Enhancing Network Anomaly Intrusion Detection with IoT Data-driven BOA- CNN-BiGRU-AAM - Net Classification

[1]Suresh G, [2] Sathya M, [3]Arthi D and [4]Arulkumaran G
[1]Department of Artificial Intelligence and Data Science, Kings Engineering College, Sriperumbudur, Tamil Nadu, India.
[2]Department of Information Technology, Nadar Saraswathi College of Engineering and Technology, Theni, India.
[3]Department of Business Administration, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, Tamil Nadu, India.
[4]School of Computing and Information Technology, REVA University, Bengaluru, India.
[1]drsureshkec@gmail.com, [2]msathya15@gmail.com, [3]arthid_mba@avinuty.ac.in, [4]erarulkumaran@gmail.com

Correspondence should be addressed to Suresh G : drsureshkec@gmail.com

Article Info
Journal of Machine and Computing (http://anapub.co.ke/journals/jmc/jmc.html)
Doi: https://doi.org/10.53759/7669/jmc202404073
Received 23 December 2023; Revised from 25 April 2024; Accepted 24 June 2024
Available online 05 July 2024.

Abstract - Network security is one of the key components of cybersecurity anomaly intrusion detection, which is responsible for identifying unusual behaviours or activities within a network that might indicate possible security breaches or threats. In this suggested intrusion detection system (IDS), network traffic data is continuously monitored via anomaly detection. The study makes utilising one of the most recent datasets to spot unusual behaviour in networks connected to the Internet of Things, the IoTID20 dataset, to facilitate this process. The preprocessing stage involves painstaking steps for smoothing, filtering, and cleaning the data. The Pinecone Optimisation algorithm (PCOA), a novel optimizer inspired by nature, is introduced in this study for the feature selection process. PCOA seeks to increase the effectiveness of feature selection while drawing inspiration from the various ways that pine trees reproduce, such as pollination and the movement of pinecones by animals and gravity. Moreover, IDS is classified using Bidirectional Gated Recurrent Unit–Additive Attention Mechanism Based on Convolutional Neural Networks (CNN-BiGRU-AAM), which makes use of deep learning's capabilities for efficient classification tasks. In addition, this work presents the Botox Optimisation Algorithm (BOA) for hyperparameter tuning, which is modelled after the way Botox functions in human anatomy. BOA uses a human-based method to adjust the hyperparameters of the model to attain the best accuracy. The results of the experiments show that the suggested methodologies are effective in improving network anomaly intrusion detection systems, with a maximum accuracy of 99.45%.

Keywords - Intrusion Detection System, Pinecone Optimization Algorithm, Botox Optimization Algorithm, Long Short-Term Memory, Transformer.

I. INTRODUCTION

The widespread use of IoT devices has increased efficiency and brought about unprecedented convenience, but it has also led to serious security and reliability concerns [1]. Making sure that devices are communicating securely is crucial since Internet of Things systems are getting more intricate and networked [2]. Mechanisms for managing trust and reputation are essential to ensuring the dependability and integrity of IoT ecosystems, especially in light of the growing number of attacks on these systems that affect everything from smart grids to the healthcare industry [3]. The automated configuration processes that allow IoT devices to self-configure and connect to networks on their own present one of the main security challenges [4]. Although these plug-and-play features increase adaptability and usability, they also introduce vulnerabilities that hackers could take advantage of. Attacks known as denial-of-service (DoS) or distributed DoS (DDoS) are very dangerous because they can disrupt services and cost organisations a lot of money [5].

In order to protect IoT networks from DoS attacks, strong methods and strategies are needed [6]. The integration of machine learning techniques into IDS is one example of the more sophisticated approaches that are becoming increasingly recognised as necessary, in contrast to the traditional defences that involve multiple verifications and network traffic filtration [7]. But despite the literature's emphasis on urgency, there's still a gap in the effective application of such cutting-edge

techniques [8,9]. Furthermore, security concerns remain as IoT systems develop to create smart environments that maximise human comfort and efficiency [10,11]. These worries are made worse by the enormous amount of IoT devices that are connected, which is predicted to surpass 22 billion in the upcoming years. Alarms are being raised across a number of industries due to the current state of many connected devices, which lacks proper security and privacy protection [12]. One of the most difficult parts of IoT security is node heterogeneity, which makes security even more difficult. To fully address these issues, traditional security techniques like intrusion detection systems, access control, and authentication are insufficient on their own [13,14].

Because deep learning offers sophisticated threat detection and mitigation capabilities, it is essential for enhancing the security of IoT systems [15,16]. IoT devices can use deep learning algorithms to analyse vast volumes of information generated by networks and identify anomalies that could be signs of security breaches, including DoS attacks [17]. Deep learning models, like recurrent neural networks (RNNs) and convolutional neural networks (CNNs), are excellent at identifying patterns and behaviours, which allows them to recognise and react to new threats instantly [18]. Furthermore, deep learning enables IoT devices to dynamically modify and enhance their defence mechanisms, enhancing resistance against changing attack tactics [19]. IoT ecosystems can strengthen their defences and successfully reduce security risks by incorporating deep learning into intrusion detection systems (IDS) and security frameworks. This will improve the general dependability and credibility of IoT communications [20].

Motivation

Research on DoS attacks in network IDS is driven by the growing number and intensity of cyberattacks that target online services and critical infrastructure. The availability, integrity, and functionality of networks are severely jeopardised by DoS attacks, so it is imperative to create effective detection and mitigation techniques. In addition, since attackers are always changing their strategies, it is critical to recognise and neutralise new DoS attack techniques in order to protect digital assets and guarantee the continuous provision of critical services. By tackling these issues through creative research, cybersecurity defences are strengthened and resilience against changing cyberthreats is encouraged.

Main Contributions

- Utilization of the IoTID20 dataset, a recent and relevant database to identify unusual behaviour in Internet of Things networks, providing a realistic basis for evaluation.
- Implementation of meticulous data preprocessing techniques including cleaning, filtering, and smoothing to ensure data quality and reliability.
- Introduction of the Pine Cone Optimization algorithm (PCOA) for feature selection, inspired by natural mechanisms of pine tree reproduction, enhancing feature selection efficiency.
- Application of CNN-BiGRU-AAM for classification of intrusion detection events, leveraging deep learning capabilities for effective classification tasks.
- Introduction of the Botox Optimization Algorithm (BOA) for hyperparameter tuning, drawing inspiration from human anatomy to achieve optimal accuracy in model tuning.
- Demonstration of significant improvement with experimental results showcasing a maximum accuracy of 99.45%, demonstrating the potency of the suggested techniques in enhancing network anomaly intrusion detection systems.

Organization of the paper

This is how the rest of the essay is organised. A selection of noteworthy, related literature will be given in Section 2. A description is given in Section 3 of the recommended approach. Explanation of the experiments as well as outcomes is given in Section 4. Section 5 summarises the findings and offers some conclusions.

II. RELATED WORKS

A dense channel-spatial attention model was presented by Safarov et al. [21] to identify and categorise DoS and distributed denial-of-service (DDoS) attacks. Their approach shows great accuracy and addresses the problems of uneven data. It enhances the cybersecurity defences of servers and websites by effectively detecting and classifying DoS and DDoS threats. Their model outperforms existing intrusion detection algorithms, obtaining remarkable accuracy rates of 99.38%, 99.26%, and 99.43%, accordingly, for the datasets UNSW_NB15, CICIDS2017, and Bot-IoT. These findings demonstrate its ability to precisely identify and classify a variety of attack kinds, improving the precision and strength of IDS.

The DoS assaults were the main focus of the Deep Learning-based IDS described by Meddeb, R., et al. [22] in tagged datasets that are employed to detect intrusions. It includes a broad spectrum of possible attacks that have an impact on traffic routing on mobile networks. The Stacked AE-IDS method models a high-level overview of pertinent components while reducing coupling, which enhances IDSs' ability to detect attacks in MANETs. Because it concentrates on DoS incidents and their effects on mobile networks' capacity to route traffic, this technique is crucial for MANET security. By detecting different attacks, especially denial-of-service (DoS) attacks, and understanding their impact on the routed services

that mobile networks offer, the Stacked AE-IDS technique can enhance the efficacy of intrusion detection systems (IDSs) and strengthen the defences of metropolitan area networks (MANNETs).

The key conclusion of this study by Ponnusamy, V., et al. [23] The current machine learning models are not well-trained to identify breaches specific to the IoT due to a lack of network traces. Specifically, the Knowledge Discovery in Databases (KDD) Cup database is examined to highlight the difficulties in creating wireless intrusion detection systems using the data properties that are already in place. Several recommendations are made to guarantee that traffic capture methods used by a wireless network (WN) are future-proof. The research paper's first section looks at different placement strategies, data collection procedures, and intrusion detection techniques. This work's primary objective is to investigate the design difficulties associated with developing an IDS in a wireless environment. The complexity of the architectural design makes It is more challenging to create an IDS in a wireless environment than in a wired network environment. Thus, this study covers the fundamentals of wired detection and deployment techniques along with cellular networks' future wireless services and design difficulties. IoT, mobile ad hoc networks, and wireless sensor networks (WSN) are the three primary wireless environments to pay attention to because they are at the forefront of future developments and are frequently targeted by attacks. Thus, developing an IDS with a focus on wireless networks is essential.

In this study, Sbai, O. and Elboukhari, M. [24] recommended the deployment of a knowledge-based intrusion detection system (KBIDS) to protect MANETs from two types of DDoS attacks: SYN flooding and UDP/data DDoS attacks. The CICDDoS2019 dataset is used with the DL precise DNN algorithm. Simulation experiments indicate that Accuracy, precision, recall, and F1-score are just a few of the performance metrics that the suggested architectural paradigm has the potential to produce.

The authors of this research, Abbood, Z.A., et al. [25], centred on maintaining security norms through network attack mitigation, malicious node detection, and efficiency assessment. Using the three methods Feedforward-Neural Network (FNN), Cascading-Back-Propagation-Neural Network (CBPNN), and CBPNN (FFNN)—it was able to identify complex patterns in MANET. Intrusion detection system (IDS) performance is frequently improved by using CNN and these essential DNN building blocks, as well as by integrating IDS with machine learning (ML). Compared to their statistical and logical counterparts, machine learning (ML) approaches provide training capabilities for MANET networks and enable adaptability to a range of contexts. When it comes to average receiving packet (ARP) and end-to-end (E2E) parameters, the suggested model performs better than another current model.

Ninu, S.B., et al. [26] offer a MANET intrusion detection method in the proposed study that is based on Exponential-Henry Gas Solubility Optimisation (EHGSO), influenced by Deep Neuro Fuzzy Networks. In the initial stages of safe routing, the best routes are selected using the recently developed EHGSO algorithm. The fitness factors for these methods are connection quality, distance, energy, and neighbour quality. The Exponential Weighted Moving Average (EWMA) along with the Henry Gas Solubility Optimisation (HGSO) are both incorporated into the proposed EHGSO. At the base station, the intrusion detection phase is initiated in the second phase by obtaining Knowledge Discovery in Databases (KDD) attributes and altering transmitted data packets. Data augmentation is carried out after the KDD features are extracted. The recommended EHGSO technique is used to train a Deep Neuro Fuzzy Network before it is utilised for intrusion detection. The recommended approach outperforms all currently used technologies. In the absence of attacks, the recommended method yields the following values: 4.123, 0.086, 95.877%, 0.342 J, 134975 kbps, 0.950, and 0.924. Among these measurements are packet corruption, jitter, recall, accuracy, and Performance and Developmental Review (PDR).

A publication by Prasad, M., et al. [27] talked about setting up a network, creating data, labelling samples, extracting features, detecting intrusions, and evaluating a paradigm for effectiveness and reliability. The assessment model assesses the dependability and efficacy of various strategies in addition to the hardware dependability and performance of intrusion detection systems using a fuzzy logic system. The results demonstrate that a trade-off is made between multiple statistical performance indicators when there is an imbalanced ratio of samples. The suggested evaluation technique calculates an intrusion detection system's scheme reliability score by examining the two best results. The suggested detection technique performs better than current methods, according to experimental results in maintaining high scheme dependability.

Research Gaps

Multiple research gaps are found in the study of DoS attacks against network intrusion detection systems (IDS). First off, while research on DoS attack detection techniques has been done, mitigation techniques have not received as much attention, leaving networks open to protracted disruptions. Second, in order to update the training and assessment of IDS models, larger datasets that faithfully capture real-world DoS attack scenarios are required. Furthermore, the development of reliable and future-proof defence mechanisms is hampered by the understudied scalability and adaptability of IDS to changing DoS attack methods. Closing these gaps is essential to improving networks' resistance to denial-of-service (DoS) attacks and maintaining security and continuity.

III. PROPOSED METHODOLOGY

Dataset Description

IoTID20 Dataset

The IoTID20 dataset contains both typical (benign) traffic and various forms of IoT attacks, such as DDoS, DoS, Mirai, and ARP Spoofing. A dataset gathered from IoT ecosystems for smart homes is called IoTID20. A range of networked gadgets are commonly found in smart homes, including wireless access points (Wi-Fi), laptops, smartphones, tablets, artificial intelligence speakers (SKTNGU), and Wi-Fi cameras (EZVIZ). The IoT victim equipment in this dataset was represented by the cameras and AI speakers, while the attacking equipment was represented by the other equipment. Scanning, denial-of-service, and man-in-the-middle attacks were all simulated using Nmap tools. Attacks using the Mirai botnet were created independently on a laptop and then modified to look like they originated on Internet of Things devices [28]. The IoTID20 dataset converted these packet files into CSV files the CICFlowMeter is used. The CSV data sets were categorised based on the IP addresses in accordance with the types of attacks and their unusual behaviour. The dataset's distribution is displayed in **Tables 1** and **Table 2** There are 86 features in the dataset.

Table 1. Attacked And Normal Instances in The Iotid20 Dataset

Binary Label Distribution	Instances
Normal 40,073	Normal 40,073
Anomaly 585,710	Anomaly 585,710

Table 2. Distribution Of Anomalous Data in Iotid20 Dataset

Category Label Distribution	Instances
Dos	59,391
Mirai	415,677
Scan	75,265
Normal	40,073
MITM	35,377

Preprocessing

Cleaning of Missing and Anomalous Data

As a crucial part of our data analysis process, data cleaning addresses erroneous, missing, or inaccurate data entries in order to improve the dataset's accuracy and dependability. This section is crucial because it establishes the validity of the analysis and modelling that follow [29].

There were multiple steps in the data cleaning process. First, we used an automated script to find and mark any values that were missing from the dataset. The anticipated data transmission intervals served as the basis for the identification criteria for missing data. Any points that did not follow these time intervals were regarded as absent.

After determining which data were missing, we filled in these gaps using an imputation technique. When dealing with time series data, we specifically applied a linear interpolation method that relies on the assumption that Two data points differ by a linear difference that can be estimated. This approach was selected due to its ease of use and efficiency in handling minor gaps in time series data.

It carefully examined the dataset to look for anomalies that might point to erroneous or missing data combined with erroneous data. We employed a Z-score analysis to identify any outliers in this. When a Z-score was higher than three, a data point was deemed extreme, and it was manually examined to see if it represented true values or anomalies brought on by mistakes in data transmission or recording.

By looking at the daily output curves' consistency, we were able to further refine the dataset. Curves that consistently displayed anomalies and markedly deviated from the established pattern without a valid reason were eliminated. This choice was made using the gradient a cutoff point that was established the mean gradient of the dataset divided by two standard deviations and the curve's standard deviation.

By making improvements to our methodology, we have been able to address potential issues with the process of gathering secondary data from power systems, enhancing the quality of our data, and offering an open and exacting framework for our analysis.

Data Denoising and Smoothing

The Extended Kalman Filter (EKF) algorithm removes noise from data and filters it to improve data accuracy. The filtered data is then smoothed producing data for additional research by utilising a smooth function. One nonlinear state estimation algorithm that can be used for the EKF is a system with nonlinear models, which is based on the Kalman Filter [29]. EKF uses the Kalman Filter technique for error correction and state estimation after linearizing nonlinear problems via the nonlinear functions' Taylor expansion.

The steps that are specific are as follows:

State Equation:

$$x(k) = f(x(k-1), u(k-1)) + \omega(k-1) \tag{1}$$

wherein $\omega(k-1)$ symbolises noise in the process.

Observation Equation:

$$y(k) = h(x(k)) + v(k) \tag{2}$$

wherein $v(k)$ *symbolises* noise from observations.
Observation Steps:
State Vector Prediction:

$$x_{\text{pred}} = [x_1 + x_2, x_2]^T \tag{3}$$

Transition Matrix:

$$F = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \tag{4}$$

Predicted Value of The Covariance Matrix:

$$P_{\text{pred}} = FP_0F^T + Q \tag{5}$$

Update Steps:
Values Observed and Anticipated:

$$y_{\text{yred}} = x_{\text{pred}}\,(1) \tag{6}$$

Observation Matrix:

$$H = [1,0] \tag{7}$$

Kalman Gain:

$$K = P_{\text{pred}}\,H^T\left(HP_{\text{pred}}\,H^T + R\right)^{-1} \tag{8}$$

Estimated State Vector:

$$x_{\text{est}} = x_{\text{pred}} + K\left(y(k) - y_{\text{yred}}\right) \tag{9}$$

Estimated Covariance Matrix:

$$P_{\text{est}} = (I - KH)P_{\text{pred}} \tag{10}$$

In this context, x_1 and x_2 indicate, respectively, the output of wind power and how quickly it changes, which make up the two parts of the state vector. Process noise has a covariance matrix of Q, while observation noise has a covariance matrix of R. The covariance matrices' starting values in the programme are as follows:

$$P_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{11}$$

The technique of moving averages is used to further smooth the data after filtering. A popular method for smoothing one-dimensional vectors and eliminating jitter and noise is the moving average method. Its central notion is as follows: For a vector of length N, a window of length m is selected. A new data point is created by averaging the data within this window. After moving the window one unit to the right and processing every data point, the process is repeated. According to equation:

$$\mathrm{y}_i = \frac{1}{\mathrm{m}} \sum_{j=1}^{m} x_{i-j+\left\lfloor \frac{m}{2} \right\rfloor} \tag{12}$$

where x_i symbolises the initial information vector's point of data i, y_i denotes the window size, m the smoothed data vector's i data point, and $\lfloor \cdot \rfloor$ the floor function.

Data Normalization

Because the installed capacity has a major impact on the wind power output curve, this paper uses the normalisation technique, that splits the daily output of wind power by the daily capacity of wind power installations, to obtain a general rule.

$$x' = \frac{x}{p} \tag{13}$$

Within the formula, x' shows the installed wind power capacity per day and the normalised wind power output value.

Feature Selection Using PCOA

Motivating Factor: The Pine Tree Life Cycle

Coniferous trees, such as pines, can be found in a range of environments. They are members of the Pinaceae, or Pinus family. They can grow to a height of 50 m with cones that are 5–10 cm in diameter, round in scale, and have dark green needles. Male and female flowers are produced on the same pine tree by certain monoecious pines, like the Scotch pine. There are male flowers at the base of tree shoots. In early summer, female flowers are pollinated by the wind. The female flower, which takes more than a year to mature, becomes tiny cones that grow quickly to full size by early summer. The cones then split off and fall to the ground as a result of gravity, or in the spring, animals scatter the seeds by eating them. The seeds eventually grow into saplings, which then yield more trees. Pine trees only get pollination from the wind, unlike other plants; insects have no effect on this process [30].

Mathematical Modelling of Pine Reproduction

A crucial stage in the life cycle of pine trees is cone seed dispersal and pollination. As a result, the current study only takes cone pollination and dispersal into account. Similar to nature, PCOA uses both gravity and animals to disperse the cones. As we proceed with the investigation, we calculate the PCOA mathematically while taking into account the dispersal of cones and pollination brought about by animals and gravity. According to earlier research, pollination, as well as the dispersal of cone seeds by animals and gravity, are critical to pine tree reproduction. There are two ways in which pine tree pollination is similar to figuring out an optimisation problem. In one way, a swarm-based operator can mimic the flow of pollen particles between pine tree cones. Here, the wind is what propels the pollen particles forward. Issues are sought after in the atmosphere of the forest, with pollen particles acting as the agents conducting the search. Every particle in the search space of the optimisation problem has a position and an objective function. Pollen grains travel from one cone of a pine tree to the next in a manner akin to exploring the problem's search space for the best solution. The female pine tree flower's development is another factor and produces tiny cones as a result of pollen particles entering the flower. Evolutionary operators can therefore simulate this process. A problem's optimal solution can be found by improving the current one, much like a pine cone evolves from one state to the next. Pine seedlings are created and grow in new environments as a result of animals and gravity dispersing pine cone seeds. The seedling will grow into a pine tree if its placement is ideal. Similar to this, if a better position is discovered, the answers in the PCOA are updated. The problem search space is represented by the pine tree forest, and the finding agents are the seeds of pine cones. Dispersal of pine cone seeds by gravitation or by animals can be replicated using population-based operators. Therefore, the aforementioned biological principles can be simulated and optimisation problems can be solved by operators based on populations and evolution. In

this way, cone seed, pine tree, and pollen locations act as search agents, and the global optimal problem is determining the ideal location for pine tree growth.

Producing the First Population in PCOA

The pine tree and its cones are found in two populations within the PCOA. The issue domain is split into equal segments. Each segment's centre contains a tree, and several cones are created to encircle each tree during algorithm initialization. For this reason, each part's boundary conditions are provided by

$$LbS = lb + (i-1) \times \frac{ub - lb}{N_{\text{tree}}} \tag{14}$$

$$UbS = ub + (i-1) \times \frac{ub - lb}{N_{\text{tree}}} \tag{15}$$

The problem's domain lower and upper bounds are denoted by lb and ub in Equations (14) and (15), and the lower and upper bounds of each segment are denoted by LbS and UbS. This is how the population of pine cones is created:

$$CX_{j,i} = LbS_i + \overrightarrow{rand}_{1\times\text{dim}} \times (\text{ rand } \times UbS_i - \text{ rand } \times LbS_i) \tag{16}$$

In Equation (16), CX is the pine cone position, $\overrightarrow{rand}_{1\times\text{dim}}$ is the normal distribution of a random number between 0 and 1, and is the normal distribution of a random vector between 0 and 1. i stand for the cone and pine tree indices of the PCOA operators, respectively. After PCOA is initialised, each tree is positioned based on where the best cone is in that section.

Gravity-Based Pine Cone Dispersal (Exploitation)

Pine cones eventually fall from the tree due to gravity as they ripen and become heavier. The problem makes the cones scatter in a small area. To model this process, the search space problem is split up into several super-cubes. These super-cubes each have a tree inside of them. For the purpose of simulating the effects of gravitational forces on cone dispersion near trees, it is essential to generate many solutions pertaining to every tree. The following operator is used in order to accomplish this goal.

$$CX_{j,i}^{\text{new}} = \begin{cases} TX_i + w_1 \times R_1 \times \left(R_2 \times (Ub_s^i - Lb_s^i - TX_i)\right), & \text{if Control}_{\text{parameter}} = 0 \\ CX_{j,i} + w_1 \times R_1 \times \left(R_2 \times \left(Ub_s^i - Lb_s^i - Tpop_{all,r1}\right) - \text{ Tbest }_{x,i}, \right. & \text{otherwise.} \end{cases} \tag{17}$$

In Equation (17), $CX_{j,i}^{\text{new}}$ is the pine cone's new location, $CX_{j,i}$ is the pine cone's updated location, TX_i is where the tree is located, Ub_s^i is the super-cube's upper bound, Lb_s^i is the super-cube's lower bound, Tpopall,r1 is a approximately chosen solution stored in PCOA's memory, Tbest $x_{x,i}$ is one of PCOA's best solutions, w_1 is an adaptive weight, r1 represents a random integer among 1 and the PCOA memory's size, and R_1 and R_2 are typical random numbers in the range of 0 and 1. i denotes the PCOA operators' pine tree index with j cone index. Over generations, the problem's boundaries (lb and ub), as well as LbS and UbS, get smaller. The research space is constrained in applying this technique to the closing region with the ideal surrounding location across generations. Using this method, between exploration and exploitation, the PCOA can achieve a balance. The following equations are defined in order to achieve this:

$$LbS = Lb + (i-1) \times \frac{Ub - Lb}{N_{\text{tree}}} \tag{18}$$

$$UbS = Ub + (i-1) \times \frac{Ub - Lb}{N_{\text{tree}}} \tag{19}$$

$$Lb = lb + \text{ Radius }_{lb} \times W \tag{20}$$

$$Ub = ub - \text{ Radius }_{ub} \times W \tag{21}$$

$$W = \min FES/FES_{\max}, 0.5 \tag{22}$$

$$\text{Radius }_{lb} = X_{\text{best}} - lb \tag{23}$$

$$\text{Radius}_{ub} = ub - X_{\text{best}} \tag{24}$$

The algorithm updates the problem's boundaries, Lb and Ub, and calculates the shrinking radius with Equations (18)-(24) from Radius $_{lb}$ and Radius $_{ub}$. From 1 to 0.5, the adaptive weight W decreases. X_{best} symbolises the ideal position as of right now. The current generation is represented by FES, with $FES_{\max}$ signifying the highest number of generations.

Pollination (Exploration)

The agents that pollinate pine trees are produced by the male cones and move towards the female cones. This movement can take place within the same tree or between the cones of different trees. Given the complexity of this movement, various mechanisms for simulating it can be defined. Two distinct approaches are defined in PCOA to emulate this motion. Algorithm 1 is used in the initial method to simulate wind pollination. As per the research conducted by, wind pollination is simulated in the second method. This is how this method is put together:

$$CX_{j,i}^{\text{new}} = CX_{j,i} + 0.5 \times \phi_{r_1} \times \left(\text{Tbest}_j - CX_{r_1,i} \right) + 0.5 \times \phi_{r_3} \times \left(\text{Tbest}_j - CX_{r3,i} \right) \tag{25}$$

∅ represents the probability of successfully pollinating pine cones in Equation (25). ∅ is computed in this way:

$$\phi_i = 1 - e^{-\gamma * \sum_{j=1}^{n} a_{i,j}} \tag{26}$$

$$a_{i,j} = \frac{\frac{\beta}{d_{i,j}^{\alpha}} - \frac{\alpha}{d_{i,j}^{\beta}}}{\beta - \alpha} \tag{27}$$

In Equations (26) and (27), $a_{i,j}$ The influence of the ith cone on the j th cone is denoted by $d_{i,j}$, and the separation in Euclides among the ith and jth cones. Furthermore, three constant parameters are α, β, and γ that provide a mathematical definition of the pollination process. The study conducted by states that the appropriate values for α, β, and γ are 3,40, and 0.62, in that order.

Algorithm 1 Pollination

$Input$: $Pollination\ cycle\ (PN), cones\ position\ (CX), cones\ fitness\ (CF)$
$Output$: $Pine\ Cone's\ position$
$Wind's\ pollination$:
1: $while\ N < PN\ do$
2: $N \leftarrow N + 1$
3: $MemoryIndex, rand \leftarrow \lfloor dim * \longrightarrow rnd \rfloor$
4: $\mu W1 = MemoryW1(MemoryIndex, rand)$
5: $\mu W2 = MemoryW2(MemoryIndex, rand)$
6: $\mu W3 = MemoryW3(MemoryIndex, rand)$
7: $\mu cr = Memorycr(MemoryIndex, rand)$
8: $W1 \leftarrow \mu W1 + \longrightarrow Rnd1 \times tan\,(\pi \times (\longrightarrow Rnd2 - 0.5))$
9: $W2 \leftarrow \mu W2 + \longrightarrow Rnd3 \times tan\,(\pi \times (\longrightarrow Rnd4 - 0.5))$
10: $W3 \leftarrow \mu W3 + 0.1 \times tan\,(\pi \times (\longrightarrow Rnd5 - 0.5))$
11: $cr \leftarrow \mu cr + 0.1 \times \longrightarrow Rnd6$
12: $if\ W1 < 0\ then$
13: $W1 \leftarrow \mu W1 + 0.1 \times tan\,(\pi \times (\longrightarrow Rnd7 - 0.5))$
14: $end\ if$
15: $if\ W2 < 0\ then$
16: $W2 \leftarrow \mu W2 + 0.1 \times tan\,(\pi \times (\longrightarrow Rnd8 - 0.5))$
17: $end\ if$
18: $if\ W3 < 0\ then$
19: $W3 \leftarrow \mu W2 + 0.1 \times tan\,(\pi \times (\longrightarrow Rnd9 - 0.5))$
20: $end\ if$
21: $W1 \leftarrow min\,(W1, 1), W2 \leftarrow min\,(W2, 1), W3 \leftarrow min(W3, 1), cr \leftarrow min\,(cr, 1), cr \leftarrow max(cr, 0)$
22: $Generate\ three\ random\ integer\ number\ (r1, r2, r3)$
23: $Choose\ Tbestrate\%\ of\ best\ solutions\ and\ save\ them\ on\ Tbest;$

24: $Choose\ Tworstrate\%\ of\ worst\ solutions\ and\ save\ them\ on\ Tworst;$
25: $if\ rand\ <\ 0.5\ then$
26: $CXnew\ \leftarrow\ XC\ +\ W1\ \times\ (Tbest\ -\ CXr1)\ +\ W2\ \times\ (CXr1\ -\ Tpopall, r2)$
27: $else$
28: $W\ =\ max(W1\ \times\ W2, (1\ -\ W1)\ \times\ W2, 1\ -\ W2)$
29: $CXnew\ \leftarrow\ W2 \times\ (W1 \times\ CX\ +\ (1 - W1)\ \times\ Tbest)\ +\ (1 - W2)\ \times\ CXr1\ + W\ \times\ (CXr3\ - Tpopall, r2))$
30: $end\ if$
31: $Evaluate\ f\ itness\ f\ unction$
32: $if\ CFnew\ <\ CF\ ||\ rand\ <\ cr\ then$
33: $XC\ \leftarrow\ XCnew, XF\ \leftarrow\ XFnew$
34: $end\ if$
35: $Update\ Tpopall\ (Add\ new\ solution\ to\ archive\ and\ remove$
$duplicate\ or\ randomly\ remove\ some\ solutions\ to\ maintain\ the\ archive\ size);$
36: $Update\ \mu W1, \mu W2, \mu cr$
37: $end\ while$
38: $return\ Pine\ Cone's\ position$

Animals' Spread of Pine Cones (Exploitation)

Pine cones are eaten by animals and then released into the surroundings. For instance, squirrels burrow pine cones in the autumn so they can eat them later. Pine cones are consumed by birds and other creatures such as deer, bears, rats, and mice. To replicate these behaviours, four operators are defined in the PCOA. According to the first operator, pine cones are carried by animals such as squirrels from an arbitrary location to a location that is in better condition for burial. To determine the ideal location, this operator uses the optimisation technique of quadratic programming. This method defines the initial point as follows:

$$X_{\text{initial}} = X_{\text{best}} + \text{rand} \times (\overline{CX} - X_{\text{best}}) \tag{28}$$

In Equation (28), X_{initial} is quadratic programming's starting position, and $\overline{CX}$ is the mean of all the locations of the pine cones. It should be mentioned that the problem's updated boundary conditions are used when conducting the quadratic programmatic search. The PCOA can more accurately approximate the global optimum with this method. The second through fourth operators are motivated by the roles that animals play in the dispersal of pine cones. The following is the design of the second operator:

$$X_{\text{animal}} = \frac{X_{\text{best}} + X_{\text{animal}}}{2} + \text{levy} \times \left(\text{levy} \times \left(lb + ub - \frac{X_{\text{best}} + X_{\text{animal}}}{2} \right) - \frac{X_{\text{best}} + X_{\text{animal}}}{2} \right) \tag{29}$$

In Equation (29), X_{animal} is the location of the pine cones that animals are carrying; The Levy distribution yielded this Levy random number. The third operator makes the idea that animals move pine cones from where they were originally found to a location close to a tree. The third operator updates the pine cone's position in this manner:

$$X_{\text{animal}} = CX + (1 - w_d) \times \overline{TX} + w_d \times \text{levy} \times (\text{levy} \times (lb + ub - \overline{TX}) - \overline{\text{Tree}_x}) \tag{30}$$

In Equation (30), $\overline{TX}$ represents the pine trees' typical position, w_d is an adaptive weight that provides:

$$w_d = \exp - 20 \times \frac{FES}{FES_{\max}} \tag{31}$$

w_d restricts the search space to an increasing amount of function assessments and results in the conversion of discovery into utilisation. The fourth operator thinks an animal that is getting closer is carrying a pine cone:

$$X_{\text{animal}} = CX + w_d \times \text{levy} \times (\text{levy} \times (lb + ub - CX) - CX) \tag{32}$$

Under PCOA, each of the four designated operators is accountable for exploitation. Algorithm 2 shows how PCOA animals distribute pine cones.

Algorithm 2 Animals scattering pine cones

$Input: P1, P2, CX, TX, FES\ and\ FESmax$
$Output: Pine\ Cone's\ position$
$Dispersing\ pine\ cone\ by\ animals:$
1: $if\ and(FES > P2 \times FESmax, rand < 0.9) \,||\, and(FES < P1 \times FESmax, rand < 0.9)\ then$
2: $Calculate\ Xinitial\ using\ Equation\ (25)$
3: $Run\ quadratic\ programming\ optimization\ method\ based\ on\ the\ Xinitial$
4: $Update\ Xbest\ if\ solution\ of\ quadratic\ programming\ was\ better$
5: $Calculate\ Xanimal\ using\ Equation\ (26)$
6: $else$
7: $Compute\ wd\ using\ Equation\ (28)$
8: $if\ rand < 0.5\ then$
9: $Calculate\ Xanimals\ using\ Equation\ (27)$
10: $else$
11: $Calculate\ Xanimals\ using\ Equation\ (29)$
12: $end\ if$
13: $end\ if$
14: $return\ Pine\ Cone's\ position$

CNN-BiGRU-AAM Classification

Convolutional Neural Networks

Textual or 1-dimensional time series data are frequently processed using one-dimensional CNNs because of their superior ability to extract features from short, fixed-length inputs over the course of a dataset [31]. An entirely new feature representation is created using the input sequence is first scanned in a sliding manner using a convolution kernel (or filter), followed by convolutional operations to extract prominent features from the sequence. At each location, the convolution operation's output ξ is determined for the convolution kernel W and the input sequence X:

$$\xi[j] = (X \cdot W)[j] = f\left(\sum_{k=0}^{K-1} X[j \times T_s + k] \cdot W[k] + b_{cnn}\right) \tag{33}$$

where $\xi[j]$ represents the output sequence's jth element as a result of the convolution operation. A section of the sequence that was entered (from j to $j + K - 1$) is multiplied element-by-element by the convolution kernel, and the result is the j th element of the output sequence (Equation 33). $X[j + k]$ is the $j + k$ th part of the input sequence, where k represents the offset that the current position of the convolution operation is represented by j, and the convolution kernel slides. T_s is the time step, as well as W[k] is the k weight of the convolution kernel. K is the convolution kernel's size, or window size; it multiplies every single component by element using a subset of the input sequence; b_{cnn} is a term of bias, and $f(\cdot)$ is a nonlinear properties-introducing activation function. PReLU (Parametric ReLU), ReLU (Rectified Linear Unit), and other popular activation functions are examples.

Subsequently, the size of the feature map is minimised via a pooling operation, which simultaneously extracts the most important features and lowers the model's computational complexity. The following is the calculation formula:

$$x[j] = \max(X[j \times T_s : (j + 1) \times T_s]) \tag{34}$$

where $x[j]$ represents the first j elements of the feature sequence that was obtained by pooling the data. Equation (34), $X[j \times T_s : (j + 1) \times T_s]$ shows that a subsequence has been chosen in the initial sequence, and its start index is $j \times T_s$ and the final index is $(j + 1) \times T_s$. Such indexing operations are usually used to intercept a specific window or sequence segment.

Bidirectional Gated Recurrent Unit Neural Network

Important historical features are preserved by both GRU and LSTM through the use of "gate" structures. In LSTM, forgetting and input gates are combined to create an update gate that consists solely of a reset gate (r) and an update gate (z). GRU performs direct output computation and lacks a kernel state.

x_t is the GRU module's current moment input; h_{t-1} is the state of the preceding moment; r_t is the reset gate; its value can be determined by applying Equation (35) to the calculation the candidate state's degree of dependence $\tilde{h}_t$ on h_{t-1}; z_t is

obtained from Equation (36), which is the update gate; and h_t is the potential output value following the processing of the reset gate. Associating Equations (35)-(38) will yield h_t.

$$r_t = \text{sigmoid}\ (U_r h_{t-1} + W_r x_t + b_r) \tag{35}$$
$$z_t = \text{sigmoid}\ (U_z h_{t-1} + W_z x_t + b_z) \tag{36}$$
$$\tilde{h}_t = \tanh\ (U_c(r_t h_{t-1}) + W_c x_t + b_c) \tag{37}$$
$$h_t = z_t h_{t-1} + (1 - z_t)\tilde{h}_t \tag{38}$$

In Equation (35), U_r and W_r are the weight matrices that correspond to the reset gate's input for the t moment and its output for the t-1 moment, respectively, and b_r is the reset gate's bias; as shown in Equation (36), U_z and W_z are the weight matrices that, in turn, represent the update gate's input for the t moment and output for the t-1 moment, and b_z indicates the update gate's bias; additionally, in Equation (37), b_c is the concealed bias of the candidate, and U_c and W_c are the matrices of weights for the candidate hidden state's input (t moment) and output (t-1 moment), respectively.
It is evident from equations (35)-(38) that the sigmoid function uses the given expression to control the update and reset gates' range of values $\delta(x) = 1/(1 + e^{-x})$. The reset gate regulates the mixture of x_t and h_{t-1},and the proportion of decreases as it moves closer to 0 at h_{t-1}. The gate for updates z_t confirms the dimensions of h_{t-1} the more it approaches 1, The more data from the prior instant is utilised in the present.

The unidirectional GRU structure can gather historical data prior to a specific date, but it is unable to gather data related to that period of time and beyond. The forward and backward outputs of two GRU layers are spliced together by BiGRU, and the precise computational procedure is represented by Equations (39)-(41). By merging past and future data, the BiGRU can more effectively obtain information about associations and sequences within the data, improving the model's performance ability when compared to the single-way GRU. Additionally, it can successfully address the flaw of requiring the acquisition of all relevant relationship data.

$$\vec{h}_t = \text{GRU}_f\ (x_t, \vec{h}_{t-1}) \tag{39}$$
$$\overleftarrow{h}_t = \text{GRU}_b\ (x_t, \overleftarrow{h}_{t-1}) \tag{40}$$
$$\hat{h}_t = \vec{h}_t \oplus \overleftarrow{h}_t \tag{41}$$

where $\vec{h}_t$ and $\overleftarrow{h}_t$ represent, respectively, forward and backward GRU passes; GRU U_f and GRU_b indicate forward and backward GRU function tandem combinations, respectively; the vector splice operation is represented by $\oplus$.

Additive Attention Mechanism

Among the attention mechanisms is the additive attention mechanism (AAM). Originally used in sentence translation, the additive attention mechanism also referred to as multilayer perceptron attention. More and more, however, image processing has been using this technique in recent years. It gives the model the flexibility to give various input sequence segments distinct weights, which makes it possible to analyse sequence data in a targeted manner. Additionally, it uses a sigmoid function to make the transformed features active following the input features' linear transformation to determine how similar the two features are, which is useful for handling nonlinear relationships. Additive attention is used to decode the data following BiGRU's encryption and CNN's processing of the time series features. Next, the output is approved for use as input. The additive attention formula is as follows:

$$y_i = \text{DecoderOutput}\ (s_t, c_t) \tag{42}$$

where y_i represents the output of the decoder at time step i. Equation (42) shows that s_t represents the additive attention decoder's hidden vector at time step t, and y_i is the outcome of the attention decoder additive. For instance, according to equation (43), the decoder obtains c_t at time step t from the encoding t outcome of the encoder $(h_0, h_1, \dots, h_t)$, which is the encoder's weighted sum of its encoding output.

$$c_t = \sum_{i=1}^{T} a_{t,i} h_i \tag{43}$$

The weights are determined in the manner described below:

$$a_{ti} = \frac{\exp\left(e_{t,i}\right)}{\sum_{j=1}^{T} \exp\left(e_{t,j}\right)} \tag{44}$$

where the attentional weight a_{ti} represents the attentional weight of the decoder The attention score, normalised by softmax, is the source of the ith time step from the t time step. This is how the attention score is determined:

$$e_{t,i} = v_a^T \tanh\left(W_a h_i + U_a s_{t-1}\right) \tag{45}$$

In additive attention, the hidden states of the encoder and decoder are transformed linearly, and then a hyperbolic tangent function is applied to determine the attention score. Equation (45) shows $e_{t,i}$ is a time step t attention score that is sent from the decoder to the encoder; At this point, the decoding result is dependent on the previous time step's latent state of the decoder, s_{t-1}, at the i-th temporal step, the encoder output $h_i \cdot v_a$ is the vector of learned weight, and W_a and U_a are the learned parameter matrices of the model.

Hyper Parameter Tuning Using BOA

This section explains the Botox Optimisation Algorithm (BOA), starting with a discussion of its theory and inspiration for hyper parameter tuning of CNN-BiGRU-AAM model. After that, a detailed mathematical modelling of the steps involved in implementing the suggested BOA approach is presented [32].

Inspiration of BOA

For many people, improving the appearance of their faces is a serious and complex concern, and the appearance of wrinkles on the face frequently causes distress. Wrinkles are caused by a combination of dermal atrophy and repeated contraction of the facial muscles beneath. This issue is addressed by carefully tiny amounts of injections of botulinum toxin into specific overactive muscles. The injection relaxes the muscles locally, which in turn causes the skin in these areas of hyperactive muscles to smooth out. For this, the powerful neurotoxin protein known as botulinum toxin, which is derived from the Clostridium botulinum bacterium and used. When this toxin is administered, the targeted muscles become momentarily paralysed, which stops wrinkles from forming in the treated area. The FDA authorised the cosmetic use of botulinum toxin, or Botox, in 2002 to treat frown lines caused by the glabellar complex muscles and in 2013 to treat crow's feet caused by the lateral orbicularis oculi muscles. Botox has a notable effect on improving the appearance of wrinkles on the face. The intelligent process of strategically injecting Botox into targeted facial areas to remove wrinkles forms the foundation for the BOA's recommended method's design.

Algorithm Initialization

Through an iterative process, the suggested BOA methodology uses the participants' collective search capabilities to operate as a population-based optimizer to produce workable solutions for optimisation problems. The BOA population in this instance is made up of people looking to get Botox injections. Each member adds to the decision variable values according to where they are in the mathematically embodied problem-solving space, and that is represented as a vector. Equation (46), which describes the population matrix, is formed by this vector, which contains the decision variables. Equation (47) is used to randomly assign each BOA member's position to begin with.

$$\boldsymbol{X} = \begin{bmatrix} \vec{X}_1 \\ \vdots \\ \vec{X}_i \\ \vdots \\ \vec{X}_N \end{bmatrix}_{N\times m} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ x_{i,1} & \cdots & x_{i,d} & \cdots & x_{i,m} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,d} & \cdots & x_{N,m} \end{bmatrix}_{N\times m}, \tag{46}$$

$$x_{i,d} = lb_d + r_{i,d} \cdot (ub_d - lb_d), i = 1, \dots, N, d = 1, \dots, m, \tag{47}$$

wherein X the population matrix of BOA, $\vec{X}_i$ is the potential remedy's (I)th BOA member, $x_{i,d}$ is its d th N is the population's size, and the number of decision variables is denoted by m in the search space (decision variable), $r_{i,d}$ are arbitrary values drawn from the range [0,1], and lb_d and ub_d are the d th decision variable's lower and upper bounds, respectively.

It is feasible to assess the problem's corresponding objective function for each individual in the BOA population since each one of them has the potential to be a solution. As a result, the range of values for One vector representation of the objective function is to use Equation (48):

$$\vec{F} = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N\times 1} = \begin{bmatrix} F(\vec{X}_1) \\ \vdots \\ F(\vec{X}_i) \\ \vdots \\ F(\vec{X}_N) \end{bmatrix}_{N\times 1} \tag{48}$$

whereas $\vec{F}$ is the evaluated objective function's vector, and F_i is the objective function that has been evaluated using the ith BOA member.

The evaluated values of the objective function provide trustworthy standards by which to measure the calibre of potential solutions. Consequently, the ideal value found for the goal function is represented by the ideal BOA participant, whereas the worst value is represented by the suboptimal member. Because every iteration changes the best candidate solution and the locations of BOA population members are updated regularly, as are their objective function values.

Mathematical Modelling of BOA

Through an iterative process, the population-based optimizer known as BOA approach skillfully provides workable solutions for optimisation problems. The Botox injection mechanism serves as an inspiration for the BOA's design, which modifies the population's locations within the search area. The simulation and schematic for Botox injections utilised to develop the recommended BOA technique.

Every person requesting One example of a BOA community member is Botox injections. How a physician administers Botox injections to specific face muscles in order to minimise wrinkles and improve appearance is mirrored in the design of the BOA. Like Botox, this is how the BOA approach enhances a candidate solution: it picks some decision variables and adds a designated value.

The quantity of face muscles needed Botox injections is thought to decrease as the algorithm iterates, according to the design of the BOA. Equation (49) is therefore used to determine the quantity of selected muscles (i.e., variables used in decision-making) for Botox injection:

$$N_b = \left\lfloor 1 + \frac{m}{t} \right\rfloor \leq m, \tag{49}$$

where N_b is the number of muscles that require injections of Botox, and t is the iteration counter's present value.

Upon seeing the candidate, the physician selects which facial muscles and wrinkles to inject Botox into. Motivated by this fact, In BOA design, equation (50) is utilised to determine which variables should be injected for every individual in the population. It is important to remember that the muscles selected for Botox injections shouldn't be used again, as Equation (50) indicates:

$$CBS_i = \{d_1, d_2, \dots, d_j, \dots, d_{N_b}\}, d_j \in \{1,2,\dots,m\} \text{ and } \forall h,k \in \{1,2,\dots,N_b\}: d_h \neq d_k. \tag{50}$$

Thus, CBS_i is the collection of potential decision factors for the ith population member chosen to receive a Botox injection; the position selected for the Botox injection is indicated by the decision variable d_j.

According to Equation (51) how much Botox each member will receive of the population is calculated in the BOA design, which is comparable to a physician's discretion in determining the appropriate dosage based on experience and patient requirements:

$$\vec{B}_i = \begin{cases} \vec{X}_{\text{mean}} - \vec{X}_i, t < \dfrac{T}{2} \\ \vec{X}_{\text{best}} - \vec{X}_i, \text{ else} \end{cases} \tag{51}$$

where $\vec{B}_i = (b_{i,1}, \dots, b_{i,j}, \dots, b_{i,m})$ is the sum that is taken into account for the ith member's Botox injection, $\vec{X}_{\text{mean}}$ is the average population position (i.e., $\vec{X}_{\text{mean}} = \frac{1}{N}\sum_{i=1}^{N}\vec{X}_i$), T is the sum of all the iterations, and $\vec{X}_{\text{best}}$ is the most ideal member of the population.

The face's appearance shifts, and wrinkles vanish, after getting a Botox injection made into the facial muscles. When the objective function value is raised, equation (52) replicates the injection of Botox into the face muscles, the new positions of each BOA member are first established in the BOA design, the corresponding member's prior position has been replaced by this new one in accordance with Equation (53):

$$\vec{X}_i^{\text{new}} : x_{i,d_j}^{\text{new}} = x_{i,d_j} + r_{i,d_j} \cdot b_{i,d_j} \tag{52}$$

$$\vec{X}_i = \begin{cases} \vec{X}_i^{\text{new}}, F_i^{\text{new}} < F_i \\ \vec{X}_i, \text{ else,} \end{cases} \tag{53}$$

where $\vec{X}_i^{\text{new}}$ is the Ith BOA member's new role following a Botox injection, x_{i,d_j}^{new} is its d_j th dimension, F_i^{new} is the value of its objective function, r_{i,d_j} is a variable that is random on the interval [0,1] with a uniform distribution, b_{i,d_j} is the d_j th the Botox injection size of the ith BOA member(i.e., $\vec{B}_i$).

Pseudocode, of the BOA Flowchart

The algorithm's first iteration ends when every BOA member's position in the search space has been updated. Equations (49)-(53) are used to update the members of the BOA population until the last iteration, at which time the algorithm advances to the following iteration using the updated values. The best-performing candidate solution found after each iteration $\vec{X}_{\text{best}}$ is saved and updated as well. Following the recommended BOA approach's complete implementation, the top contender solution $\vec{X}_{\text{best}}$ The solution to the given problem is introduced as stored during the algorithm's iterations. The steps involved in implementing BOA are displayed in Algorithm 3 as pseudocode.

Algorithm 3. Pseudocode of the BOA.

$Start\ the\ BOA.$

1. $Input\ problem\ information: variables, objective\ function, and\ constraints.$
2. $Set\ the\ BOA\ population\ size\ N\ and\ the\ total\ number\ of\ iterations\ T.$
3. $Generate\ the\ initial\ population\ matrix\ at\ random\ using\ Equation\ (47).$
4. $Evaluate\ the\ objective\ function.$
5. $Determine\ the\ best\ candidate\ solution\ \rightarrow\ Xbest.$
6. $For\ t\ =\ 1\ to\ T$
7. $Update\ number\ of\ decision\ variables\ for\ Botox\ injections\ using\ Equation\ (49).$
8. $For\ i\ =\ 1\ to\ N$
9. $Determine\ the\ variables\ that\ are\ considered\ for\ Botox\ injection\ using\ Equation\ (50).$
10. $Calculate\ the\ amount\ of\ Botox\ injection\ using\ Equation\ (51).$
11. $For\ j\ =\ 1\ to\ Nb$
12. $Calculate\ the\ new\ position\ of\ the\ ith\ BOA\ member\ using\ Equation\ (52).$
13. End
14. $Evaluate\ the\ objective\ function\ based\ on\ \rightarrow X\ new\ i.$
15. $Update\ the\ ith\ BOA\ member\ using\ Equation\ (53).$
16. End
17. $Save\ the\ best\ candidate\ solution\ obtained\ so\ far.$
18. End
19. $Output\ the\ best\ quasi-optimal\ solution\ obtained\ with\ the\ BOA.$

$End\ the\ BOA.$

IV. RESULTS AND DISCUSSIONS

Experimental Setup

Different software and hardware setups were used to conduct the suggested study. The specifications for the proposed system are listed in **Table 3.** It was determined that these specifications worked well for teaching huge amounts of data.

Table 3. System Configuration

Hardware	Environment
Operation Scheme	Windows-10
CPU	17
Memory	8
Development situation	Python 3.6

Performance Metrics

The four data prediction situations in intrusion detection are known as "TP," "FN," "TN," and "FP," with the following definitions for each

1) $TP\ (True\ Positive)$: $Both\ the\ actual\ label\ and\ the\ predicted\ label\ are\ positive;$

2) FN $(False\ Negative)$: $The\ actual\ label\ is\ positive, while\ the\ predicted\ label\ is\ negative;$
3) TN $(True\ Negative)$: $Both\ the\ actual\ label\ and\ the\ predicted\ label\ are\ negative;$
4) FP $(False\ Positive)$: $The\ actual\ label\ is\ negative, while\ the\ predicted\ label\ is\ positive;$

The confusion matrix collected from the above four situations is exposed in **Table 4.**

Following the above description, to evaluate IDS performance, three metrics are frequently employed: detection rate (DR)

Table 4. Confusion Matrix

Actuals	Positive	Negative	Total
Positive	TP	FN	P (Actual)
Negative	FP	TN	N (Actual)
Total	P (Predicted)	N (Predicted)	P+N

As illustrated in (54), coverage is a measurement of the classifier's capacity to recognize positive examples, and detection rate (DR) is a measure of coverage.

$$DR = \frac{TP}{TP+FN} \tag{54}$$

To evaluate how well a model does at classifying samples, you may look at its accuracy, distinct as the proportion of precise predictions compared to the total trials, as shown in (55).

$$Accuracy = \frac{TP+TN}{TP+FN+TN+FP} \tag{55}$$

The percentage of all negative events is accounted for by the false alarm rate (FAR), which is defined as (56).

$$FAR = \frac{FP}{TN+FP} \tag{56}$$

Types of Attacks Classification Validation

Table 5 summarizes the analysis of different types of attacks in IoTID20 Dataset

Table 5. Different Types of Attacks Analysis

Category	Accuracy	Precision	Recall	F-score	DR	FAR
Mirai	98.67	99.08	98.61	98.80	98.71	98.66
Scan	99.09	98.53	98.14	98.33	98.89	98.06
Normal	98.96	98.00	99.00	99.00	98.04	98.98
Dos	98.30	98.32	98.51	98.41	98.29	98.59
MITM	99.14	99.08	99.11	98.63	98.19	98.91

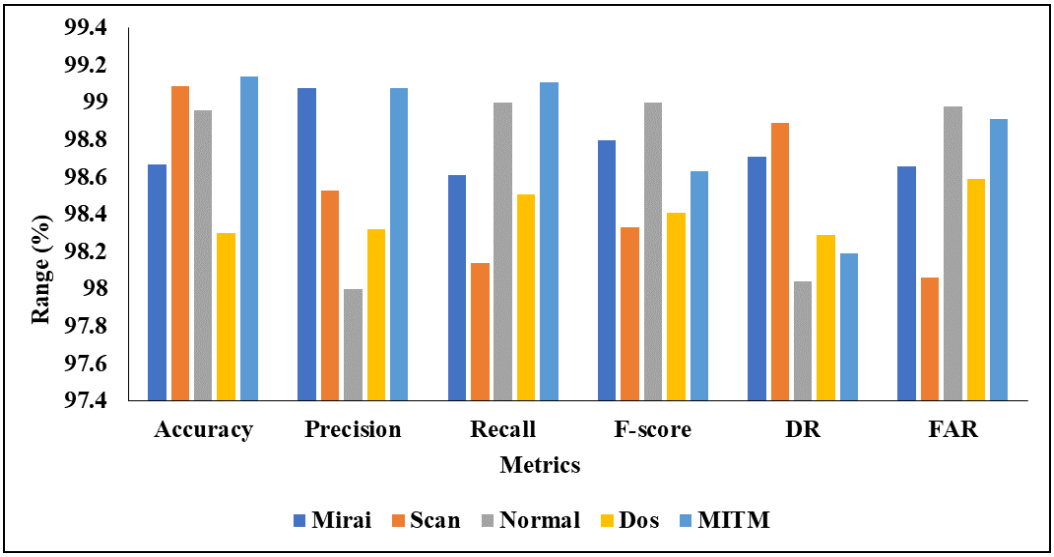

Fig 1. Graphical Analysis of Types of Attacks Classification.

Table 5 and **Fig 1** presents an examination of various attack kinds, such as MITM (Man-in-the-Middle), Mirai, Scan, Normal, and DoS. The table showcases various performance metrics such as accuracy, F-score, detection rate (DR), precision, recall, and false alarm rate (FAR) for each attack category. Mirai exhibits high accuracy (98.67%) and precision (99.08%), with strong recall (98.61%) and F-score (98.80%). Similarly, Scan shows excellent performance with accuracy (99.09%), precision (98.53%), recall (98.14%), and F-score (98.33%). Normal traffic also demonstrates strong metrics across the board, indicating a well-balanced classification. DoS and MITM attacks display competitive performance, with accuracy ranging from 98.30% to 99.14% and precision, recall, and F-score values hovering around the high nineties. These results underscore the classification model's ability to reliably identify and differentiate between various types of network attacks.

Classification Validation of Various Models

Table 6 presents the classification analysis of the suggested CNN-BiGRU-AAM model compared to other existing models.

Table 6. Classification Analysis of Suggested Model with Other Existing Models

Technique	Accuracy	Precision	Recall	F-score	DR	FAR
LSTM	88.32	87.97	87.40	87.69	89.13	76.37
Bi-LSTM	92.73	89.65	88.32	89.92	91.58	78.81
GRU	95.52	90.47	89.43	90.44	90.03	77.18
Bi-GRU	96.97	92.00	92.00	93.00	94.00	78.00
CNN	98.34	93.44	94.46	95.45	96.33	73.66
Proposed CNN-BiGRU-AAM model	99.18	96.71	97.92	97.30	98.61	72.61

Table 6 and **Fig 2** illustrates the classification analysis comparing the proposed CNN-BiGRU-AAM model with other existing models. The LSTM model attained an ACC of 88.32%, precision of 87.97%, recall of 87.40%, F-score of 87.69%, detection rate (DR) of 89.13%, and false alarm rate (FAR) of 76.37%. Bi-LSTM exhibited improved performance with an accuracy of 92.73%, precision of 89.65%, recall of 88.32%, F-score of 89.92%, DR of 91.58%, and FAR of 78.81%. The GRU model attained an accuracy of 95.52%, precision of 90.47%, recall of 89.43%, F-score of 90.44%, DR of 90.03%, and FAR of 77.18%. Bi-GRU further enhanced the results with an accuracy of 96.97%, precision of 92.00%, recall of 92.00%, F-score of 93.00%, DR of 94.00%, and FAR of 78.00%. The CNN model achieved impressive performance with an accuracy of 98.34%, precision of 93.44%, recall of 94.46%, F-score of 95.45%, DR of 96.33%, and FAR of 73.66%. Finally, the proposed CNN-BiGRU-AAM model outperformed all other models, achieving an accuracy of 99.18%, precision of 96.71%, recall of 97.92%, F-score of 97.30%, DR of 98.61%, and FAR of 72.61%. These findings show that, in comparison to other models, the recommended model performs better in correctly classifying the provided data.

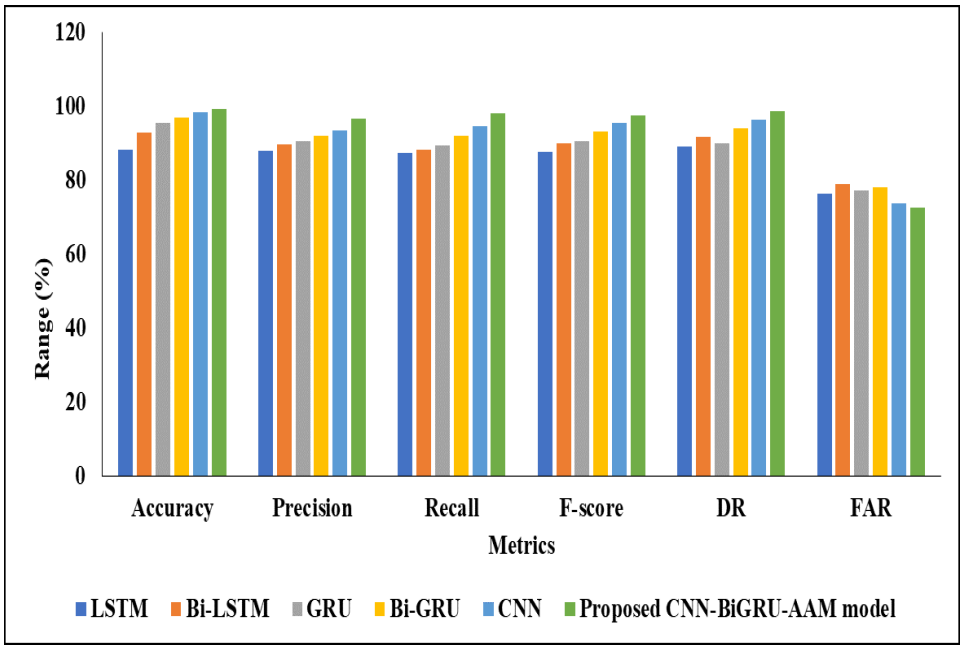

Fig 2. Graphical Analysis of Various Classification Models.

Feature Selection validation

Table 7 compares the analysis accuracy results both with and without the choice of PCOA features.

Table 7. Accuracy Analysis Both with And Without the Choice of PCOA Features

Technique	Without Feature Selection	With PCOA
LSTM	92.3	95.9
Bi-LSTM	93.7	96.8
GRU	94.8	97.3
Bi-GRU	95.5	97.9
CNN	96.8	98.2
Proposed CNN-BiGRU-AAM model	97.6	99.2

Table 7 and **Fig 3** presents an accuracy analysis comparing the performance of various techniques with and without PCOA feature selection. Without feature selection, the LSTM model achieves an ACC of 92.3%, which increases to 95.9% when PCOA is applied. Similarly, the Bi-LSTM model demonstrates an accuracy improvement from 93.7% to 96.8% with PCOA. The GRU model's accuracy rises from 94.8% to 97.3% with PCOA, while the Bi-GRU model sees an increase from 95.5% to 97.9%. For the CNN model, the accuracy improves from 96.8% to 98.2% with PCOA. Notably, the proposed CNN-BiGRU-AAM model achieves an accuracy enhancement from 97.6% to 99.4% with the application of PCOA feature selection. These results highlight the efficiency of PCOA in raising the precision of various techniques in the classification task.

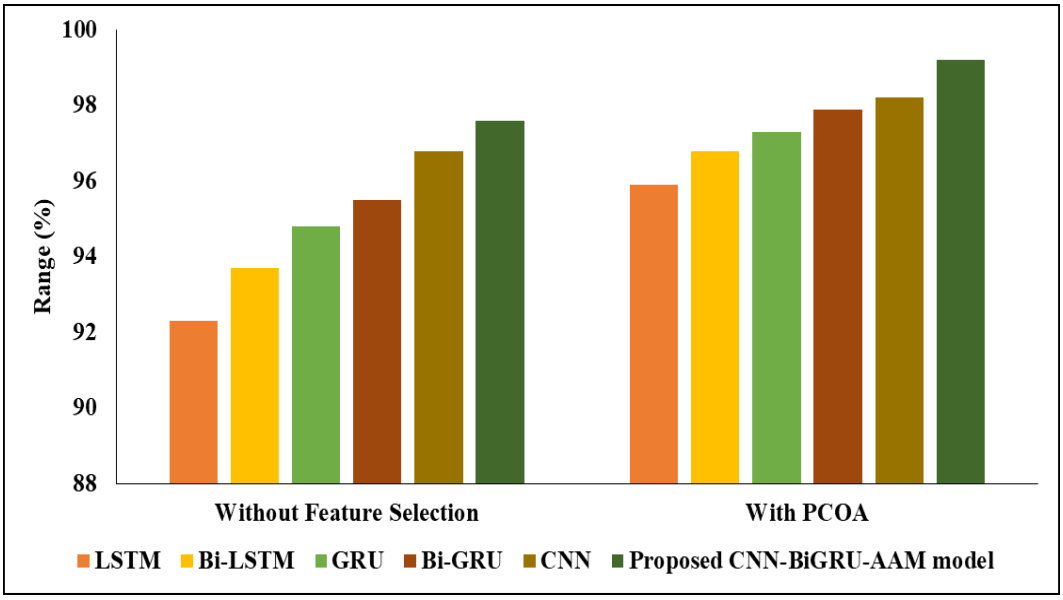

Fig 3. Graphical Validation of Feature Selection.

V. CONCLUSION

This study offers a thorough method for improving network anomaly IDS in IoT environments, in conclusion. The suggested system guarantees the accuracy and dependability of the input data by utilising the IoTID20 dataset and applying rigorous data preprocessing methods. By utilising mechanisms inspired by nature and derived from pine tree reproduction processes, the Pine Cone Optimisation algorithm (PCOA) for feature selection improves anomaly detection efficiency. Furthermore, the effective classification of intrusion detection events using CNN-BiGRU-AAM demonstrates the capability of deep learning. Additionally, the creative application of the Botox Optimisation Algorithm (BOA) for

hyperparameter tuning shows how to optimise model performance in a way that is human-inspired while maintaining the highest level of accuracy in anomaly detection tasks. The experimental findings demonstrate the efficacy of the recommended techniques, attaining a 99.2% maximum accuracy. These findings demonstrate the possibility of enhancing the security and dependability of IoT networks through the integration of deep learning methodologies and advanced optimisation techniques. All things considered, this research makes a significant contribution to the advancement of IDS for network anomalies, opening the door to more reliable and effective cybersecurity solutions in Internet of Things environments. Future research could examine the scalability, real-time execution, and robustness testing of suggested approaches to ensure wider relevance.

Data Availability

No data was used to support this study.

Conflicts of Interests

The author(s) declare(s) that they have no conflicts of interest.

Funding

No funding agency is associated with this research.

Competing Interests

There are no competing interests

References:

[1]. Y. Otoum, D. Liu, and A. Nayak, "DL-IDS: a deep learning–based intrusion detection framework for securing IoT," Transactions on Emerging Telecommunications Technologies, vol. 33, no. 3, Nov. 2019, doi: 10.1002/ett.3803.

[2]. A. DURAISAMY, M. SUBRAMANIAM, and C. R. RENE ROBIN, "An Optimized Deep Learning Based Security Enhancement and Attack Detection on IoT Using IDS and KH-AES for Smart Cities," Studies in Informatics and Control, vol. 30, no. 2, pp. 121–131, Jun. 2021, doi: 10.24846/v30i2y202111.

[3]. L. Aversano, M. L. Bernardi, M. Cimitile, R. Pecori, and L. Veltri, "Effective Anomaly Detection Using Deep Learning in IoT Systems," Wireless Communications and Mobile Computing, vol. 2021, pp. 1–14, Oct. 2021, doi: 10.1155/2021/9054336.

[4]. J. Rubia J and B. Lincy R, "Evolutionary meta-heuristic optimized model: An application to plant disease diagnosis," Journal of Intelligent & Fuzzy Systems, vol. 45, no. 6, pp. 10967–10983, Dec. 2023, doi: 10.3233/jifs-213423.

[5]. A. Verma and V. Ranga, "Machine Learning Based Intrusion Detection Systems for IoT Applications," Wireless Personal Communications, vol. 111, no. 4, pp. 2287–2310, Nov. 2019, doi: 10.1007/s11277-019-06986-8.

[6]. I. Idrissi, M. Azizi, and O. Moussaoui, "IoT security with Deep Learning-based Intrusion Detection Systems: A systematic literature review," 2020 Fourth International Conference On Intelligent Computing in Data Sciences (ICDS), Oct. 2020, doi: 10.1109/icds50568.2020.9268713.

[7]. A. Khatib, M. Hamlich, and D. Hamad, "Machine Learning based Intrusion Detection for Cyber-Security in IoT Networks," E3S Web of Conferences, vol. 297, p. 01057, 2021, doi: 10.1051/e3sconf/202129701057.

[8]. A. Sarwar, A. M. Alnajim, S. N. K. Marwat, S. Ahmed, S. Alyahya, and W. U. Khan, "Enhanced Anomaly Detection System for IoT Based on Improved Dynamic SBPSO," Sensors, vol. 22, no. 13, p. 4926, Jun. 2022, doi: 10.3390/s22134926.

[9]. Y. N. Soe, Y. Feng, P. I. Santosa, R. Hartanto, and K. Sakurai, "Implementing Lightweight IoT-IDS on Raspberry Pi Using Correlation-Based Feature Selection and Its Performance Evaluation," Advances in Intelligent Systems and Computing, pp. 458–469, Mar. 2019, doi: 10.1007/978-3-030-15032-7_39.

[10]. E. Altulaihan, M. A. Almaiah, and A. Aljughaiman, "Cybersecurity Threats, Countermeasures and Mitigation Techniques on the IoT: Future Research Directions," Electronics, vol. 11, no. 20, p. 3330, Oct. 2022, doi: 10.3390/electronics11203330.

[11]. Farah, A. Cross Dataset Evaluation for IoT Network Intrusion Detection. Ph.D. Thesis, The University of Wisconsin-Milwaukee, Milwaukee, WI, USA, 2020.

[12]. H. Liu and B. Lang, "Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey," Applied Sciences, vol. 9, no. 20, p. 4396, Oct. 2019, doi: 10.3390/app9204396.

[13]. Jency Rubia, Babitha Lincy, Sherin Shibi, Sheeba, "An effective transfer learning model for multiclass brain tumor classification using MRI images", AIP Conf. Proc. 2904, 020009 (2023), https://doi.org/10.1063/5.0170435

[14]. H. Hindy, R. Atkinson, C. Tachtatzis, J.-N. Colin, E. Bayne, and X. Bellekens, "Utilising Deep Learning Techniques for Effective Zero-Day Attack Detection," Electronics, vol. 9, no. 10, p. 1684, Oct. 2020, doi: 10.3390/electronics9101684.

[15]. J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, "CNN-Based Network Intrusion Detection against Denial-of-Service Attacks," Electronics, vol. 9, no. 6, p. 916, Jun. 2020, doi: 10.3390/electronics9060916.

[16]. I. Idrissi, M. Azizi, and O. Moussaoui, "Accelerating the update of a DL-based IDS for IoT using deep transfer learning," Indonesian Journal of Electrical Engineering and Computer Science, vol. 23, no. 2, p. 1059, Aug. 2021, doi: 10.11591/ijeecs. v23.i2.pp1059-1067.

[17]. G. Kocher and G. Kumar, "Machine learning and deep learning methods for intrusion detection systems: recent developments and challenges," Soft Computing, vol. 25, no. 15, pp. 9731–9763, Jun. 2021, doi: 10.1007/s00500-021-05893-0.

[18]. E.-H. Qazi, M. Imran, N. Haider, M. Shoaib, and I. Razzak, "An intelligent and efficient network intrusion detection system using deep learning," Computers and Electrical Engineering, vol. 99, p. 107764, Apr. 2022, doi: 10.1016/j.compeleceng.2022.107764.

[19]. S. Sriram, A. Shashank, R. Vinayakumar, and K. P. Soman, "DCNN-IDS: Deep Convolutional Neural Network Based Intrusion Detection System," Computational Intelligence, Cyber Security and Computational Models. Models and Techniques for Intelligent Systems and Automation, pp. 85–92, 2020, doi: 10.1007/978-981-15-9700-8_7.

[20]. R. Lohiya and A. Thakkar, "Intrusion Detection Using Deep Neural Network with AntiRectifier Layer," Lecture Notes in Networks and

Systems, pp. 89–105, 2021, doi: 10.1007/978-981-33-6173-7_7.

[21]. F. Safarov, M. Basak, R. Nasimov, A. Abdusalomov, and Y. I. Cho, "Explainable Lightweight Block Attention Module Framework for Network-Based IoT Attack Detection," Future Internet, vol. 15, no. 9, p. 297, Sep. 2023, doi: 10.3390/fi15090297.

[22]. R. Meddeb, F. Jemili, B. Triki, and O. Korbaa, "A deep learning-based intrusion detection approach for mobile Ad-hoc network," Soft Computing, vol. 27, no. 14, pp. 9425–9439, May 2023, doi: 10.1007/s00500-023-08324-4.

[23]. V. Ponnusamy, M. Humayun, N. Z. Jhanjhi, A. Yichiet, and M. Fahhad Almufareh, "Intrusion Detection Systems in Internet of Things and Mobile Ad-Hoc Networks," Computer Systems Science and Engineering, vol. 40, no. 3, pp. 1199–1215, 2022, doi: 10.32604/csse.2022.018518.

[24]. O. Sbai and M. Elboukhari, "Deep learning intrusion detection system for mobile ad hoc networks against flooding attacks," IAES International Journal of Artificial Intelligence (IJ-AI), vol. 11, no. 3, p. 878, Sep. 2022, doi: 10.11591/ijai. v11.i3. pp878-885.

[25]. Abbood, Z.A., Atilla, D.Ç. and Aydin, Ç., 2023. Intrusion Detection System Through Deep Learning in Routing MANET Networks. Intelligent Automation & Soft Computing, 37(1).

[26]. S. B. Ninu, "An intrusion detection system using Exponential Henry Gas Solubility Optimization based Deep Neuro Fuzzy Network in MANET," Engineering Applications of Artificial Intelligence, vol. 123, p. 105969, Aug. 2023, doi: 10.1016/j.engappai.2023.105969.

[27]. M. Prasad, S. Tripathi, and K. Dahal, "An intelligent intrusion detection and performance reliability evaluation mechanism in mobile ad-hoc networks," Engineering Applications of Artificial Intelligence, vol. 119, p. 105760, Mar. 2023, doi: 10.1016/j.engappai.2022.105760.

[28]. E. Altulaihan, M. A. Almaiah, and A. Aljughaiman, "Anomaly Detection IDS for Detecting DoS Attacks in IoT Networks Based on Machine Learning Algorithms," Sensors, vol. 24, no. 2, p. 713, Jan. 2024, doi: 10.3390/s24020713.

[29]. S. Hu et al., "Feature Extraction Approach for Distributed Wind Power Generation Based on Power System Flexibility Planning Analysis," Electronics, vol. 13, no. 5, p. 966, Mar. 2024, doi: 10.3390/electronics13050966.

[30]. M. Valikhan Anaraki and S. Farzin, "The Pine Cone Optimization Algorithm (PCOA)," Biomimetics, vol. 9, no. 2, p. 91, Feb. 2024, doi: 10.3390/biomimetics9020091.

[31]. S. Liu, W. Lin, Y. Wang, D. Z. Yu, Y. Peng, and X. Ma, "Convolutional Neural Network-Based Bidirectional Gated Recurrent Unit–Additive Attention Mechanism Hybrid Deep Neural Networks for Short-Term Traffic Flow Prediction," Sustainability, vol. 16, no. 5, p. 1986, Feb. 2024, doi: 10.3390/su16051986.

[32]. M. Hubálovská, Š. Hubálovský, and P. Trojovský, "Botox Optimization Algorithm: A New Human-Based Metaheuristic Algorithm for Solving Optimization Problems," Biomimetics, vol. 9, no. 3, p. 137, Feb. 2024, doi: 10.3390/biomimetics9030137.