intrusion detection model based on a convolutional neural network. *Journal of Multimedia Information System, 6(4),* 165–172. https://doi.org/10.33851/JMIS.2019.6.4.165
68. Khan, M. A. (2021). HCRNNIDS: Hybrid convolutional recurrent neural network-based network intrusion detection system. *Processes, 9(5),* 834. https://doi.org/10.3390/pr9050834
69. Chua, T. H., Salam, I. (2022). Evaluation of machine learning algorithms in network-based intrusion detection system. arXiv:2203.05232.
70. Yuwono, D. T. (2022). Analysis performance intrusion detection system in detecting cyber-attack on apache web server. *IT Journal Research and Development, 6(2),* 169–178. https://doi.org/10.25299/itjrd.2022.7853
71. Ujjan, R. M. A., Pervez, Z., Dahal, K. (2020). Snort based collaborative intrusion detection system using blockchain in SDN. *Proceedings of the 13th IEEE International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, pp. 1–7. Island of Ulkulhas, Maldives.
72. Jaw, E., Wang, X. (2022). A novel hybrid-based approach of snort automatic rule generator and security event correlation (SARG-SEC). *Peer Journal of Computer Science, 8(1),* 1–37. https://doi.org/10.7717/peerj-cs.900
73. Saputra, F. A., Salman, M., Hasim, J. A. N., Nadhori, I. U., Ramli, K. (2022). The next-generation NIDS platform: Cloud-based snort NIDS using containers and big data. *Big Data Cognitive Computing, 6(19),* 1–15. https://doi.org/10.3390/bdcc6010019
74. Gupta, A., Sharma, L. S. (2020). Performance analysis and comparison of snort on various platforms. *International Journal of Computer Information Systems and Industrial Management Applications, 10,* 23–32.