sustainability

MDPI

Article

Design of Metaheuristic Optimization Algorithms for Deep Learning Model for Secure IoT Environment

Amit Sagu [1], Nasib Singh Gill [1], Preeti Gulia [1], Pradeep Kumar Singh [2] and Wei-Chiang Hong [3,4,*]

[1] Department of Computer Science & Applications, Maharshi Dayanand University, Rohtak 124001, India
[2] School of Technology Management & Engineering, Narsee Monjee Institute of Management Studies (NMIMS), Chandigarh 160014, India
[3] Department of Information Management, Asia Eastern University of Science and Technology, New Taipei 22046, Taiwan
[4] Department of Information Management, Yuan Ze University, Chungli 320315, Taiwan
* Correspondence: samuelsonhong@gmail.com

Abstract: Because of the rise in the number of cyberattacks, the devices that make up the Internet of Things (IoT) environment are experiencing increased levels of security risks. In recent years, a significant number of centralized systems have been developed to identify intrusions into the IoT environment. However, due to diverse requirements of IoT devices such as dispersion, scalability, resource restrictions, and decreased latency, these strategies were unable to achieve notable outcomes. The present paper introduces two novel metaheuristic optimization algorithms for optimizing the weights of deep learning (DL) models, use of DL may help in the detection and prevention of cyberattacks of this nature. Furthermore, two hybrid DL classifiers, i.e., convolutional neural network (CNN) + deep belief network (DBN) and bidirectional long short-term memory (Bi-LSTM) + gated recurrent network (GRU), were designed and tuned using the already proposed optimization algorithms, which results in ads to improved model accuracy. The results are evaluated against the recent approaches in the relevant field along with the hybrid DL classifier. Model performance metrics such as accuracy, rand index, f-measure, and MCC are used to draw conclusions about the model's validity by employing two distinct datasets. Regarding all performance metrics, the proposed approach outperforms both conventional and cutting-edge methods.

Keywords: deep learning models; IoT security; optimization algorithms

Citation: Sagu, A.; Gill, N.S.; Gulia, P.; Singh, P.K.; Hong, W.-C. Design of Metaheuristic Optimization Algorithms for Deep Learning Model for Secure IoT Environment. *Sustainability* **2023**, *15*, 2204. https://doi.org/10.3390/su15032204

Academic Editor: Fabrizio D'Ascenzo

Received: 24 December 2022
Revised: 17 January 2023
Accepted: 20 January 2023
Published: 25 January 2023

1. Introduction

Internet of Things (IoT) refers to any device that can automatically gather and transmit data over a network without human involvement. It is steadily growing in importance and is now pervasive throughout our daily lives [1]. Furthermore, it is a revolution that brings together a wide range of diverse elements such as smart systems, automatic devices, actuators, and sensors. However, growing worries about security, such as software bugs and cyberattacks, can stop many people from using IoT devices. Such IoT security issues are especially significant for businesses operating in industries such as healthcare, financial services, industrial production, transportation, commerce, and other fields that have already begun adopting IoT systems. Another reason to prioritize security when developing IoT systems is to protect their data, smart devices collect massive amounts of sensitive data, including personal information, which must be safeguarded against intrusion. In addition, having a password that is too easy to guess is another problem that the user needs to be mindful of. Figure 1 displays the top ten passwords that are both easily guessed and come as the default option for IoT devices. Deep Learning (DL) can safeguard the IoT environment by automating the process of scanning and managing IoT devices across the entire network. A DL approach can perform scans on every device connected to the network, automatically preventing and terminating any attacks it finds. It enables IoT security to make predictions

based on historical behavior. When it detects a known vulnerability or attack, such as a distributed denial of service (DDoS) attack (Figure 2 is showing how rapidly it is growing over time), it analyzes the current behavior of the network and compares it to the behavior patterns found in instances of attacks. Learning from traffic patterns and giving a heuristic solution to IoT large-scale attacks are two examples of how DL can be used to deliver considerable value, as well as to stimulate the construction and training of models [2,3]. The aging process and the persistent analysis of data assist DL algorithms to produce smarter decisions or increased forecast accuracy [4].

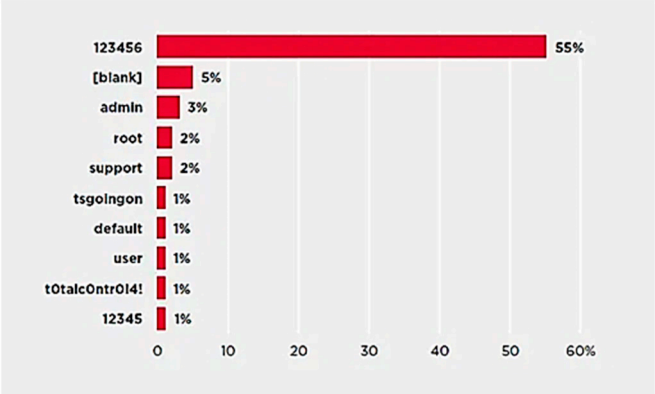

Figure 1. Top common passwords used in IoT devices [5].

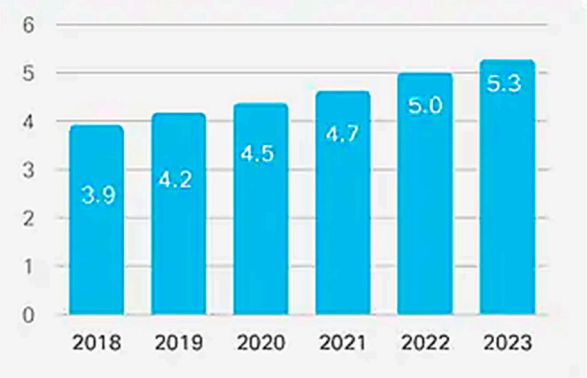

Figure 2. Increases in DDoS attacks over the years [6].

A wide range of built-in and customized DL models were used in several research works to determine the optimum solution for large-scale attacks generated by IoT devices. The following is a list of models identified through data analysis that are frequently used by researchers and produce strong results in identifying malicious network traffic.

Convolution neural network (CNN)—CNNs are a subset of neural networks that are designed to categorize input data, most commonly images, into a variety of distinct categories. Networks are composed of one or more convolutional layers [7]. The convolutional layer, pooling layer, and fully connected layer make up the foundational architecture of CNN. These layers are placed one on top of the other. CNNs are renowned for their ability to produce higher levels of accuracy while completing difficult problems. The IoT devices are resource-constrained, but CNN necessarily requires a large compute power to deliver high accuracy [8]. It is essential to develop a balanced CNN model that works effectively and gives the best accuracy while using the least amount of processing power. This necessitates reducing the size of the CNN model such that it can be implemented on an IoT device, which takes both prior knowledge and extensive experimentation. In [9], the author built twelve different CNN architectures, each with a unique set of features, with the goal of automatically detecting the directionality of textures, whereas in [10], the authors devel-

oped a universal pipeline with the goal of boosting the accuracy of CNN-based approaches through the use of computer graphics, and they validated it using data from many different disciplines and datasets.

Deep belief network (DBN)—The DBNs were developed as a means of resolving issues that arise during the training of traditional neural networks in deep layered networks. These issues include slow learning, becoming stuck in local minima as a result of poor parameter selection, and the requirement of a large number of training datasets. DBN is made up of multiple layers of neural networks, which are also referred to as "Boltzmann Machines" [11]. Hidden neurons, visible neurons, and layers that create an output layer are all components of the DBN framework, which is an intellectual model. The DBN is sufficiently trained to be able to recognize the presence of potential threats within the network based on the features that are retrieved.

Recurrent neural network (RNN)—RNNs are a specific kind of artificial neural network that processes data in sequential order. For ordinal or temporal issues, such as translation services, natural language processing (NLP), voice recognition, and image captioning, these deep learning methods are frequently utilized. RNNs, in the same manner as feedforward neural networks, make use of training data to learn. They are differentiated by their "memory", which allows them to use information gathered from previous inputs to modify the input and output of the system at the current time. In contrast to typical deep neural networks, which operate under the assumption that inputs and outputs are not reliant on one another, recurrent neural networks include outputs that are determined by the elements that came before them in the sequence [12]. Because of this, RNN is a great option for evaluating the traffic on IoT networks for anomaly identification [13]. Figure 3 depicts the general architecture of an RNN.

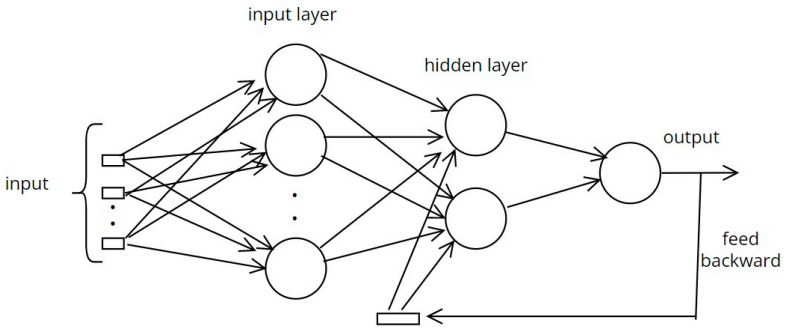

Figure 3. Recurrent Neural Network Architecture [14].

Long short-term memory (LSTM)—This well-known RNN design was initially proposed by Sepp Hochreiter and Juergen Schmidhuber as a solution to the issue of vanishing gradients. The name "gate" refers to each of these three components of an LSTM cell. The initial component is known as the Forget gate, the second component is called the Input gate, and the third and final component is called the Output gate. The Forget gate is responsible for deciding what information should be discarded and what should be preserved. LSTMs were developed with the express goal of avoiding the long-term dependency issue [15].

Gated recurrent unit (GRU)—There is a type of RNN known as the GRU, and in some circumstances, it is superior to long-term and short-term memory. The GRU is more efficient and requires less memory than LSTM. Additionally, GRUs solve the problem of vanishing gradients, which is a problem that traditional recurrent neural networks face while trying to update the weights of the network. The update gate and the reset gate are the two gates that are utilized by GRU to overcome challenges. These gates determine what information is permitted to come through to the output and can be educated to retain information from further back in the process [16].

During the process of training the DL model, we need to make adjustments to the weights of each epoch and strive to achieve the lowest possible loss. An optimization algorithm modifies model parameters such as weights to reduce loss and enhance accuracy. The stochastic gradient descent is one such effective DL training approach. For a scenario with a large number of training examples, it is both simple and quick to apply. However, it was challenging to parallelize this procedure with a graphical process unit (GPU) since it requires a number of different manual tuning schemes to achieve the correct parameters, and the process itself is primarily sequential. Thus, metaheuristic optimization algorithms can be used to find ideal weights for the DL models. Metaheuristic algorithms draw their inspiration from nature and can address optimization issues by emulating either natural or physical processes. In the fields of research, technology, engineering, and business, these optimization methods have been used to find solutions to a wide variety of optimization issues [17]. These algorithms are growing in popularity since the algorithms rely on straightforward ideas and are straightforward to put into practice [18]; additionally, for large-scale, complicated, nonlinear search and optimization issues, traditional optimization techniques may be insufficient and inappropriate [19]. The evaluation of general purposed metaheuristic approaches is broken down into a total of nine distinct categories. These categories include "biology-based, physics-based, swarm-based, social-based, music-based, chemistry-based, sport-based, mathematics-based, and hybrid" [20]. In swarm-based techniques, the population is initially generated at random and then evolved throughout successive generations. The best individuals from one generation are always mixed with those from the following generation to create the next generation of individuals, which is one of the strengths of these approaches. This makes it possible to achieve optimal population levels throughout several generations. These techniques attempt to imitate the social behavior of groups of animals. The particle swarm optimization (PSO) algorithm, which was developed by Eberhart [21], is currently the most often used. The various swarm-based optimization algorithms that we came across in our literature review are summarized in Table 1 below.

Table 1. Swarm-based Optimization Algorithms.

Algorithm	Inspiration	Year	Features	Challenges
ACO [22]	Ant Colony	2006	Ability to scale, maintain stability, and adapt to changing circumstances	Parameter initializations through trial-and-error
WSA [23]	Wasp	2007	Excels in solving dynamic SAT problems.	Better heuristics can lead to a more effective local search
DPO [24]	Dolphin	2009	Achieving a significant value in the initial few steps and breaking through local minimum	It can be compared and perhaps combined with others.
Ant Lion Optimizer (ALO) [25]	Ant Lion	2015	The algorithm has few adaptable parameters; therefore, it can solve varied situations.	There are different optimization problems that can be solved in different fields.
WOA [18]	Whale	2016	The algorithm can solve issues with unknown search spaces.	Binary and multi-objective versions are possible.
SSA [26]	Salpa	2017	Among the existing optimization algorithms, methods have merits based on simulations, results, findings, analyses, discussions, and conclusions.	It is recommended to tackle both single- and multi-objective issues in a variety of disciplines.
Grey Wolf Optimization (GWO) [27]	Wolf	2019	The findings demonstrated a significant level of local optima avoidance	The goals of this algorithm need to be multi-objective.

Table 1. *Cont.*

Algorithm	Inspiration	Year	Features	Challenges
Sea Lion Optimization (SLnO) [28]	Sea Lion	2019	The optimization results are competitive with other metaheuristic methods.	Recommended to simulate the algorithm on the real problem.
Butterfly Optimization Algorithm (BOA) [29]	Butterfly	2019	Simulation findings show that the algorithm performs well compared to competing algorithms.	modifying BOA and analyzing its outcomes after being applied to combinatorial issues would be an intriguing subject to look into.
Cat and Mouse Based Optimizer (CMBO) [30]	Cat and Mice	2021	The performance of the CMBO demonstrated its superiority and more competitiveness in comparison to the other offered algorithms.	It is recommended to test the method with real-world problems when using binary and multi-objective versions of the algorithm.

Bird flocking, animal herd, bacterial development, and schools of fish are all inspirations for swarm intelligence algorithms. Figure 4 displays the overall framework that all metaheuristic algorithms adhere to.

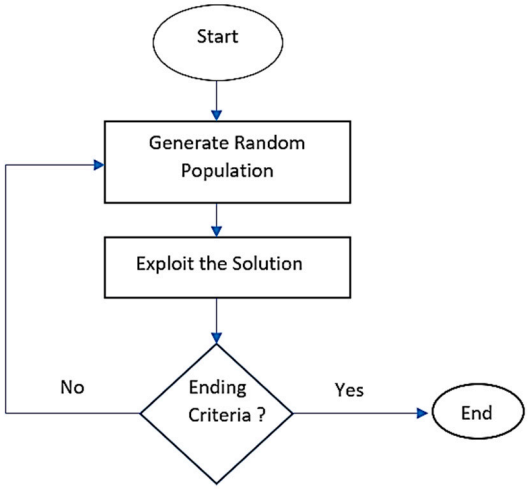

Figure 4. Generic Flowchart of Metaheuristic Algorithm.

However, there are a few old or standard algorithms that are still being utilized, despite the fact that they are not very accurate. Thus, two novel metaheuristic optimization algorithms were proposed in order to resolve the issue of low accuracy. The first optimization algorithm that is proposed is termed the seagull adapted elephant herding optimization algorithm (SAEHO) [31]. This method takes the logic of two independent metaheuristic optimization algorithms, namely, the elephant herding optimization (EHO) [32] and seagull optimization algorithm (SOA) [33], and combines them into a single solution. On the other hand, the second algorithm is referred to as self-upgraded cat and mouse optimization (SU-CMO) [34], and it is an improvement on the already-existing optimization technique known as cat- and mouse-based optimizer (CMBO) [30]. Consequently, both algorithms are put to use in the hybrid deep learning classifier model in order to achieve optimal weighting. The two approaches that were implemented in the current paper are as follows:

- In the first approach, two different types of deep learning models, namely, CNN and DBN, are combined to create a hybrid classifier. This classifier is then used in the problem of identifying intrusions into the IoT system. In order to perfect the model, the proposed SAEHO is utilized for its training.

- In the second approach, SU-CMO is put to use in order to train the hybrid classifier deep learning model that is made up of GRU and Bi-LSTM.

Following that, both of the suggested frameworks are contrasted with conventional and state of the art techniques. Despite the fact that many distinct intrusion detection systems are now being developed, there are still issues that might be improved. One of the issues that has to be properly managed is a higher rate of false errors, i.e., false positive and false negative. The objective of current research work is to develop a metaheuristic optimization algorithm for improving DL models for IoT attack classification.

Section 2 of this paper provides a literature review of the related work. Section 3 briefly describes two novel metaheuristic optimization algorithms. The IoT security attack classification framework is described in Section 4. The results are depicted in Section 5, and the conclusion and future work are provided in Section 6.

2. Related Work

A number of researchers have contributed to the categorization of IoT security threats and have also proposed optimization techniques [35,36].

A nature-inspired meta-heuristic optimization technique that they term the whale optimization algorithm (WOA) was proposed by Mirjalili et. al. [18] in 2016. The algorithm imitates whales' predatory behavior to solve the target problem. It comprises three operators to imitate the behavior of humpback whales, i.e., searching for prey, surrounding prey, and using bubble nets to catch their food. The WOA is evaluated using a total of 39 problems, 29 of which are mathematical optimization challenges and 6 of which are structural design issues. The findings showed that the WOA method has the potential to be highly successful in addressing real situations that include unknown search spaces. Slow convergence can be seen in WOA.

Seyedali et. al. [27] introduced a new metaheuristic optimization algorithm termed the grey wolf optimization (GWO), which was inspired by a group of grey wolves. It is a simulation of the social structure and hunting process of grey wolves found in nature. For the purpose of mimicking the hierarchy of leadership, four distinct varieties of grey wolves, designated as "alpha, beta, delta, and omega", are used. In the mathematical model, an alpha wolf is considered to be the optimal solution. The beta wolf, delta wolf and omega wolves are, respectively, regarded to be second best, third best, and the rest of the candidate solutions. However, this algorithm is susceptible to low precision, poor local searching, and slow convergence.

Raja et. al. [28] proposed a nature-inspired metaheuristic optimization method, named the sea lion optimization (SLnO) algorithm, that mimics the hunting techniques used by sea lions in their natural environment. Moreover, this research work was carried out on twenty-three different mathematical optimization problems in order to investigate the exploration phase, the exploitation phase, and the recommended method's convergence behavior. The algorithm avoids local optimum solutions and converges quickly over rounds.

Seyedali et. al. [37] presented an ant lion optimizer (ALO), which is a relatively new meta-heuristic algorithm that mathematically describes the natural interaction of ants and antlions. The method was designed to address optimization issues that consider random ant walks, building traps, entrapping ants in traps, collecting prey, and re-building traps. Ants' unpredictable walks assist to avoid becoming stuck in local optima; moreover, few tunable parameters make it easy to use.

In 2021, Laith et. al. [38] introduced the Aquila Optimizer (AO), a population-based optimization approach inspired by the activities of aquilas (a bird) in the wild as they are in the process of catching their meal. The algorithm is modeled after four different procedures, i.e., choosing the available search space, probing inside the confines of a divergent search, using resources within the confines of a convergent search space, and walk and catch the prey. Optimization begins the improvement methods by producing a random set of potential solutions (population). Through repetition, AO search techniques, i.e., "expanded exploration, narrowed exploration, expanded exploitation, and narrowed exploitation",

find near-optimal or best-obtained solutions. In complicated optimization situations, an AO may have low convergence or fall in sub-optimal areas.

In 2018, Arora et. al. [29] made an effort to find a solution to the global optimization problem by putting up a novel optimization algorithm named the butterfly optimization algorithm (BOA). The algorithm that was presented is a simulation of the butterfly's foraging and mating behavior. The performance of the method was analyzed and compared with that of several other metaheuristic algorithms after it was tested and verified on a set of thirty benchmark test functions. While BOA is applied to various fields of optimization problems due to its performance. it also has some cons, such as a smaller population and an increased chance of being stuck in a "local optimum".

In [39], Read et. al. provided an overview study that analyzes the false data injection (FDI) assault in the smart grid according to attack model, attack impact, and attack target. A number of important assessment criteria were utilized in relation to the needs of the power systems and the smart grid security in estimating the efficacy of the multiple cyberattack models that were found in the literature that was surveyed. This was implemented in order to determine the associated challenges with these models. In addition to this, several future research topics for FDI assaults are suggested as a means of enhancing the smart grid cybersecurity architecture.

In [40], Khosravani et. al. outlined the process of developing a knowledge-based system to enhance the conventional injection molding procedure through the incorporation of relevant information. The proposed system is to assist industries involved in injection molding by using cutting-edge technologies to enhance production and raise levels of efficiency. The study is concentrated on "simulation and generative design" for the optimization of production, "additive manufacturing", and "virtual reality". Research and experimentation have the potential to develop and enrich the present cost-effective and appropriate solutions that are the outcome of the knowledge-based system that is being proposed.

3. Proposed Metaheuristic Algorithms

3.1. Proposed Seagull Adapted Elephant Herding Optimization (SAEHO) Algorithm

The rationale of the EHO and SAO were combined in the SAEHO (Figure 5) that is presented. The drawback of EHO is that it does not use the needed data to find out what searches are going on now and in the future. In SAO, on the other hand, the constraints are very slow and tedious to figure out. When working through optimization challenges, this often becomes a difficulty. Due to this, we have combined the SOA with the EHO. There are reports that the hybrid models show promise for solving some search difficulties with improved outcomes [41]. EHO emulates the normal behavior of elephants, which is to live as a clan led by a female matriarch. These elephants live in harmony with their environment. The EHO may be broken down into its three primary rules as a guiding concept.

- The population is organized into a number of clans, and each clan has a predetermined ratio of males to females.
- There are always a certain number of male elephants that choose to be alone and live apart from their clans.
- The leader of each elephant clan is a female elephant, known as the matriarch of their clans.

In EHO, the elephants depict the decision variable or solution, whereas the clan of elephant indicates the recommended solution. The matriarch is the best solution and the male elephant who leaves the clan is considered the worst solution. There are mainly two operators that are used in EHO, i.e., the clan updating operator and the separating operator. In the clan updating operator, each clan is updating according to the EHO conventional

equation; however, as per the proposed SAEHO, the clan updating is performed through the SAO given in Equation (1).

$$R_b(\tilde{t}) = \widehat{B} \times \left(R_{best}(\tilde{t}) - R_{\widehat{l}}(\tilde{t}) \right) \quad (1)$$

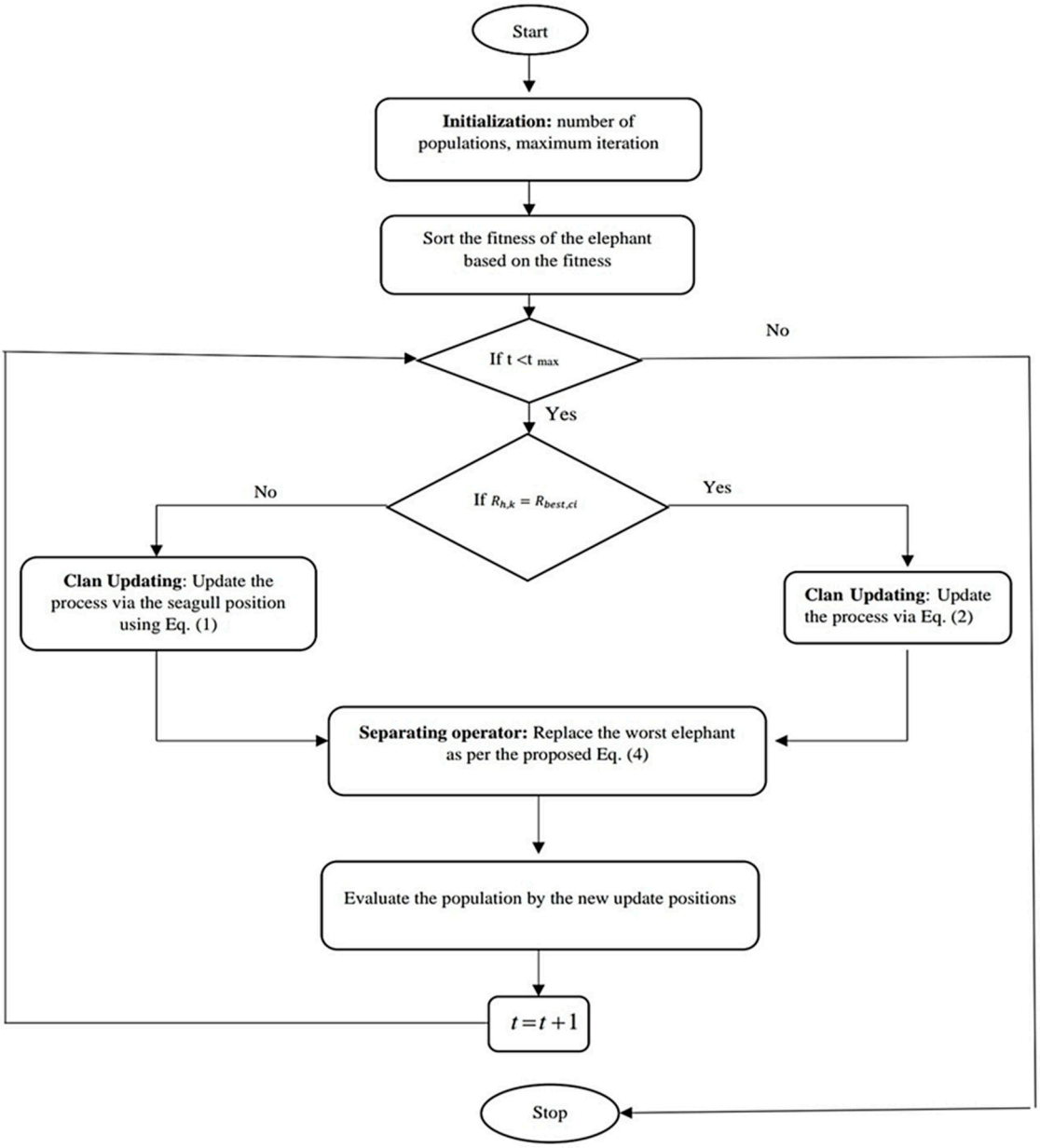

Figure 5. Flowchart for proposed SAEHO Algorithm.

R_b shows the location of the seagull search agent $R_{\widehat{l}}$ in the direction of the best fit search agent R_{best}. $\tilde{t}$ denotes the present iteration, whereas $\widehat{B}$ randomized for appropriate balancing. However, Equation (2) is used in order to update the best fit elephant.

$$R_{n,h,k} = \eta \times R_{cen,h} \quad (2)$$

In Equation (2), $R_{n,h,k}$ is a newly updated position, η belongs to [0,1], and $R_{cen,h}$ can be calculated by Equation (3).

$$R_{cen,h} = \frac{1}{G_h} \times \sum_{k-1}^{G_h} R_{h,k,\tilde{d}} \tag{3}$$

The separating is performed via Equation (4).

$$R_{worst,h} = R(R_{best} - R_{worst})_{min} \tag{4}$$

R_{min} represents the lowest bound, whereas R_{best} and R_{worst} indicate the best and the worst elephant in the clan h, respectively.

3.2. Self-Upgraded Cat and Mouse Optimizer (SU-CMO) Algorithm

The existing CMBO model [30] mimics the natural behavior of a cat and mouse race. After that, its mathematical model is shown so that it may be used to optimize solutions to a variety of issues. The search agents in the CMBO model are separated into two groups, one consisting of cats and the other of mice, which search the problem space using random movements. The model performs a two-phase update on the population members. The first phase involved simulating the cat's pursuit of mice by modeling their behavior, and in the second phase, the model simulates the mice's attempt to save their lives by running away to safe havens. As a result of the CMBO model's poor accuracy, several adjustments were made in an effort to improve the model's precision. Figure 6 is an illustration of the steps in the algorithm.

The fitness of each search agent is computed using Equation (5)

$$Obj = \min(Error) \tag{5}$$

The agents are ranked according to their fitness using Equations (6) and (7).

Where B^t denotes sorted population matrix, and Obj^t is considered the sorted objective function-based vector:

$$B^t = \begin{bmatrix} B_1^t \\ B_2^t \\ \vdots \\ B_C^t \end{bmatrix}_{C*m} = \begin{bmatrix} z_{1,1}^t & \cdots & z_{1,d}^t & \cdots & z_{1,m}^t \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ z_{i,1}^t & \cdots & z_{i,d}^t & \cdots & z_{i,m}^t \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ z_{C,1}^t & \cdots & z_{C,d}^t & \cdots & z_{C,m}^t \end{bmatrix}_{C*m} \tag{6}$$

$$Obj^t = \begin{bmatrix} Obj_1^t & min(Obj) \\ Obj_2^t & min(Obj) \\ \vdots & \vdots \\ Obj_C^t & min(Obj) \end{bmatrix}_{C*1} \tag{7}$$

In the Equations (8) and (9), M and C are utilized in order to choose the population of mice and cats.

$$M = \begin{bmatrix} M_1 = X_1^t \\ \vdots \\ M_i = X_i^t \\ \vdots \\ M_{C_m} = X_{C_m}^t \end{bmatrix}_{C_m*m} = \begin{bmatrix} z_{1,1}^t & \cdots & z_{1,d}^t & \cdots & z_{1,m}^t \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ z_{i,1}^t & \cdots & z_{i,d}^t & \cdots & z_{i,m}^t \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ z_{C_m,1}^t & \cdots & z_{C_m,d}^t & \cdots & z_{C_m,m}^t \end{bmatrix}_{C_m*m} \tag{8}$$

$$C = \begin{bmatrix} C_1 = X_{C_m+1}^t \\ \vdots \\ C_i = X_{C_m+j}^t \\ \vdots \\ C_{B_c} = X_{C_m+C_d}^t \end{bmatrix}_{C_d * m} = \begin{bmatrix} z_{C_m+1,1}^t & \cdots & z_{C_m+1,e}^t & \cdots & z_{C_m+1,m}^t \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ z_{C_m+j,1}^t & \cdots & z_{C_m+j,e}^t & \cdots & z_{C_m+j,m}^t \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ z_{C_m+C_d,1}^t & \cdots & z_{C_m+C_d,e}^t & \cdots & z_{C_m+C_d,m}^t \end{bmatrix}_{C_d * m} \quad (9)$$

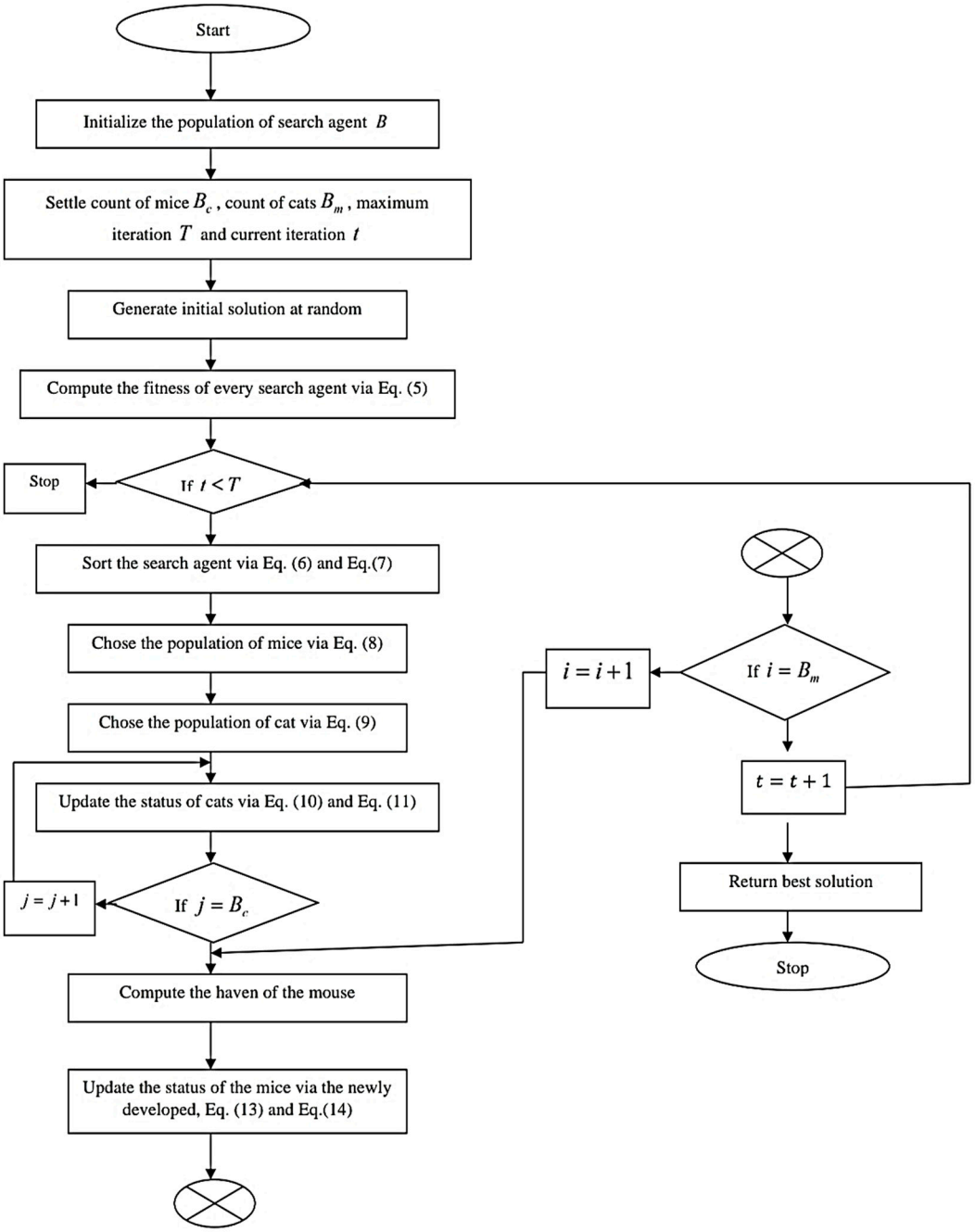

Figure 6. Self-Upgraded Cat and Mouse Optimization Algorithm.

The state of the cat may be updated by utilizing Equations (10) and (11), where C_j^{new} indicates the new status of the jth cat.

$$C_j^{new} = \left[C_{j,d} + r * \left(M_{k,d} - I * C_{j,d} \right) \right] \quad (10)$$

$$I = round(1 + rand) \tag{11}$$

However, according to the original CMBO, Equation (12) is used to update the status of mice, but according to the new proposed SU-CMO, the mice updating is performed using Equations (13) and (14). M_i denotes the position of the ith mouse, whereas M_i^{new} is the new position of the mouse in the ith position. The algorithm's notation is shown in Table 2.

$$M_i = \begin{cases} M_i^{new} & | \ F_i^{m,new} < F_i^m \\ M_i & | \ else \end{cases} \tag{12}$$

$$M_i = \begin{cases} M_i^{new} & | \ F_i^{m,new}.ra_1 < F_i^m \\ M_i & | \ else \end{cases} \tag{13}$$

$$M_i = \begin{cases} M_i^{new} & | \ F_i^{m,new}.ra_2 < F_i^m \\ M_i & | \ else \end{cases} \tag{14}$$

Table 2. Notation.

Symbols	Description
B^t	Sorted Population Matrix
Cm	Count of mice
Cd	Count of cat
D	Cat Population
Dj	ith cat
I	Iteration
M	Mice population
r	Random number within [0,1]
Obj^t	sorted objective function-based vector
$z_{i,d}^t$	ith population of sorted population matrix

The values of ra_1 and ra_2 are 1.25 and 1.75, respectively, which was considered based on the trial-and-error method. Figure 6 shows the model flowchart of the proposed SU-CMO.

The conditions for stopping the algorithm are determined by the current value of the t variable, which indicates the total number of iterations. When all iterations are finished, the ideal solution can be withdrawn at that point.

4. IoT Security Attacks Detection Framework

Both of the above-mentioned proposed algorithms, SAEHO and SU-CMO, are utilized in the process of detecting security threats in an IoT environment. Each of the algorithms have a separate IoT framework designed for it. Two IoT security threat detection frameworks, based on deep learning models, were developed. Consequently, we termed it a hybrid deep learning classifier since it utilizes two distinct deep learning models. The framework-I (Figure 7) combines two DL models for classifying attacks; specifically, it consists of a CNN and a DBN. However, the framework-II (Figure 8) uses the GRU and Bi-LSTM DL models. The framework-I optimized the CNN and DBN deep learning models using the proposed SAEHO optimization algorithm. In contrast, the framework-II used the proposed SU-CMO optimization method to optimize the Bi-LSTM deep learning model. Preprocessing, feature extraction, hybrid deep learning classifiers optimized by the suggested optimization method, and finally, attack classification outcomes are the standard operating procedures for both IoT frameworks. In the preprocessing phase, data are cleaned and prepared for analysis by removing outliers or null values and removing redundant or extraneous information. Following that, the statistical and higher-order statistical features are extracted, as part of the feature extraction process.

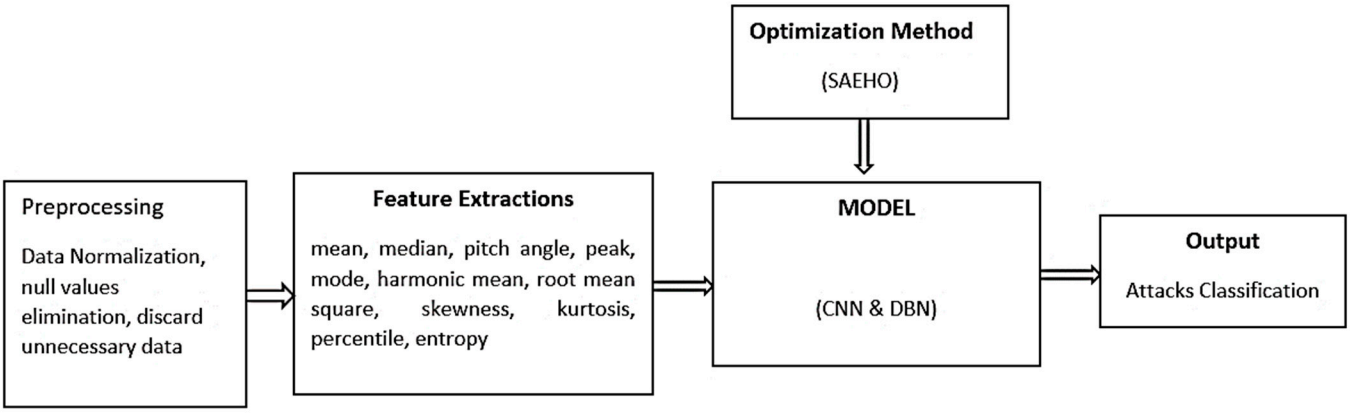

Figure 7. IoT Security Attack Detection Framework-I.

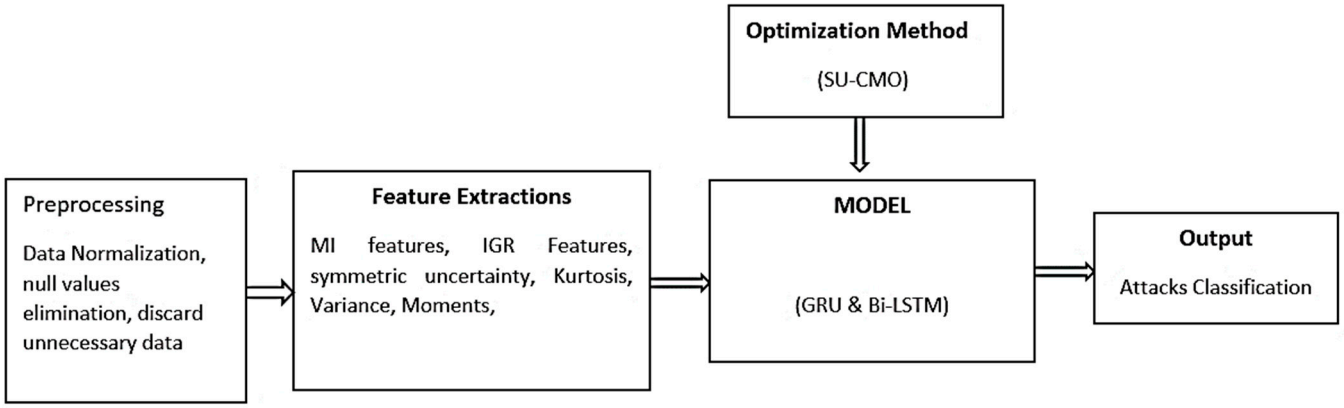

Figure 8. IoT Security Attack Detection Framework-II.

Statistical feature extraction: In this section, the statistical features known as the standard deviation (SD), mean, harmonic mean (HM), median, pitch, peak, pitch angle, root mean square (RMS), and amplitude are calculated.

Mean: The concept of "mean value" refers to the operation in which the total sum of all values is divided by the total number of all values.

Median: The term "median" refers to the operation of sorting the value that falls in the center of a dataset in descending order. If there are two values that fall in the center of the dataset, then the median of the data is determined by taking the mean of those two values.

SD: It is a measurement of the values that make up a set of dispersion or the degree of variance. A smaller standard deviation suggests that the values in question tend to be closer to the mean value, whereas a bigger SD shows that the values in question span across a wider range.

Mode: The term "mode" refers to the "value that appears most frequently in the dataset." It is one of the most common "central tendency measures" (i.e., to use with "nominal data"), which are characterized by entirely arbitrary classifications. The value in a set of observations that occurs most frequently is referred to as the mode of those observations.

HM: To calculate it, the total number of observations is divided by the reciprocal of each number in the series. The value is obtained. One sort of numerical average is referred to as the HM.

RMS: It is also known as the quadratic mean, and it is utilized in both the field of mathematics and the field of statistics. It is the square root of the total sum of squares for each observation's data.

Peak amplitude: The "peak amplitude" of a "sinusoidal waveform" is the point at which the level of a wave's largest positive or negative deviation from zero reference level is reached.

Pitch Angle: The angle formed between the longitudinal axis and the horizontal plane is denoted by this term.

Higher-order statistical feature extraction: Higher order statistical characteristics include the limitations listed below, such as "kurtosis, skewness, entropy, percentile, energy, mean, frequency", etc.

Skewness: It is a measurement of how much of a departure from the normal distribution the distribution of a random variable exhibits.

Kurtosis: The term "Kurtosis" refers to a statistical metric that determines the degree to which the tails of a distribution deviate from the tails of a normal distribution.

Entropy: It is a term that refers to the "average degree of information, surprise, or uncertainty that is inherent in the probable consequence of the variable according to the information theory".

Percentile: A score that falls at or below a given percentage of the total scores in a frequency distribution is referred to as a percentile. In statistics, a percentile refers to a score that falls at or below a particular percentage.

Variance: It is defined as the "mean squared disparity" between each and every data point and the calculated center of distribution for the population as a whole.

Moment: In probability theory and statistics, it refers to the point at which the probability distribution takes into account the arbitrary variable. It is the average value of an integer power that was defined, and its difference from the mean of the arbitrary variable is its power.

MI Feature: It is defined as the computation of information that is sent back and forth between two collections of random variables.

Symmetric Uncertainty: It calculates the features by basing them on the evaluated symmetric uncertainty correlation metrics between the class and the feature.

5. Results and Discussion

5.1. Experiment Setup

Python is used in the process of implementing both frameworks. Two different IoT datasets, i.e., dataset 1: https://research.unsw.edu.au/projects/unsw-nb15-dataset (accessed on 5 March 2022)., and dataset 2: https://cloudstor.aarnet.edu.au/plus/s/umT99TnxvbkkoE (accessed on 15 March 2022)., , are used to evaluate the performance of the framework. From dataset 1, attributes including "bytes, dport, dur, dpkts, dbytes, drate, mean, min, max, pkts, rate, sport, seq, stddev, sum, spkts, sbytes, srate" are used, from dataset 2, "ct_srv_src, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm, ct_src_ltm, ct_srv_dst, dur, dpkts, dbytes, dwin, dtcpb, dloss, dload, rate, spkts, sbytes, sttl, sloss, sinpkt, stcpb, smean, tcprtt, trans_depth" attributes are included. The leftover attributes that are not being used are removed from both databases.

Initial steps involve preprocessing the dataset following the statistical and higher order statistical feature extraction, including mean, median, standard deviation, kurtosis, entropy, variance, moment, etc. Various performance metrics are used to evaluate the framework performance, including accuracy, MCC, rand index, and F-measure. The hybrid classifier + SAEHO is compared with the traditional and state of the art methods including neural network (NN) [42], support vector machine (SVM) [43], decision tree (DT) [44], WOA, GWO, and SLnO; on the other hand, the hybrid classifier + SU-CMO is compared with NN, RNN, GRU, SVM, K-nearest neighbor (KNN), ALO, AO, BOA, and CMBO. It was found that both of the proposed models work more efficiently than traditional and state of the art methods.

There are a total of 120 instances that were collected from dataset 1, of which 60 are considered "Normal" and the remaining 60 are considered "Attack", which is DoS (Denial

of Service). However, in dataset 2, a total of 2000 instances were acquired, 1000 of which are classified as "Normal", while the other instances are classified as "Attack".

The term "Attack" is further subdivided into eight distinct types of IoT assaults, which are referred to, respectively, as "Analysis, Backdoor, DoS, Exploit, Fuzzers, Reconnaissance, Shell Code, and Worms". However, all of the different types of attacks fall under a single category, which is referred to simply as "Attack". Figure 9 depicts a chart depicting the categorization of the instances identified in datasets 1 and 2.

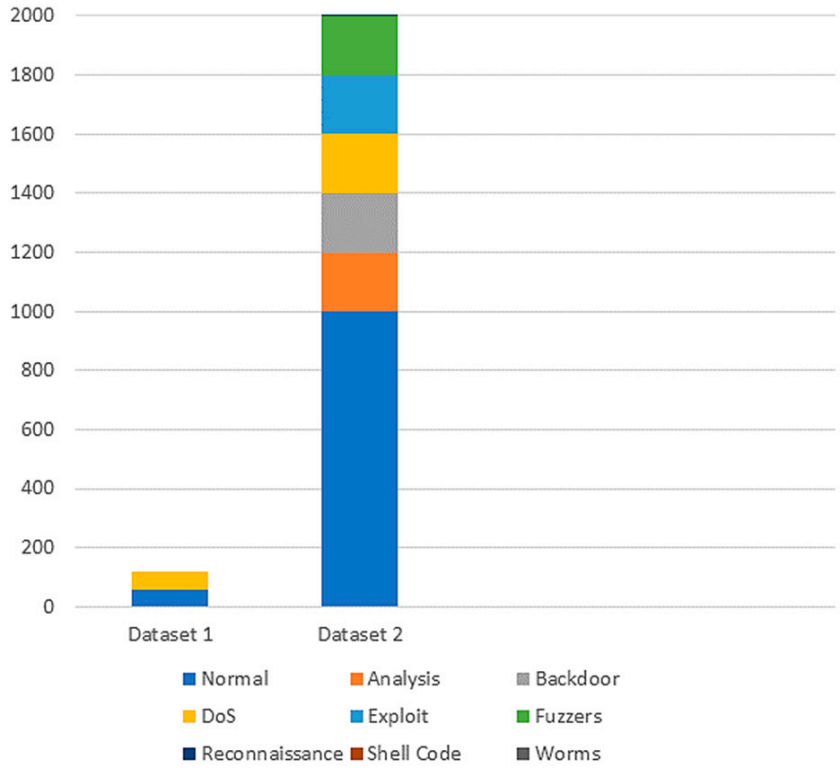

Figure 9. Attack instances in Dataset 1 and Dataset 2.

In CNN, the convolution layer uses the ReLU activation function, and it is followed by the leaky ReLU layer, the pooling layer, the flatten layer, and then, two dense layers. The ReLU and SoftMax activation functions are utilized by the dense layers. Additionally, the mean error squared loss function is employed in the CNN deep learning model. While working with DBN, the sigmoid activation function is used. On the other hand, in the Bi-LSTM and GRU deep learning model, ReLU and sparse_categorical_crossentropy is used as activation and loss function, respectively.

Table 3 provides details on the operating environment, including the necessary software and hardware.

Table 3. Hardware and Software Requirement.

Processor	**11th Gen Intel(R) Core (TM) i5-1135G7 @ 2.40GHz 2.42 GHz**
Installed RAM	16.0 GB (15.7 GB usable)
System type	64-bit operating system, x64 based processor
Operating System Edition	Windows 11 Home Single Language

5.2. Performance Analysis

When it comes to developing an efficient machine learning model, one of the most crucial phases is to conduct an evaluation of the performance of the machine learning model. A variety of measures are applied in order to assess the effectiveness of the model

or its level of excellence. In this study, we verified the model using four distinct measures: accuracy, rand index, F-measure, and MCC. The accuracy in Equation (15) is calculated by comparing the number of right predictions to the total number of predictions, and it is considered to be one of the most important classification metrics.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{15}$$

Utilizing the accuracy measure is beneficial when the target variable classes in the data are equally balanced. It is strongly advised that the accuracy measure be avoided whenever the target variable predominantly belongs to a single class. Rand index is another metric that is frequently utilized. It is a metric that may be utilized to evaluate the degree to which the outcomes of two distinct clustering strategies are comparable to one another. It is simple to determine the value using Equation (16).

$$Rand\ Index = \frac{Total\ number\ of\ agreeing\ pair}{Total\ number\ of\ pair} \tag{16}$$

The F-measure is derived in Equation (17) by determining the harmonic mean of the recall and precision. It makes it possible to evaluate a model by taking into consideration both its precision and its recall while only utilizing a single score. This is beneficial for explaining the performance of the model, as well as when comparing different models.

$$F - Measure = \frac{2 * (Precision * Recall\ )}{Precision + Recall} \tag{17}$$

where precision and recall are represented in Equations (18) and (19).

$$Precision = \frac{TP}{TP + FP} \tag{18}$$

$$Recall = \frac{TP}{TP + FN} \tag{19}$$

MCC in Equation (20) is a statistical rate that is more trustworthy and delivers a high score only if the prediction produced good results in all of the four areas of the confusion matrix, i.e., true positive (TP), true negative (TN), false positive (FP), and false negative (FN).

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{20}$$

Tables 4–6 provide the accuracy analysis of the SAEHO + hybrid classifier at learning rates of 60, 70, and 80 for dataset 1, whereas Tables 7–9 shows accuracy analysis for dataset 2. The SAEHO + hybrid classifier exhibits the maximum accuracy (0.908) on dataset 1 for a learning rate of 80. Moreover, for dataset 2, the maximum accuracy (0.928) is achieved with a learning rate of 80, outperforming WOA + hybrid classifiers, GWO + hybrid classifiers, SLnO + hybrid classifiers, NNs, SVMs, and DT. Similarly, SAEHO + hybrid classifier exhibits the best MCC, rand index, and F-measure values, i.e., 0.766, 0.978, and 0.79 for dataset 1 and 0.786, 0.998, and 0.81 for dataset 2.

Table 4. Performance of Hybrid Classifier + SAEHO for dataset 1 for 60 percent Learning Rate.

Model	WOA + Hybrid Classifier	GWO + Hybrid Classifier	SLnO + Hybrid Classifier	NN	SVM	DT	Proposed SAEHO + Hybrid Classifier
Accuracy	0.842	0.833	0.837	0.838	0.844	0.836	0.905
MCC	0.044	−0.008	0.015	0.021	0.133	0.008	0.751
Rand Index	0.917	0.912	0.915	0.915	0.918	0.914	0.977
F-Measure	0.131	0.083	0.105	0.11	0.220	0.098	0.776

Table 5. Performance of Hybrid Classifier + SAEHO for dataset 1 for 70 percent Learning Rate.

Model	WOA + Hybrid Classifier	GWO + Hybrid Classifier	SLnO + Hybrid Classifier	NN	SVM	DT	Proposed SAEHO + Hybrid Classifier
Accuracy	0.835	0.838	0.838	0.825	0.837	0.825	0.908
MCC	0.007	0.021	0.023	−0.058	0.094	−0.058	0.766
Rand Index	0.914	0.915	0.915	0.908	0.914	0.908	0.978
F-Measure	0.097	0.110	0.112	0.037	0.185	0.037	0.790

Table 6. Performance of Hybrid Classifier + SAEHO for dataset 1 for 80 percent Learning Rate.

Model	WOA + Hybrid classifier	GWO + Hybrid Classifier	SLnO + Hybrid Classifier	NN	SVM	DT	Proposed SAEHO + Hybrid Classifier
Accuracy	0.873	0.874	0.922	0.879	0.880	0.913	0.916
MCC	0.082	0.088	0.124	0.017	0.153	0.171	0.720
Rand Index	0.948	0.948	0.997	0.960	0.956	0.983	0.992
F-Measure	0.168	0.173	0.211	0.111	0.242	0.252	0.750

Table 7. Performance of Hybrid Classifier + SAEHO for dataset 2 for 60 percent Learning Rate.

Model	WOA + Hybrid Classifier	GWO + Hybrid Classifier	SLnO + Hybrid Classifier	NN	SVM	DT	Proposed SAEHO + Hybrid Classifier
Accuracy	0.843	0.844	0.842	0.829	0.840	0.853	0.896
MCC	0.052	0.058	0.044	−0.032	0.113	0.111	0.700
Rand Index	0.918	0.918	0.917	0.910	0.916	0.923	0.972
F-Measure	0.138	0.143	0.131	0.061	0.202	0.192	0.730

Table 8. Performance of Hybrid Classifier + SAEHO for dataset 2 for 70 percent Learning Rate.

Model	WOA + Hybrid Classifier	GWO + Hybrid Classifier	SLnO + Hybrid Classifier	NN	SVM	DT	Proposed SAEHO + Hybrid Classifier
Accuracy	0.872	0.863	0.917	0.888	0.884	0.896	0.925
MCC	0.074	0.021	0.095	0.071	0.173	0.068	0.771
Rand Index	0.947	0.942	0.995	0.965	0.958	0.974	0.997
F-Measure	0.161	0.113	0.185	0.160	0.260	0.158	0.796

Table 9. Performance of Hybrid Classifier + SAEHO for dataset 2 for 80 percent Learning Rate.

Model	WOA + Hybrid Classifier	GWO + Hybrid Classifier	SLnO + Hybrid Classifier	NN	SVM	DT	Proposed SAEHO + Hybrid Classifier
Accuracy	0.865	0.868	0.918	0.875	0.877	0.885	0.928
MCC	0.037	0.051	0.103	−0.008	0.134	0.001	0.786
Rand Index	0.944	0.945	0.995	0.958	0.954	0.968	0.998
F-Measure	0.127	0.140	0.192	0.087	0.225	0.097	0.810

Tables 10–12 compare the results of the SU-CMO + hybrid classifier with AO + hybrid classifier, ALO + hybrid classifier, CMBO + hybrid classifier, NN, RNN, and GRU for datasets 1, whereas Tables 13–15 compare the results for dataset 2. They demonstrate the maximum performance value in terms of accuracy, F-measure, MCC, and rand index for both datasets. Figures 9 and 10 depict the overall performance comparison of SAEHO + hybrid with the SU-CMO + hybrid classifier in terms of accuracy, rand index, MCC, and F-measure.

Table 10. Performance of SU-CMO + Hybrid Classifier for Dataset 1 for 60 Percent Learning Rate.

Metric	AO + Hybrid Classifier	ALO + Hybrid Classifier	CMBO + Hybrid Classifier	NN	RNN	GRU	SU-CMO + Hybrid Classifier
Accuracy	0.799	0.815	0.606	0.818	0.588	0.501	0.848
F-measure	0.758	0.799	0.362	0.831	0.692	0.550	0.840
MCC	0.601	0.634	0.353	0.641	0.248	0.016	0.701
Rand index	0.889	0.898	0.778	0.908	0.767	0.707	0.921

Table 11. Performance of SU-CMO + Hybrid Classifier for Dataset 1 for 70 Percent Learning Rate.

Metric	AO + Hybrid Classifier	ALO + Hybrid Classifier	CMBO + Hybrid Classifier	NN	RNN	GRU	SU-CMO + Hybrid Classifier
Accuracy	0.598	0.655	0.612	0.688	0.632	0.500	0.772
F-measure	0.284	0.468	0.392	0.741	0.700	0.538	0.851
MCC	0.254	0.383	0.334	0.408	0.329	0.006	0.551
Rand index	0.756	0.801	0.791	0.829	0.795	0.702	0.878

Table 12. Performance of SU-CMO + Hybrid Classifier for Dataset 1 for 80 Percent Learning Rate.

Metric	AO + Hybrid Classifier	ALO + Hybrid Classifier	CMBO + Hybrid Classifier	NN	RNN	GRU	SU-CMO + Hybrid Classifier
Accuracy	0.678	0.616	0.661	0.786	0.625	0.637	0.837
F-measure	0.648	0.703	0.527	0.798	0.689	0.684	0.823
MCC	0.361	0.294	0.402	0.573	0.279	0.291	0.681
Rand index	0.823	0.785	0.816	0.886	0.790	0.798	0.915

Table 13. Performance of SU-CMO + Hybrid Classifier for Dataset 2 for 60 Percent Learning Rate.

Metric	AO + Hybrid Classifier	ALO + Hybrid Classifier	CMBO + Hybrid Classifier	NN	RNN	GRU	SU-CMO + Hybrid Classifier
Accuracy	0.567	0.812	0.761	0.790	0.692	0.662	0.817
F-measure	0.681	0.814	0.774	0.786	0.727	0.641	0.806
MCC	0.186	0.612	0.518	0.580	0.390	0.327	0.640
Rand index	0.753	0.898	0.869	0.891	0.833	0.813	0.904

Table 14. Performance of SU-CMO + Hybrid Classifier for Dataset 2 for 70 Percent Learning Rate.

Metric	AO + Hybrid Classifier	ALO + Hybrid Classifier	CMBO + Hybrid Classifier	NN	RNN	GRU	SU-CMO + Hybrid Classifier
Accuracy	0.589	0.678	0.579	0.708	0.629	0.541	0.823
F-measure	0.688	0.524	0.651	0.724	0.657	0.632	0.823
MCC	0.220	0.406	0.187	0.419	0.258	0.091	0.681
Rand index	0.760	0.816	0.766	0.841	0.791	0.735	0.915

Table 15. Performance of SU-CMO + Hybrid Classifier for Dataset 2 for 80 Percent Learning Rate.

Metric	AO + Hybrid Classifier	ALO + Hybrid Classifier	CMBO + Hybrid Classifier	NN	RNN	GRU	SU-CMO + Hybrid Classifier
Accuracy	0.791	0.675	0.595	0.767	0.638	0.680	0.844
F-measure	0.771	0.542	0.326	0.787	0.674	0.686	0.841
MCC	0.573	0.421	0.301	0.546	0.285	0.369	0.700
Rand index	0.89	0.821	0.771	0.875	0.798	0.829	0.927

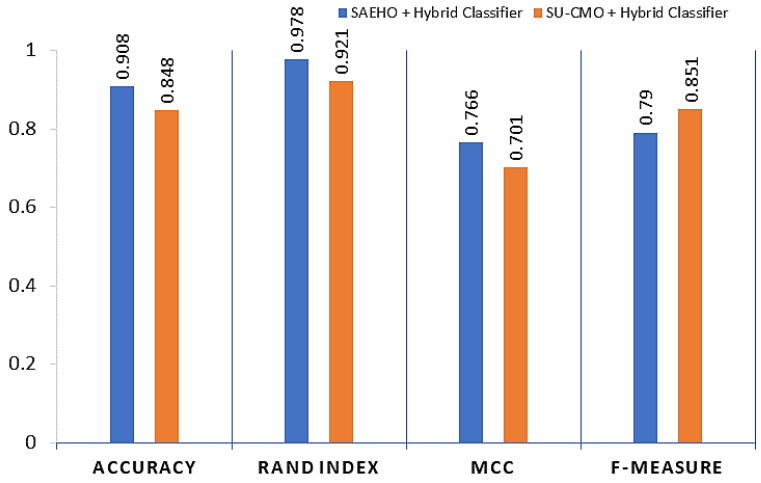

Figure 10. Performance of SAEHO + Hybrid Classifier and SU-CMO + Hybrid Classifier for Dataset 1.

Using the SAEHO + hybrid classifier, the mean values for dataset 1 are 0.067815 and the standard deviation is 0.00845, while for dataset 2, values are 0.056796 and 0.013462, for mean and standard deviation, respectively. In a similar manner, the mean and standard deviation values for the SU-CMO + hybrid classifier are 0.052639 and 0.01129 for dataset 1.

In addition, the mean and standard deviation values for dataset 2 are 0.070559 and 0.007432, respectively.

In Figures 10 and 11, SAEHO + hybrid classifier and SU-CMO + hybrid classifier show promising results in classifying IoT security attacks over various methods for both datasets. However, dataset 2 contains various kinds of IoT security attacks, including DoS, reconnaissance, analysis, exploit, shell code, backdoor, fuzzer, and worms, but both frameworks, i.e., SAEHO + hybrid classifier and SU-CMO + hybrid classifier, classify these attacks into a single category, i.e., "Attack". Additionally, there is no provision to take the action to prevent the attack when it occurs.

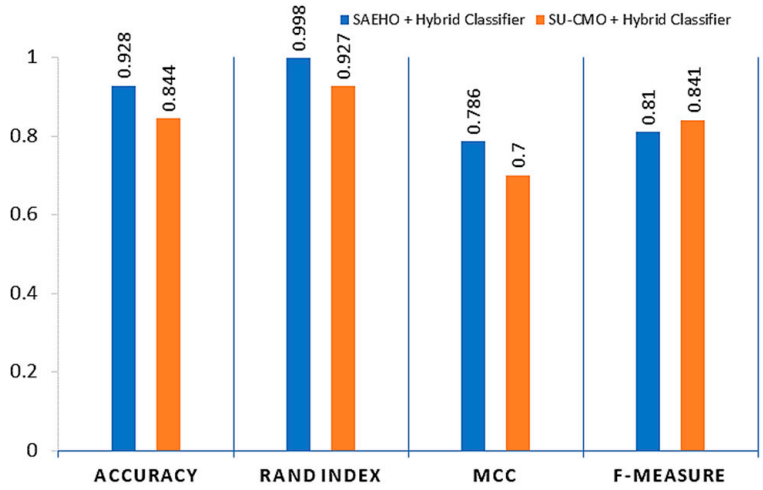

Figure 11. Performance of SAEHO + Hybrid Classifier and SU-CMO + Hybrid Classifier for Dataset 2.

6. Conclusions and Future Works

The optimization approach is an essential part of the training process for the machine learning models. Even though there are a number of effective search optimization techniques and algorithms, there is yet no optimum algorithm that is certain to work for every situation. Consequently, new optimization algorithms are regularly presented, or some efficient adjustments to existing algorithms have been accomplished. In this paper, two different metaheuristic optimization algorithms, namely, SAEHO and SU-CMO, were presented in order to train the DL models that were used to categorize several types of security threats that may occur in an IoT environment. SAEHO is a hybrid algorithm that combines two distinct algorithms, namely, EHO and SOA, while on the other hand, SU-CMO is the improved version of existing CMBO optimization algorithm. Consequently, the CNN and DBN DL models in framework-I are trained with the SAEHO optimization algorithm, while the Bi-LSTM model in framework-II is trained with the SU-CMO optimization method. Both the frameworks are validated using two separate datasets, i.e., dataset 1 and dataset 2. Four distinct performance metrics, i.e., accuracy, rand index, f-measure, and MCC, are utilized to demonstrate the validity of both frameworks. The proposed frameworks provide better outcomes when contrasted with either conventional or state-of-the-art approaches. The proposed optimization strategy has the potential to improve the accuracy of a wide variety of DL models when employed in the appropriate context. However, there is no assurance that the best option is among the solutions considered. As a result, the metaheuristic solution to an optimization issue must be viewed as good and not optimal. A significant portion of this paper is focused on the discussion for maintaining the safety of the IoT network, so the suggested optimization algorithm may be assessed in terms of time complexity and search space as a future work reference.

Author Contributions: Conceptualization, A.S., N.S.G. and P.G.; methodology, A.S., N.S.G., P.G., P.K.S. and W.-C.H.; writing—original draft preparation, A.S., N.S.G., P.G., P.K.S. and W.-C.H.; writing—review and editing, A.S., N.S.G., P.G., P.K.S. and W.-C.H.; supervision, N.S.G., P.G., P.K.S. and W.-C.H. All authors have read and agreed to the published version of the manuscript. All authors have equal contribution in this manuscript.

Funding: This research is under the sponsored grant: National Science and Technology Council, Taiwan (MOST 111-2410-H-161-001).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data are available on request from the submitting author.

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

References

1. Kumar, S.; Tiwari, P.; Zymbler, M. Internet of Things is a revolutionary approach for future technology enhancement: A review. *J. Big Data* **2019**, *6*, 111. [CrossRef]
2. Chaabouni, N.; Mosbah, M.; Zemmari, A.; Sauvignac, C.; Faruki, P. Network Intrusion Detection for IoT Security Based on Learning Techniques. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2671–2701. [CrossRef]
3. Ahmad, R.; Alsmadi, I. Machine learning approaches to IoT security: A systematic literature review [Formula presented]. *Internet Things* **2021**, *14*, 2021. [CrossRef]
4. What is Machine Learning? | IBM. Available online: https://www.ibm.com/cloud/learn/machine-learning (accessed on 1 August 2022).
5. Threat Landscape Trends—Q1 2020 | Broadcom Software Blogs. Available online: https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/threat-landscape-q1-2020 (accessed on 1 August 2022).
6. Cisco Annual Internet Report—Cisco Annual Internet Report (2018–2023) White Paper—Cisco. Available online: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (accessed on 1 August 2022).
7. De Assis, M.V.O.; Carvalho, L.F.; Rodrigues, J.J.P.C.; Lloret, J.; Proença, M.L. Near real-time security system applied to SDN environments in IoT networks using convolutional neural network. *Comput. Electr. Eng.* **2020**, *86*, 106738. [CrossRef]
8. Liang, F.; Yu, W.; Liu, X.; Griffith, D.; Golmie, N. Toward Edge-Based Deep Learning in Industrial Internet of Things. *IEEE Internet Things J.* **2020**, *7*, 4329–4341. [CrossRef]
9. Kociołek, M.; Kozłowski, M.; Cardone, A. A Convolutional Neural Networks-Based Approach for Texture Directionality Detection. *Sensors* **2022**, *22*, 562. [CrossRef]
10. Marenić, E.; Seychal, G.; Passieux, J.C. Data driven approach in multiphysics framework: Application to coupled electro-mechanical problems. *Comput. Methods Appl. Mech. Eng.* **2022**, *395*, 114959. [CrossRef]
11. Sridhar, S.; Dhanasekaran, D.; Charlyn, G.; Latha, P. Content-Based Movie Recommendation System Using MBO with DBN. *Intell. Autom. Soft Comput.* **2023**, *35*, 3241–3257. [CrossRef]
12. DiPietro, R.; Hager, G.D. Deep learning: RNNs and LSTM. In *Handbook of Medical Image Computing and Computer Assisted Intervention*; Academic Press Inc.: Cambridge, MA, USA, 2020; pp. 503–519. [CrossRef]
13. Reddy, D.K.; Behera, H.S.; Nayak, J.; Vijayakumar, P.; Naik, B.; Singh, P.K. Deep neural network based anomaly detection in Internet of Things network traffic tracking for the applications of future smart cities. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4121. [CrossRef]
14. Gupta, T.K.; Raza, K. Optimization of ANN Architecture: A Review on Nature-Inspired Techniques. In *Machine Learning in Bio-Signal Analysis and Diagnostic Imaging*; Elsevier: Amsterdam, The Netherlands, 2019; pp. 159–182. [CrossRef]
15. Roy, P.K.; Tripathy, A.K.; Weng, T.H.; Li, K.C. Securing social platform from misinformation using deep learning. *Comput. Stand. Interfaces* **2023**, *84*, 103674. [CrossRef]
16. Cakir, S.; Toklu, S.; Yalcin, N. Rpl attack detection and prevention in the internet of things networks using a gru based deep learning. *IEEE Access* **2020**, *8*, 183678–183689. [CrossRef]
17. Hakim, W.L.; Rezaie, F.; Nur, A.S.; Panahi, M.; Khosravi, K.; Lee, C.W.; Lee, S. Convolutional neural network (CNN) with metaheuristic optimization algorithms for landslide susceptibility mapping in Icheon, South Korea. *J. Environ. Manag.* **2022**, *305*, 114367. [CrossRef]
18. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
19. Alatas, B.; Bingöl, H.; Bingol, H. Comparative assessment of light-based intelligent search and optimization algorithms. *Light Eng.* **2020**, *28*, 86–93. [CrossRef]
20. Akyol, S.; Alatas, B. Plant intelligence based metaheuristic optimization algorithms. *Artif. Intell. Rev.* **2017**, *47*, 417–462. [CrossRef]
21. Gad, A.G. Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review. *Arch. Comput. Methods Eng.* **2022**, *29*, 2531–2561. [CrossRef]

22. Akhtar, A. Evolution of Ant Colony Optimization Algorithm—A Brief Literature Review. *arXiv* **2019**, arXiv:1908.08007. [CrossRef]
23. Pinto, P.C.; Runkler, T.A.; Sousa, J.M.C. Wasp Swarm Algorithm for Dynamic MAX-SAT Problems. In Proceedings of the 8th International Conference on Adaptive and Natural Computing Algorithms, Part I (ICANNGA '07), Warsaw, Poland, 11–14 April 2007; Springer: Berlin/Heidelberg, Germany; pp. 350–357. [CrossRef]
24. Dhanya, D.; Arivudainambi, D. Dolphin partner optimization based secure and qualified virtual machine for resource allocation with streamline security analysis. *Peer Peer Netw. Appl.* **2019**, *12*, 1194–1213. [CrossRef]
25. Kumar, S.; Kumar, A. A brief review on antlion optimization algorithm. In Proceedings of the IEEE 2018 International Conference on Advances in Computing, Communication Control and Networking, ICACCCN, Greater Noida, India, 12–13 October 2018; pp. 236–240. [CrossRef]
26. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]
27. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
28. Masadeh, R.; Mahafzah, B.A.; Sharieh, A. Sea Lion Optimization algorithm. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 388–395. [CrossRef]
29. Arora, S.; Singh, S. Butterfly optimization algorithm: A novel approach for global optimization. *Soft Comput.* **2019**, *23*, 715–734. [CrossRef]
30. Dehghani, M.; Hubálovský, Š.; Trojovský, P. Cat and mouse based optimizer: A new nature-inspired optimization algorithm. *Sensors* **2021**, *21*, 5214. [CrossRef] [PubMed]
31. Sagu, A.; Gill, N.S.; Gulia, P. Hybrid Deep Neural Network Model for Detection of Security Attacks in IoT Enabled Environment. *Int. J. Adv. Comput. Sci.Appl.* **2022**, *13*, 2022. [CrossRef]
32. Elhosseini, M.A.; el Sehiemy, R.A.; Rashwan, Y.I.; Gao, X.Z. On the performance improvement of elephant herding optimization algorithm. *Knowl. Based Syst.* **2019**, *166*, 58–70. [CrossRef]
33. Dhiman, G.; Kumar, V. Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems. *Knowl. Based Syst.* **2019**, *165*, 169–196. [CrossRef]
34. Sagu, A.; Gill, N.S.; Gulia, P.; Chatterjee, J.M.; Priyadarshini, I. A Hybrid Deep Learning Model with Self-Improved Optimization Algorithm for Detection of Security Attacks in IoT Environment. *Future Internet* **2022**, *14*, 301. [CrossRef]
35. Anand, P.; Singh, Y.; Selwal, A.; Singh, P.K.; Felseghi, R.A.; Raboaca, M.S. IoVT: Internet of Vulnerable Things? Threat Architecture, Attack Surfaces, and Vulnerabilities in Internet of Things and Its Applications towards Smart Grids. *Energies* **2020**, *13*, 4813. [CrossRef]
36. Malhotra, P.; Singh, Y.; Anand, P.; Bangotra, D.K.; Singh, P.K.; Hong, W.-C. Internet of Things: Evolution, Concerns and Security Challenges. *Sensors* **2021**, *21*, 1809. [CrossRef]
37. Mirjalili, S. The ant lion optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [CrossRef]
38. Abualigah, L.; Yousri, D.; Elaziz, M.A.; Ewees, A.A.; Al-qaness, M.A.A.; Gandomi, A.H. Aquila Optimizer: A novel meta-heuristic optimization algorithm. *Comput. Ind. Eng.* **2021**, *157*, 107250. [CrossRef]
39. Reda, H.T.; Anwar, A.; Mahmood, A. Comprehensive survey and taxonomies of false data injection attacks in smart grids: Attack models, targets, and impacts. *Renew. Sustain. Energy Rev.* **2022**, *163*, 112423. [CrossRef]
40. Khosravani, M.R.; Nasiri, S.; Reinicke, T. Intelligent knowledge-based system to improve injection molding process. *J. Ind. Inf. Integr.* **2022**, *25*, 100275. [CrossRef]
41. Beno, M.M.; Valarmathi, I.R.; Swamy, S.M.; Rajakumar, B.R. Threshold prediction for segmenting tumour from brain MRI scans. *Int. J. Imaging Syst. Technol.* **2014**, *24*, 129–137. [CrossRef]
42. Mohan, Y.; Chee, S.S.; Xin DK, P.; Foong, L.P. Artificial Neural Network for Classification of Depressive and Normal in EEG. In Proceedings of the 2016 IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES), Kuala Lumpur, Malaysia, 4–8 December 2016.
43. Cervantes, J.; Garcia-Lamont, F.; Rodríguez-Mazahua, L.; Lopez, A. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing* **2020**, *408*, 189–215. [CrossRef]
44. Lan, T.; Hu, H.; Jiang, C.; Yang, G.; Zhao, Z. A comparative study of decision tree, random forest, and convolutional neural network for spread-F identification. *Adv. Space Res.* **2020**, *65*, 2052–2061. [CrossRef]