sensors

MDPI

Article

Enhancing Intrusion Detection Systems for IoT and Cloud Environments Using a Growth Optimizer Algorithm and Conventional Neural Networks

Abdulaziz Fatani [1,2], Abdelghani Dahou [3], Mohamed Abd Elaziz [4,5,6,7], Mohammed A. A. Al-qaness [8], Songfeng Lu [9,10,*], Saad Ali Alfadhli [11] and Shayem Saleh Alresheedi [12]

1 School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
2 Computer Science Department, Umm Al-Qura University, Makkah 24381, Saudi Arabia
3 Faculty of Computer Sciences and Mathematics, Ahmed Draia University, Adrar 01000, Algeria
4 Department of Mathematics, Faculty of Science, Zagazig University, Zagazig 44519, Egypt
5 Artificial Intelligence Research Center (AIRC), Ajman University, Ajman 346, United Arab Emirates
6 Department of Artificial Intelligence Science and Engineering, Galala University, Suze 435611, Egypt
7 Department of Electrical and Computer Engineering, Lebanese American University, Byblos 13-5053, Lebanon
8 College of Physics and Electronic Information Engineering, Zhejiang Normal University, Jinhua 321004, China
9 Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China
10 Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen 518057, China
11 Department of Computer Techniques Engineering, Imam Al-Kadhum College, Baghdad 10081, Ira
12 War College, National Defense University, Riyadh 12211, Saudi Arabia
* Correspondence: lusongfeng@hust.edu.cn

Abstract: Intrusion detection systems (IDS) play a crucial role in securing networks and identifying malicious activity. This is a critical problem in cyber security. In recent years, metaheuristic optimization algorithms and deep learning techniques have been applied to IDS to improve their accuracy and efficiency. Generally, optimization algorithms can be used to boost the performance of IDS models. Deep learning methods, such as convolutional neural networks, have also been used to improve the ability of IDS to detect and classify intrusions. In this paper, we propose a new IDS model based on the combination of deep learning and optimization methods. First, a feature extraction method based on CNNs is developed. Then, a new feature selection method is used based on a modified version of Growth Optimizer (GO), called MGO. We use the Whale Optimization Algorithm (WOA) to boost the search process of the GO. Extensive evaluation and comparisons have been conducted to assess the quality of the suggested method using public datasets of cloud and Internet of Things (IoT) environments. The applied techniques have shown promising results in identifying previously unknown attacks with high accuracy rates. The MGO performed better than several previous methods in all experimental comparisons.

Keywords: metaheuristics; cyber security; intrusion detection system; Internet of Things (IoT); Growth Optimizer; CNNs

check for updates

Citation: Fatani, A.; Dahou, A.; Abd Elaziz, M.; Al-qaness, M.A.A.; Lu, S.; Alfadhli, S.A.; Alresheedi, S.S. Enhancing Intrusion Detection Systems for IoT and Cloud Environments Using Growth Optimizer Algorithm and Conventional Neural Networks. *Sensors* **2023**, *23*, 4430. https://doi.org/10.3390/s23094430

Academic Editor: Stefan Poslad

Received: 12 February 2023
Revised: 7 April 2023
Accepted: 18 April 2023
Published: 30 April 2023

1. Introduction

The need to secure online data, information, and related systems has grown in importance with the development of information communication, particularly the Internet. Sensitive data is created, transported, stored, or updated worldwide daily in enormous quantities. Private emails, financial transactions, simple holiday photos, and military communications are all examples of sensitive information. Malicious parties have sought to steal, alter, or erase this information for a long time. Hackers and other hostile actors have developed, exploited, and enhanced various cyberattacks to accomplish these objectives [1].

A paradigm shift from straightforward defense mechanisms to complex defense systems was necessary for this new era of cyber security. While simple network security measures such as firewalls may have been enough in the past, the sophistication of cyberattacks has made them ineffective when used alone. Intrusion Detection Systems (IDS) are currently the cornerstone of cyber security to defend against these sophisticated attacks [1]. In the cloud and IoT, there are three primary divisions of cloud services: infrastructure-as-a-service (IaaS), platform-as-a-service (PaaS), and software-as-a-service (SaaS). To give users security, it is necessary to address the weaknesses and problems each of these services and methods possesses [2]. In recent years, different methods have been proposed for the IDS, such as the traditional machine learning techniques, for example, the support vector machine (SVM) [3,4], decision trees [5,6], k-means clustering [7,8], and others. The recent advances in deep neural networks, including conventional neural networks (CNNs) and recurrent neural networks (RNNs), were also adopted in this field [9]. Several IDS were developed based on ANNs, such as RNNs [10] and CNNs [11].

In recent years, a new direction was utilized for the IDS by employing the power of the metaheuristic optimization algorithms adopted in different and complex engineering and optimization problems, including IDS. For example, Alazab et al. [12] employed the moth–flame optimizer algorithm to build an IDS method. The MFO was a feature selection method that enhanced the classifier's performance (Decision Tree). The evaluation showed that the classification accuracy of the DT was improved by applying the MFO. In [13], the authors applied a combined MH method using the firefly algorithm (FA) and ant lion optimization algorithm to build an efficient IDS system. Zhou et al. [14] employed the bat algorithm as a feature selection to build an IDS. It was evaluated with random forest classifier, C4.5, and ForestPA. It is clear that MH optimization algorithms have shown significant performance in IDS applications; thus, they have been widely adopted, such as whale optimization algorithm [15], particle swarm optimization algorithm [16], Aquila optimization algorithm [17], reptile search algorithm [18], salp swarm algorithm [19], and many others.

Paper Contribution

Following the successful applications of MH optimization algorithms in IDS, we propose an efficient feature selection technique called MGO. This method is developed based on two aspects; the first is to utilize the power of the Growth Optimizer (GO) in the exploration phase of the search process. The second aspect is to employ the integration between GO and WOA in the exploitation phase. The main objective of this study can be simplified as the following points:

1. Suggest a different method for securing IoT by combining DL and feature selection techniques.
2. Use a CNN model to analyze network traffic records and identify complex feature representations.
3. Create a modified version of Growth Optimizer (GO) for improved intrusion detection in IoT environments. The modification uses the operators of the Whale Optimization Algorithm (WOA). The proposed method, called MGO, is employed to address the issue of discrete feature selection.
4. Evaluate the performance of the MGO against established methods using four actual intrusion datasets.

The paper is structured as follows: Section 2 explains the employed methods, Section 3 outlines the proposed IoT security system, Section 4 assesses the system, and Section 5 concludes the results.

2. Background

2.1. Growth Optimizer

In this section, the Growth Optimizer (GO) simulates how people train and reflect as they progress in society. In the learning phase, the information is collected from the

environment, whereas the reflection aims to examine the shortcomings and improve the learning method.

In general, the GO starts by using Equation (1) to generate the population X which stands for the solutions for the tested problem.

$$X_i = r \times (U - L) + L,\ i = 1, ..., N \tag{1}$$

where r is the random value and the limits of the search domain of the problem are represented using U and L. N refers to the total number of solutions in X.

Following [20], X is divided into three parts according to the parameters named $P_1 = 5$. The first part comprises the leader and the elites (varying from 2 to P_1). The second part contains the middle level (i.e., from $P_1 + 1$ to $N - P_1$) and the third part contains the bottom level (i.e., $N - P_1 + 1$ and N), whereas the best solution is the leader of the upper level.

2.1.1. Learning Stage

By confronting disparities between people, examining the causes of those differences, and learning from them, individuals can be greatly helped in their progress. The GO's learning stage simulates four key gaps that are formulated:

$$\begin{aligned} G_1 &= X_b - X_{bt} \\ G_2 &= X_b - X_w \\ G_1 &= X_{bt} - X_w \\ G_1 &= X_{r1} - X_{r2} \end{aligned} \tag{2}$$

where X_b, X_{bt}, X_w indicate best, better, and worst solution, respectively; in addition, X_{r1}, and X_{r2} are two random solutions. $G_k (k = 1, 2, 3, 4)$ stands for the gap used to improve the skills learned and decrease the difference between them. Moreover, to reflect the variation between the groups, the parameter named learning factor (LF) is applied and its formulation is given as:

$$LF_k = \frac{||G_k||}{\sum_{k=1}^{4} ||G_k||}, k = 1, 2, 3, 4 \tag{3}$$

Following [20], the individual can assess his learned knowledge using the parameter (SF_i):

$$SF_i = \frac{GR_i}{GR_{max}} \tag{4}$$

where GR_{max} and GR_i represent the maximum growth resistance of X and the growth of X_i, respectively.

According to the information collected from LF_k and SF_i each X_i can receive new knowledge from the solution belonging to each gap G_k using the knowledge acquisition (KA_k) that is defined as:

$$KA_k = SF_i \times LF_k \times G_k,\ k = 1, 2, 3, 4 \tag{5}$$

After that, the solution X_i can improve its information using the following formula:

$$X_i(t+1) = X_i(t) + \sum_{k=1}^{4} KA_k \tag{6}$$

The quality of the updated version of X_i is computed and compared with the previous one to determine whether there is a significant difference between them.

$$X_i(t+1) = \begin{cases} X_i(t+1) & if\ f(X_i(t+1)) \leq f(X_i(t)) \\ \begin{cases} X_i(t+1) & if\ r_1 < P_2, ind(i) = ind(1) \\ X_i(t) & otherwise \end{cases} & otherwise \end{cases}, \tag{7}$$

where r_2 stands for a random number and $P_2 = 0.001$ is the probability retention. $ind(i)$ refers to the ranking of X_i based on the ascending order X using the fitness value.

2.1.2. Reflection Stage

The solution must develop their ability to reflect on the knowledge they have learned, meaning that X must identify all of their areas of weakness, make up for them, and retain their information. They ought to adopt the undesirable attributes of successful X while retaining their outstanding qualities. When the lesson of a specific aspect cannot be mended, the prior information should be abandoned and systematic learning should resume. Equations (8) and (9) can be used to mathematically model this process.

$$X_i(t+1) = \begin{cases} \begin{cases} r_4 \times (U - L) & if\ r_3 < AF \\ X_i(t) + r_5 \times (X_R - X_i(t)) & otherwise \end{cases} & otherwise \\ X_i(t) & otherwise \end{cases}, \quad (8)$$

$$AF = 0.01 + 0.99 \times (1 - \frac{FEs}{max_{FE}}) \quad (9)$$

where r_3, r_4, and r_5 are random values. X_R refers to a solution defined as the top $P_1 + 1$ solutions in X. AF refers to the attenuation factor which depends on function evaluation FE and the total number of functions evaluations max_{FE}.

After the complete reflection stage, X_i should evaluate its growth, similar to the learning phase. Therefore, Equation (7) is also applied to achieve this task.

2.2. Whale Optimization Algorithm

The WOA [21] draws inspiration from the unique hunting strategy used by a particular species of killer whale known as humpback, whose hunting style is bubble-net feeding. WOA's mathematical formulation depends on how it behaves when hunting. Each whale's location can be represented by the solution X_b, which can be updated depending on how the whale behaves when attacking its prey. The whales can attack their prey using two different methods. The humpback whale locates its prey and encircles it using the first strategy, known as encircling prey. WOA presupposes that the target prey is the best option ($X_b(t)$). The other whales attempt to update their locations in the direction of $X_b(t)$ after it has been identified (found), as in Equation (10):

$$Dis_i = |B \odot X_b(t) - X_i(t)|,\ B = 2r \quad (10)$$

$$X_i(t+1) = X_b(t) - A \odot Dis_i,\ A = 2a \odot r - a \quad (11)$$

where Dis_i stands for the distance between $X_i(t)$ and $X_b(t)$. $r \in [0, 1]$ refers to a random value. In addition, a denotes a parameter that decreases from 2 to 0 during the process of updating the solution, formulated as $a = a - t\frac{a}{t_{max}}$ (t_{max} is the total of iterations).

The second strategy is called the bubble-net attack. This phase has two main steps: spiral updating location and shrinking encircling mechanism, and reducing the value of a in Equation (11) for satisfying the process of shrinking encircling. The whale's locations, X_i and X_b, are separated by the following distance, which is calculated by the spiral updating position method [21]:

$$X(t+1) = Dis' \odot e^{bl} \odot cos(2\pi l) + X_b(t) \quad (12)$$

In Equation (12), l stands for a constant value which represents the shape of the logarithmic spiral.

The whales can also swim simultaneously around the X_b utilizing a spiraling path and a contracting circle. The following equation depends on integrating Equations (10)–(11) and Equation (12) [21]; therefore, X can be enhanced as:

$$X(t+1) = \begin{cases} X_b(t) - A \odot Dis & if \quad p \geq 0.5 \\ Dis' \odot e^{bl} \odot cos(2\pi l) + X_b(t) & if \quad p < 0.5 \end{cases}. \tag{13}$$

In Equation (13), $p \in [0, 1]$ refers to a probability value used to identify the strategy of updating. In addition, X_i can be enhanced using a random selecting solution X_r instead of X_b as represented using Equation (14) [21]:

$$X(t+1) = X_r - A \odot Dis \tag{14}$$

$$Dis = |B \odot X_{rand} - X(t)| \tag{15}$$

3. Proposed Method

The steps of the developed IoT security are introduced in this section. The developed technique depends on improving the performance of the Growth Optimizer using the Whale Optimization Algorithm (WOA).

3.1. Prepare IoT Dataset

The developed MGO starts by preparing the IoT dataset by normalizing it. This is performed using the min−max technique that applied to the IoT data DS [22], which is represented as

$$DS = \begin{bmatrix} ds_{11} & ds_{12} & ... & ds_{1d} \\ ds_{21} & ds_{22} & ... & ds_{2d} \\ ... & ... & ... & ... \\ ds_{n1} & ds_{n2} & ... & ds_{nd} \end{bmatrix} \tag{16}$$

where $DS_i = [ds_{i1}, ds_{i2}, ..., ds_{id}]$ denotes the features of traffic i, whereas n and d are sample and feature numbers, respectively. The normalized version of DS based on min−max technique is represented as [22]:

$$NDS_{ij} = \frac{Ds_{ij} - min(DS_j)}{max(DS_j) - min(DS_j)} \tag{17}$$

3.2. CNN for Feature Extraction

Convolutional neural networks (CNNs) are widely used in computer vision as they are robust feature extraction modules, especially when employing pre-trained models alongside transfer learning methods. Meanwhile, CNN is also used in applications where the data are one-dimensional such as in natural language processing. Our study aims to train a DL model that benefits from the big data generated from IoT devices to perform network intrusion detection and reduce processing complexity and inference time. Thus, this section proposes a light CNN model to automatically learn helpful patterns/representations rather than relying on the raw data collected from experimental and real network intrusion detection experiments. In addition, we extract the learned features for further processing (feature selection) to improve the overall framework performance (detection accuracy) and reduce the dimensionality space of the represented feature to accelerate the inference process.

The proposed CNN architecture receives a set of samples, X, where each row is a one-dimensional raw sample representing a network traffic record which includes several network attributes (columns) related to the possible attack class, such as flags related to the IP address, TCP flags, destination, source information, type of service, communication protocols, and protection protocols. The CNN architecture, as shown in Figure 1, is

composed of two convolution blocks (ConvBlock) to learn spatial relations between raw attributes and generate new representations as output (feature maps). Each ConvBlock comprises a convolutional layer with a one-dimensional kernel k, activation function, and pooling operation. Each ConvBlock uses a kernel of size 1×3 and 64 output channels to produce the output feature maps out(t) which is a new transformation of the input raw data at a certain timestamp t where i is the input channel. A non-linear activation function name rectified linear unit (ReLU) after each convolution operation is followed by a max-pooling with size two to output the final feature maps.

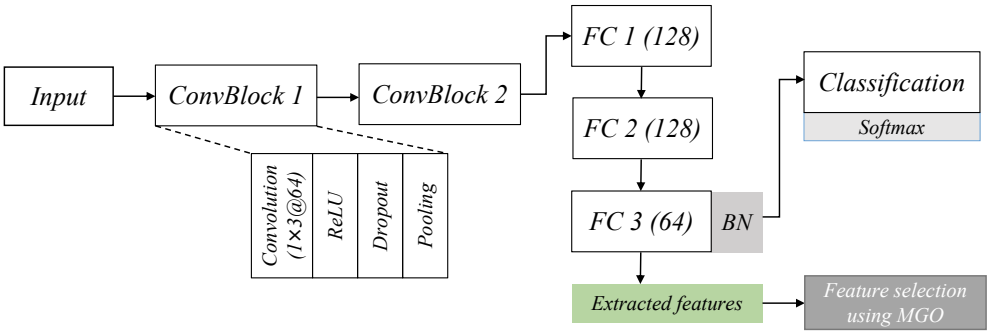


Figure 1. The CNN architecture employed for feature extraction.

3.3. Feature Selection-Based MGO Approach

We developed an alternative FS approach based on a modified version of GO algorithm using WOA as given in Figure 2. This algorithm allocates the relevant features from those extracted using the CNN model.

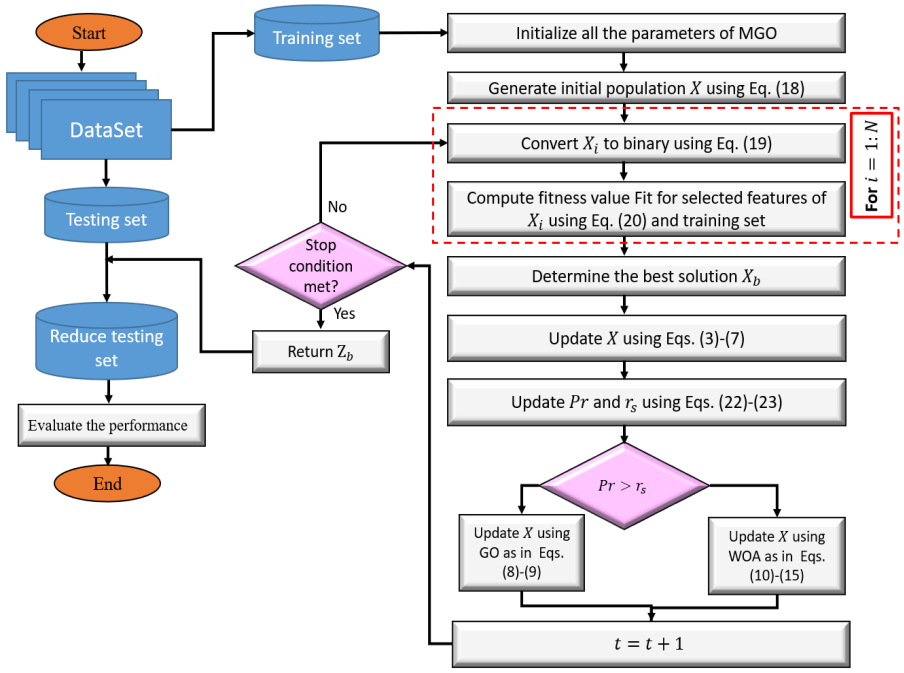


Figure 2. The workflow structure of the MGO feature selection technique.

The first step in MGO as FS approach is to split the data into training and testing sets, which represent 80% and 20%, respectively. Then, the initial solution X is built as given in Equation (18).

$$X_i = LB + rand(1, D) \times (UB - LB),\ i = 1, 2, ..., N \tag{18}$$

where N stands for the total number of solutions and D the number of features. LB and UB are the limits of the search domain. $rand(1, D)$ stands for the random value with D values.

The next step is to generate the Boolean version of X_i using the following formula:

$$BX_{ij} = \begin{cases} 1 & if\ X_{ij} > 0.5 \\ 0 & otherwise \end{cases} \tag{19}$$

We select only the features corresponding to ones in BX_i and remove the other features. Then, we compute the fitness value of X_i as:

$$Fit_i = (1 - \lambda) \times (\frac{|BX_i|}{D}) + \lambda \times \gamma_i \tag{20}$$

where γ_i stands for the classification error based on KNN using the training sets, whereas, $\lambda \in [0, 1]$ is the weight used to control the balancing between γ_i and the $(\frac{|BX_i|}{D})$, which represents the ratio of relevant features.

Thereafter, we determined the best solution X_b and used it to enhance the current solutions by combining GO and WOA. This was conducted using GO operators in the exploration phase, while the following integration schema was used during the exploitation phase.

$$X_i = \begin{cases} X_{WOA} & if\ Pr > r_s \\ X_{GO} & otherwise \end{cases} \tag{21}$$

where X_{GO} refers to using the operators of GO that were used to update X_i and X_{WOA} is the operator of WOA defined in Equations (10)–(15). Pr is the probability of each X_i and it is defined as:

$$Pr = \frac{Fit_i}{\sum_{i=1}^{N} X_i} \tag{22}$$

where Fit_i stands for the fitness value of X_i. In addition, the value of r_s is updated using the following formula.

$$r_s = min(Pr) + rand \times (max(Pr) - min(pr)) \tag{23}$$

where min and max are the minimum and maximum functions, respectively.

Then, the stop condition is checked and in case they are met, the update process is stopped. Otherwise, we repeat it again. After that, X_b is used to remove irrelevant features from the testing set and evaluate this process using different performance criteria.

The time complexity of the developed MGO as FS method depends on some factors such as (1) the size of population N, (2) the dimension of features D, and (3) the number of iterations t_{max}. So, the complexity of MGO is formulated as:

$$O(MGO) = N + t \times (N \times D + (O(WOA) + O(GO_{Reflection})) \times D) \tag{24}$$

$$O(MGO) = N \times D + t \times (N \times D + (K_w \times D + (N - K_w) \times D)) \tag{25}$$

So,

$$O(MGO) = N \times D + t \times (N \times D) = O(N \times D \times t) \tag{26}$$

where K_w refers to the number of solutions that will be updated using WOA.

4. Experimental Series and Results

The section uses a set of experimental series to assess the developed IoT security based on a modified version of the GO algorithm using WOA. These experimental series are implemented using a set of real-world IoT datasets.

4.1. Evaluation Measures

The effectiveness of the suggested technique and all comparing methodologies is evaluated using several indicators.

- Average accuracy (AV_{Acc}: This measure stands for the rate of a correct intrusion detected using the algorithm and it is represented as:

$$AV_{Acc} = \frac{1}{N_r} \sum_{k=1}^{N_r} Acc_{Best}^{k}, \tag{27}$$

$$Acc_{Best} = \frac{TP + TN}{TP + FN + FP + TN}$$

in which $N_r = 30$ indicates the iteration numbers.

- Average Recall (AV_{recall}): is the percentage of intrusion predicted positively (it can be called true positive rate (TPR)). It can be computed as:

$$AV_{Recall} = \frac{1}{N_r} \sum_{k=1}^{N_r} Recall_{Best}^{k}, \; Recall_{Best} = \frac{TP}{TP + FN} \tag{28}$$

- Average Precision (AV_{Prec}): stands for the rate of TP samples of all positive cases with the formulation:

$$AV_{Prec} = \frac{1}{N_r} \sum_{k=1}^{N_r} Prec_{Best}^{k}, Prec_{Best} = \frac{TP}{FP + TP} \tag{29}$$

- Average F1-measure (AV_{F1}): can be computed as:

$$AV_{F1} = 2\frac{Recall \times Precision}{Precision + Recall} \tag{30}$$

- Average G-mean (AV_{GM}): can be computed as:

$$AV_{GM} = \sqrt{Recall \times Precision} \tag{31}$$

4.2. Experiments Setup

In our experiments, we trained the CNN model on each dataset record for 100 epochs with early stopping and Adam with a 0.005 learning rate to update the network parameters. A batch of size 2024 is used to iterate over the data samples. In addition, batch normalization and dropout with a 0.38 ratio were used as regularization techniques to prevent overfitting, increase generalization, and accelerate the training. The network hyper-parameters were selected based on several experiments with different setups where the best hyper-parameters were used in our experiments that maximize the detection accuracy. The CNN was developed using Pytorch framework (https://pytorch.org/, accessed on 15 January 2023) and the training was performed using Nvidia GTX 1080.

To test the performance of the developed MGO, we compared it to several optimizers, namely, the traditional WOA [21], the traditional GO, grey wolf optimizer (GWO) [23], Transient Search Optimization (TSO) [24], firefly algorithm (FFA) [25], and moth flame optimization (MFO) [26]. We set the parameters of each algorithm based on its original implementation, whereas the iterations number is set to 50, and the agent number is 20.

4.3. Experimental Datasets

To validate the proposed framework, we used four well-known datasets for network intrusion detection, which are publicly available. The datasets used to train and test the proposed framework are KDDCup-99, NSL-KDD, BoT-IoT, and CICIDS-2017. Figure 3 shows the corresponding statistics of the datasets used in our experiments. The KDD

(Knowledge Discovery and Data Mining) Cup 1999 dataset (KDDCup-99) was created in 1999 for the KDD cup competition organized by the Defence Advanced Research Project Agency (DARPA). The KDDCup-99 collects TCP/IP dump files containing network traffic records recorded for two months. The total number of records is around five million, with 41 features. We used the 10% version of the KDDCup-99 dataset, which contains less than one million records with four attack types, including user-to-root (U2R), probing, remote-to-user (R2L), and denial-of-service (DoS) besides normal traffic. The NSL-KDD dataset is a distilled version of the KDDCup-99 dataset with 41 features. The CICIDS-2017 [27] dataset provides more realistic network traffic records with 79 features collected by the CICFlowMeter tool focusing on SSH, HTTP, HTTPS, email, and FTP protocols. The dataset is presented in several CSV files, where we used four files to train and test the framework. The total used network records are around 600 thousand, with seven attack types and normal traffic. For the Bot-IoT dataset [28], we used the 5% version with 3.5 million records collected from various IoT devices.

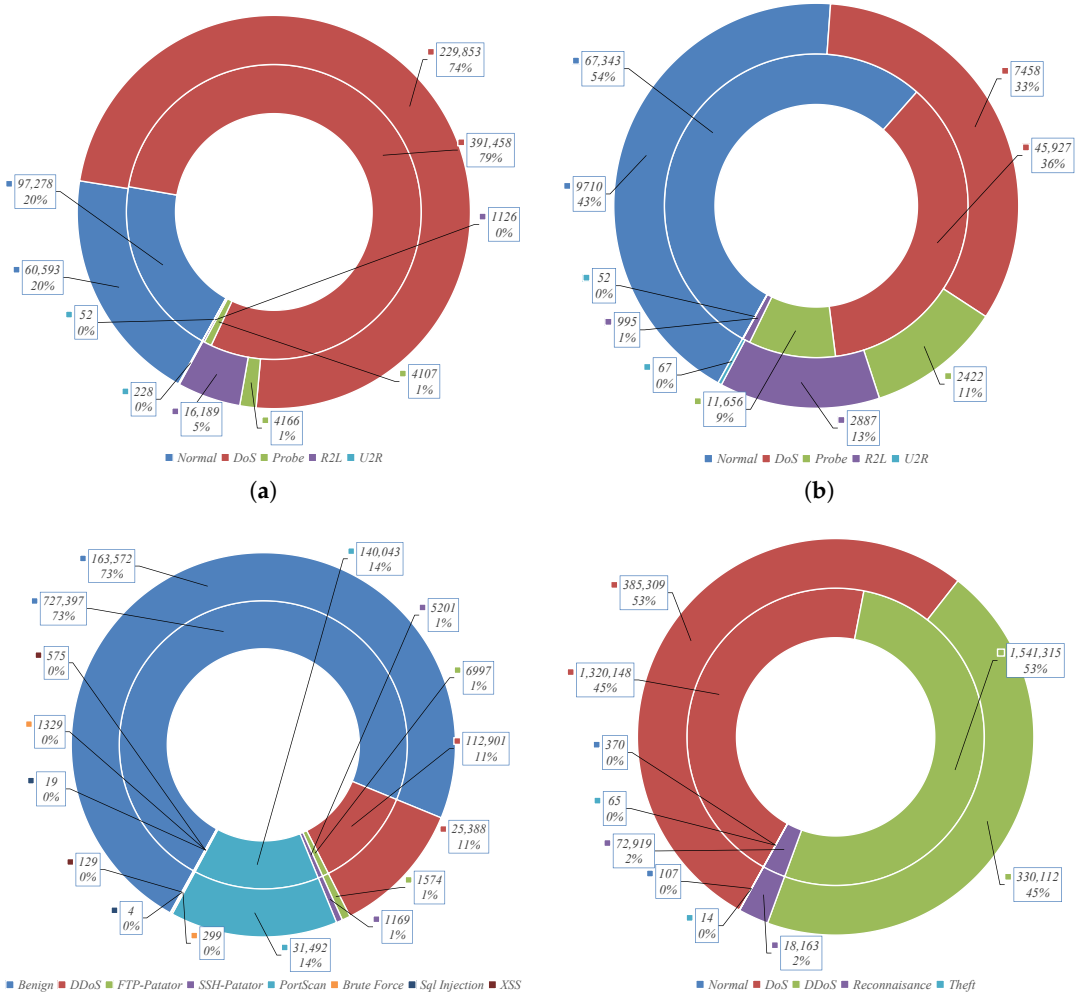


Figure 3. Datasets statistics where the Test set is the outer circle and the Train set is the inner circle. (**a**) KDDCup-99. (**b**) NSL-KDD. (**c**) CICIDS-2017. (**d**) Bot-IoT.

4.4. Result and Discussions

This section discusses the comparison results between the developed MGO and other methods to improve the quality of IoT security. Table 1 shows the average over 25 independent runs for each algorithm using the performance measures.

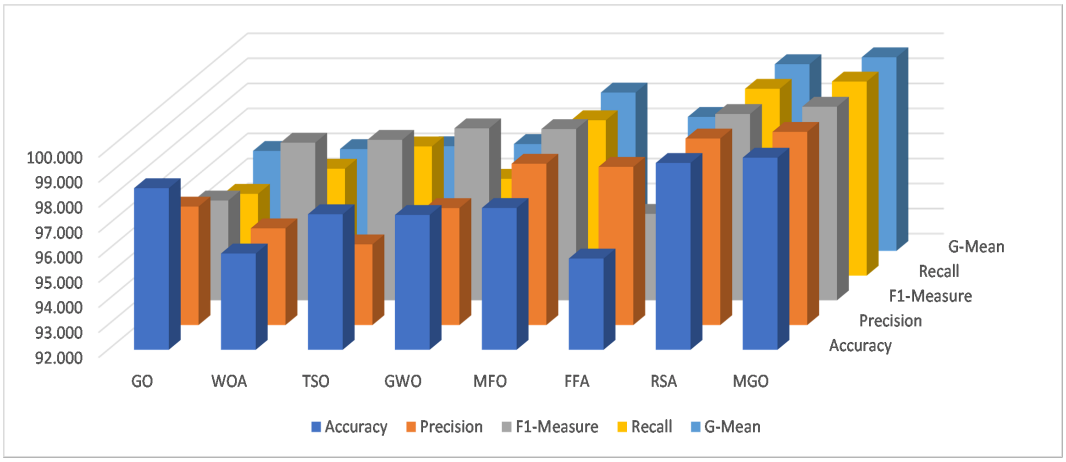
In the multi-classification analysis, the MGO algorithm demonstrates superior efficiency compared to other algorithms during the learning phase on the datasets (i.e., KDD99, NSL-KDD, BIoT, and CIC2017). However, it falls behind the RSA regarding performance on the BIoT dataset. Furthermore, MGO excels in detecting attack types using testing samples on all four datasets compared to other methods.

In addition, the accuracy value of RSA is better than other methods, followed by MFO which allocates the third rank overall to the other methods during the training stage, whereas the Accuracy of RSA based on the testing set is the best after the developed MGO algorithm. Based on Precision, F1-Measure, and Recall, the FFA, GWO, and MFO is the best algorithm that allocates the second rank among the tested algorithms within the testing phase.

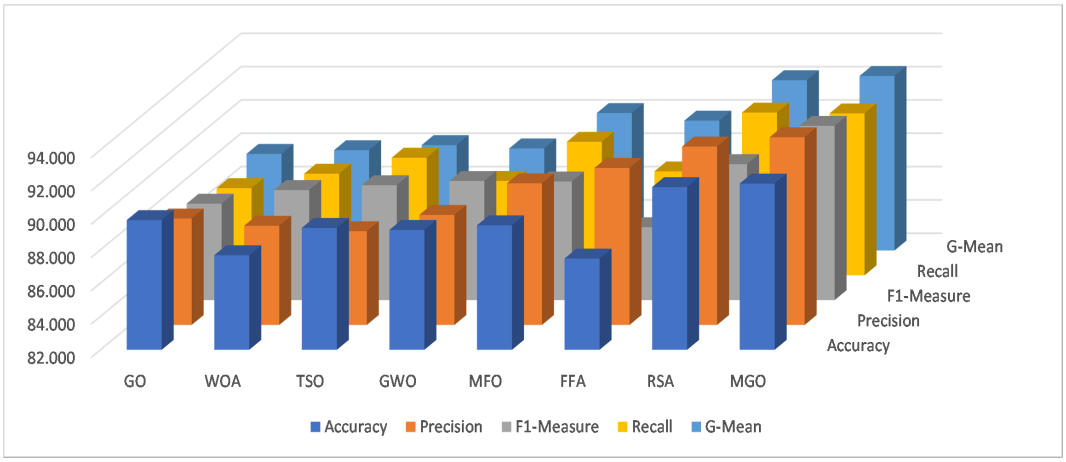
Table 1. The performance of MGO using the IoT security datasets. (Note: bold indicate best results).

		Train					**Test**				
		Accuracy	**Precision**	**F1**	**Recall**	**GM**	**Accuracy**	**Precision**	**F1**	**Recall**	**GM**
KDD99	GO	98.515	93.483	92.652	91.835	92.655	90.615	84.249	83.797	83.350	83.798
	WOA	92.275	92.414	97.304	93.126	92.769	84.375	82.501	87.351	85.225	83.852
	TSO	95.439	91.027	97.437	94.919	92.953	87.536	80.791	87.479	87.016	83.846
	GWO	95.513	94.062	98.482	92.383	93.219	87.618	84.131	88.533	84.488	84.310
	MFO	96.073	97.631	98.371	97.123	97.377	88.175	87.763	88.420	89.225	88.491
	FFA	91.988	97.328	91.538	93.368	95.328	84.318	91.609	84.285	85.698	88.604
	RSA	**99.910**	99.909	99.906	99.910	99.910	**92.040**	89.684	**89.985**	**92.040**	90.855
	MGO	**99.910**	**99.959**	**99.946**	**99.933**	**99.946**	**92.040**	**90.841**	90.941	91.040	**90.941**
NSL-KDD	GO	97.108	95.104	93.017	91.020	93.040	70.224	72.200	69.794	67.544	69.833
	WOA	91.947	92.080	96.968	92.797	92.438	67.951	71.131	68.907	68.801	69.956
	TSO	95.078	90.657	97.067	94.558	92.587	71.330	71.298	69.697	70.810	71.053
	GWO	95.182	93.724	98.143	92.052	92.884	71.066	72.151	69.948	67.936	70.012
	MFO	95.745	97.297	98.035	96.795	97.046	71.626	76.122	69.844	72.676	74.379
	FFA	91.660	96.991	91.201	93.040	94.995	67.437	75.873	62.944	68.817	72.259
	RSA	99.201	99.158	99.148	99.201	99.180	76.107	82.171	71.731	76.107	79.081
	MGO	**99.214**	**99.458**	**99.437**	**99.416**	**99.437**	**76.725**	**83.105**	**79.759**	**76.672**	**79.824**
BIoT	GO	99.068	99.107	99.076	99.045	99.076	99.141	98.100	98.371	98.644	98.372
	WOA	99.472	99.472	99.472	99.472	99.472	98.956	98.957	99.005	98.964	98.960
	TSO	99.460	99.459	99.459	99.460	99.460	98.986	98.941	99.005	98.981	98.961
	GWO	99.477	99.476	99.476	99.477	99.477	98.990	98.975	99.019	98.959	98.967
	MFO	99.480	99.480	99.480	99.480	99.480	98.998	99.013	99.020	99.009	99.011
	FFA	99.479	99.478	99.478	99.479	99.478	98.954	99.007	98.949	98.968	98.987
	RSA	98.829	98.829	98.829	98.829	98.829	99.020	99.098	99.070	99.038	99.068
	MGO	**99.629**	**99.529**	**99.629**	**99.729**	**99.629**	**99.220**	**99.188**	**99.218**	**99.248**	**99.218**
CIC2017	GO	99.130	99.239	99.204	99.170	99.204	99.170	99.020	99.215	99.410	99.215
	WOA	99.690	99.490	99.450	99.690	99.590	99.430	99.240	99.190	99.430	99.335
	TSO	99.680	99.750	99.680	99.710	99.730	99.420	99.480	99.420	99.450	99.465
	GWO	99.370	99.430	99.380	99.560	99.495	99.110	99.180	99.120	99.300	99.240
	MFO	99.360	99.370	99.480	99.430	99.400	99.100	99.120	99.220	99.170	99.145
	FFA	99.450	99.480	99.600	99.740	99.610	99.200	99.220	99.350	99.490	99.355
	RSA	99.911	99.910	99.889	99.911	99.910	99.911	99.907	99.888	99.911	99.909
	MGO	**99.941**	**99.920**	**99.926**	**99.931**	**99.926**	**99.941**	**99.947**	**99.942**	**99.936**	**99.942**

Figure 4 illustrates the average performance of each method across various datasets for various measures. The MGO method has the highest average performance for training and testing in multi-classification, followed by the Accuracy method which has better accuracy. The RSA method has better Recall in both training and testing, and GWO has a better F1-Measure in both training and testing. MFO and FFA have higher Precision in training and testing sets, respectively.



(**a**)



(**b**)

Figure 4. The average overall of the tested sets for multi-classification. (**a**) Train Set. (**b**) Testing set.

To further analyze the results, we applied the Friedman test to determine if there was a significant difference between the different methods. The Friedman test gave us the mean rank for each method, as seen in Table 2. From these mean ranks, we can see that MGO has the highest mean rank across all training and testing set performance measures, followed by RSA overall performance measures in case of using the training set. Meanwhile, the mean rank of the accuracy, precision, F1-measure, recall, and GM for the GO, FFA, MFO, TSO, and FFA, respectively, is the second rank after the developed MGO in the case of the testing set.

Table 2. Results of Friedman test.

		GO	WOA	TSO	GWO	MFO	FFA	RSA	MGO
Training	Accuracy	3.7500	3.5000	3.5000	4.0000	4.7500	3.0000	5.6250	7.8750
	Precision	2.5000	3.2500	2.7500	3.7500	5.2500	5.0000	5.5000	8.0000
	F1-Measure	1.7500	3.2500	4.2500	4.7500	5.2500	3.2500	5.5000	8.0000
	Recall	1.2500	3.5000	4.5000	3.0000	5.2500	5.0000	5.5000	8.0000
	GM	2.0000	2.7500	3.5000	3.7500	5.2500	5.2500	5.5000	8.0000
Testing	Accuracy	4.7500	3.0000	4.0000	3.5000	4.2500	1.7500	6.8750	7.8750
	Precision	2.5000	2.7500	2.7500	3.2500	4.7500	5.5000	6.7500	7.7500
	F1-Measure	2.2500	2.6250	4.1250	4.5000	5.0000	2.5000	7.0000	8.0000
	Recall	1.5000	3.2500	5.0000	2.0000	4.7500	4.5000	7.2500	7.7500
	GM	1.2500	2.7500	3.7500	3.5000	4.5000	5.2500	7.0000	8.0000

4.5. Comparison with Existing Methods

This section compares the results of the developed MGO with other techniques as given in Table 3. Most of those methods may use either one or two datasets. From these results, we can observe that in the case of KDD99, the accuracy of MGO is better than the method applied in [29]; however, the method presented in [30] has better performance than MGO. In the case of the BIoT, we can observe that the developed MGO performs better than other methods mentioned in this study, followed by the method introduced in Churcher et al. [31] (KNN) which is superior to other methods. For the CICIDS2017 dataset, we noticed that MGO provided better results than the competitive methods.

Table 3. Comparison with other methods.

Dataset	Work	Accuracy
KDD Cup 99	Wu [29]	85.24
	Farahnakian et al. [30]	96.53
	MGO	0.9204
NSL-KDD	Ma et al. [32] SCDNN	72.64
	Javaid et al. [33] STL	74.38
	Tang et al. [34] DNN	75.75
	Imamverdiyev et al. [35] Gaussian–Bernoulli RBM	73.23
	MGO	76.725
BIoT	[36] (BiLSTM)	98.91
	Alkadi et al. [36] (NB)	97.5
	Alkadi et al. [36] (SVM)	97.8
	Churcher et al. [31] (KNN)	99
	Churcher et al.[31] (SVM)	79
	Churcher et al. [31] (DT)	96
	Churcher et al. [31] (NB)	94
	Churcher et al. [31] (RF)	95
	Churcher et al. [31] (ANN)	97
	Churcher et al. [31] (LR)	74
	MGO	99.22
CICIDS2017	Vinayakumar [37]	94.61
	Laghrissi et al. [38]	85.64
	Alkahtani et al. [39]	80.91
	MGO	99.941

From the previous results, it is clear that the developed method has a high potential to improve the prediction of attacks in IoT environments. However, the method has some limitations, such as being time-consuming due to the model learning process. These limitations can be addressed by using transfer learning techniques. In addition, the MGO still requires handling the imbalanced datasets in IoT and this can be handled by using the mechanism mentioned in [40].

5. Conclusions and Future work

Our study investigated the development of a two-phase framework to improve the detection accuracy over existing intrusion detection systems (IDS). In addition, the developed framework integrates a deep learning (DL) model and swarm intelligence (SI) technique to combine both techniques' advantages and facilitate the deployment of the framework in the Internet of Things (IoT) system. We implemented a convolutional neural network architecture as a core feature extraction module to learn and extract new feature representation from the raw input data (network traffic records). In addition, we proposed a novel feature selection (FS) approach based on a modified variant of the Growth Optimizer (GO) algorithm to reduce the extracted feature representation space, speed up the inference, and improve the overall framework performance on IDS. The proposed FS method relies on applying the GWO to boost the search process of the traditional GO algorithm. Thus, the

results show that the suggested method performed best compared to several optimization techniques using different evaluation indicators with several public IDS datasets. In future work, the developed MGO can be extended and experimented with in different applications such as healthcare, human activity recognition, fake news detection, and others.

Author Contributions: Conceptualization: A.F., A.D. and M.A.A.A.-q.; Data curation: A.F., M.A.A.A.-q., S.A.A. and S.S.A.; Formal analysis: M.A.A.A.-q., M.A.E., S.A.A. and S.S.A.; Funding acquisition: A.F., Investigation: A.F.; Methodology: A.F. and A.D.; Resources: A.F.; Project administration: A.F. and S.L.; Software: M.A.A.A.-q.; Supervision: A.F. and S.L.; Validation: S.L., S.A.A. and S.S.A.; Visualization: A.D. and M.A.A.A.-q.; Writing—original draft preparation: A.F., A.D. and M.A.A.A.-q.; Writing—review and editing: M.A.A.A.-q., S.L., S.A.A. and S.S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Key R&D Program of China under Grand No. 2021YFB2012202 and the Hubei Provincial Science and Technology Major Project of China under Grant No. 2020AEA011 and the Key Research & Development Plan of Hubei Province of China under Grant No. 2021BAA171, 2021BAA038, 2020BAB100 and the project of Science, Technology and Innovation Commission of Shenzhen Municipality of China under Grant No. JCYJ20210324120002006 and JSGG20210802153009028, the 2021 Industrial Technology Basic Public Service Platform Project of the Ministry of Industry and Information Technology of PRC under Grand No. 2021-0171-1-1.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All the datasets are public, as we described in the main text.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Debicha, I.; Bauwens, R.; Debatty, T.; Dricot, J.M.; Kenaza, T.; Mees, W. TAD: Transfer learning-based multi-adversarial detection of evasion attacks against network intrusion detection systems. *Future Gener. Comput. Syst.* **2023**, *138*, 185–197. [CrossRef]
2. Lata, S.; Singh, D. Intrusion detection system in cloud environment: Literature survey & future research directions. *Int. J. Inf. Manag. Data Insights* **2022**, *2*, 100134.
3. Schueller, Q.; Basu, K.; Younas, M.; Patel, M.; Ball, F. A hierarchical intrusion detection system using support vector machine for SDN network in cloud data center. In Proceedings of the 2018 28th International Telecommunication Networks and Applications Conference (ITNAC), Sydney, Australia, 21–23 November 2018; pp. 1–6.
4. Wei, J.; Long, C.; Li, J.; Zhao, J. An intrusion detection algorithm based on bag representation with ensemble support vector machine in cloud computing. *Concurr. Comput. Pract. Exp.* **2020**, *32*, e5922. [CrossRef]
5. Peng, K.; Leung, V.; Zheng, L.; Wang, S.; Huang, C.; Lin, T. Intrusion detection system based on decision tree over big data in fog environment. *Wirel. Commun. Mob. Comput.* **2018**, *2018*. [CrossRef]
6. Modi, C.; Patel, D.; Borisanya, B.; Patel, A.; Rajarajan, M. A novel framework for intrusion detection in cloud. In Proceedings of the Fifth International Conference on Security of Information and Networks, Jaipur, India, 25–27 October 2012; pp. 67–74.
7. Kumar, G.R.; Mangathayaru, N.; Narasimha, G. An improved k-Means Clustering algorithm for Intrusion Detection using Gaussian function. In Proceedings of the International Conference on Engineering & MIS 2015, Istanbul, Turkey, 24–26 September 2015; pp. 1–7.
8. Zhao, X.; Zhang, W. An anomaly intrusion detection method based on improved k-means of cloud computing. In Proceedings of the 2016 Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC), Harbin, China, 21–23 July 2016; pp. 284–288.
9. Hodo, E.; Bellekens, X.; Hamilton, A.; Dubouilh, P.L.; Iorkyase, E.; Tachtatzis, C.; Atkinson, R. Threat analysis of IoT networks using artificial neural network intrusion detection system. In Proceedings of the 2016 International Symposium on Networks, Computers and Communications (ISNCC), Yasmine Hammamet, Tunisia, 11–13 May 2016; pp. 1–6.
10. Almiani, M.; AbuGhazleh, A.; Al-Rahayfeh, A.; Atiewi, S.; Razaque, A. Deep recurrent neural network for IoT intrusion detection system. *Simul. Model. Pract. Theory* **2020**, *101*, 102031. [CrossRef]
11. Wu, K.; Chen, Z.; Li, W. A novel intrusion detection model for a massive network using convolutional neural networks. *IEEE Access* **2018**, *6*, 50850–50859. [CrossRef]
12. Alazab, M.; Khurma, R.A.; Awajan, A.; Camacho, D. A new intrusion detection system based on moth–flame optimizer algorithm. *Expert Syst. Appl.* **2022**, *210*, 118439. [CrossRef]
13. Samadi Bonab, M.; Ghaffari, A.; Soleimanian Gharehchopogh, F.; Alemi, P. A wrapper-based feature selection for improving performance of intrusion detection systems. *Int. J. Commun. Syst.* **2020**, *33*, e4434. [CrossRef]

14. Zhou, Y.; Cheng, G.; Jiang, S.; Dai, M. Building an efficient intrusion detection system based on feature selection and ensemble classifier. *Comput. Netw.* **2020**, *174*, 107247. [CrossRef]
15. Mojtahedi, A.; Sorouri, F.; Souha, A.N.; Molazadeh, A.; Mehr, S.S. Feature Selection-based Intrusion Detection System Using Genetic Whale Optimization Algorithm and Sample-based Classification. *arXiv* **2022**, arXiv:2201.00584.
16. Talita, A.; Nataza, O.; Rustam, Z. Naïve bayes classifier and particle swarm optimization feature selection method for classifying intrusion detection system dataset. *J. Phys. Conf. Ser.* **2021**, *1752*, 012021. [CrossRef]
17. Fatani, A.; Dahou, A.; Al-Qaness, M.A.; Lu, S.; Elaziz, M.A. Advanced feature extraction and selection approach using deep learning and Aquila optimizer for IoT intrusion detection system. *Sensors* **2021**, *22*, 140. [CrossRef]
18. Dahou, A.; Abd Elaziz, M.; Chelloug, S.A.; Awadallah, M.A.; Al-Betar, M.A.; Al-qaness, M.A.; Forestiero, A. Intrusion Detection System for IoT Based on Deep Learning and Modified Reptile Search Algorithm. *Comput. Intell. Neurosci.* **2022**, *2022*, 6473507. [CrossRef]
19. Karuppusamy, L.; Ravi, J.; Dabbu, M.; Lakshmanan, S. Chronological salp swarm algorithm based deep belief network for intrusion detection in cloud using fuzzy entropy. *Int. J. Numer. Model. Electron. Netw. Devices Fields* **2022**, *35*, e2948. [CrossRef]
20. Zhang, Q.; Gao, H.; Zhan, Z. H.; Li, J.; and Zhang, H. Growth Optimizer: A powerful metaheuristic algorithm for solving continuous and discrete global optimization problems. *Knowl.-Based Syst.* **2023**, *261*, 110206. [CrossRef]
21. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
22. Abualigah, L.; Abd Elaziz, M.; Sumari, P.; Geem, Z.W.; Gandomi, A.H. Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer. *Expert Syst. Appl.* **2022**, *191*, 116158. [CrossRef]
23. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
24. Fatani, A.; Abd Elaziz, M.; Dahou, A.; Al-Qaness, M.A.; Lu, S. IoT Intrusion Detection System Using Deep Learning and Enhanced Transient Search Optimization. *IEEE Access* **2021**, *9*, 123448–123464. [CrossRef]
25. Yang, X.S.; He, X. Firefly algorithm: recent advances and applications. *Int. J. Swarm Intell.* **2013**, *1*, 36–50. [CrossRef]
26. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl.-Based Syst.* **2015**, *89*, 228–249. [CrossRef]
27. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* **2018**, *1*, 108–116.
28. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Gener. Comput. Syst.* **2019**, *100*, 779–796. [CrossRef]
29. Wu, P. Deep learning for network intrusion detection: Attack recognition with computational intelligence. Ph.D. Thesis, UNSW Sydney, Kensington, Australia, 2020.
30. Farahnakian, F.; Heikkonen, J. A deep auto-encoder based approach for intrusion detection system. In Proceedings of the 2018 20th International Conference on Advanced Communication Technology (ICACT), Chuncheon, Republic of Korea, 11–14 February 2018; pp. 178–183.
31. Churcher, A.; Ullah, R.; Ahmad, J.; Ur Rehman, S.; Masood, F.; Gogate, M.; Alqahtani, F.; Nour, B.; Buchanan, W.J. An experimental analysis of attack classification using machine learning in IoT networks. *Sensors* **2021**, *21*, 446. [CrossRef]
32. Ma, T.; Wang, F.; Cheng, J.; Yu, Y.; Chen, X. A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks. *Sensors* **2016**, *16*, 1701. [CrossRef]
33. Javaid, A.; Niyaz, Q.; Sun, W.; Alam, M. A deep learning approach for network intrusion detection system. In Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies (Formerly BIONETICS), New York, NY, USA, 3–5 December 2015; pp. 21–26.
34. Tang, T.A.; Mhamdi, L.; McLernon, D.; Zaidi, S.A.R.; Ghogho, M. Deep learning approach for network intrusion detection in software defined networking. In Proceedings of the 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), Fez, Morocco, 26–29 October 2016; pp. 258–263.
35. Imamverdiyev, Y.; Abdullayeva, F. Deep learning method for denial of service attack detection based on restricted boltzmann machine. *Big Data* **2018**, *6*, 159–169. [CrossRef]
36. Alkadi, O.; Moustafa, N.; Turnbull, B.; Choo, K.K.R. A deep blockchain framework-enabled collaborative intrusion detection for protecting IoT and cloud networks. *IEEE Internet Things J.* **2020**, *8*, 9463–9472. [CrossRef]
37. Vinayakumar, R.; Alazab, M.; Soman, K.; Poornachandran, P.; Al-Nemrat, A.; Venkatraman, S. Deep learning approach for intelligent intrusion detection system. *IEEE Access* **2019**, *7*, 41525–41550. [CrossRef]
38. Laghrissi, F.; Douzi, S.; Douzi, K.; Hssina, B. Intrusion detection systems using long short-term memory (LSTM). *J. Big Data* **2021**, *8*, 65. [CrossRef]
39. Alkahtani, H.; Aldhyani, T.H. Intrusion detection system to advance internet of things infrastructure-based deep learning algorithms. *Complexity* **2021**, *2021*, 5579851. [CrossRef]
40. Luque, A.; Carrasco, A.; Martín, A.; de Las Heras, A. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognit.* **2019**, *91*, 216–231. [CrossRef]