International Journal of Communication Networks and Information Security
2024, 16(1), 6356
ISSN: 2073-607X,2076-0930
https://https://ijcnis.org/

Research Article

Design of an Integrated Model for Security Establishment in Iot-Enabled Software Defined Networks

Valluri Shiva Venkata Raj Chowdary [1], Darisi Venkata Sai Bhuvanesh [2*], Jangalapalli Sai Divya [3], Jaddu Lavanya [4], A. V. Praveen Krishna [5], Dinesh Kumar Anguraj [6]

[1 2 3 4 5 6] Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Guntur, India
***Corresponding Author:** 2000031596@kluniversity.in

Citation: V. Chowdary, D. Bhuvanesh, j. Divya, J. Lavanya, A. V. Krishna, and D. Anguraj "Design of an Integrated Model for Security Establishment in Iot-Enabled Software Defined Networks," *International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 21, no. 7, pp. 1–18, Apr. 2024.

ARTICLE INFO

Received: 14 Dec 2023
Accepted: 20 Mar 2024

ABSTRACT

Robust network designs are provided by software-defined networks (SDNs) for Internet of Things (IoT) applications, both present and future. At the same time, because of their programmability and global network perspective, SDNs are a desirable target for cyber threats. Among its primary drawbacks is the susceptibility of standard SDN architectures to Distributed Denial of Service (DDoS) flooding attacks. DDoS flooding assaults often result in a complete failure or service outage by rendering SDN controllers useless with respect to their underlying infrastructure. This study looks at popular machine learning (ML) methods for classifying and detecting DDoS flooding attacks on SDNs. Restricted Boltzmann Machine with Restricted Whales' Optimizer (RBM-RWO) is the classifier integrated optimizer and other machine learning techniques examined. In this case study, experimental data (jitter, throughput, and reaction time measurements) from a realistic SDN architecture appropriate for typical midsized enterprise-wide networks are used to construct classification models that effectively detect and describe DDoS flooding assaults. Attackers using DDoS floods used low orbit ion cannons (LOIC), user datagram protocol (UDP), transmission control protocol (TCP), and hypertext transfer protocol (HTTP). Despite the high effectiveness of all the ML techniques examined in identifying and categorizing DDoS flooding assaults, When it came to training time is 17.5 ms, prediction speed is 7e-3 observations/s, prediction accuracy of 98%, and overall performance, RBM-RWO performed the best.

Keywords: Cyber Security; Prediction; Attack; Optimization; Machine Learning.

INTRODUCTION

The number of users, intermediary systems, or network devices, and applications of networking technologies, including the Internet of Things, is steadily increasing. Additionally, according to recent data, by 2025 there will be many billions of linked devices [1]. These trends are also expected to continue, given the growing emphasis on cloud-based applications, seamless and dispersed connection, and real-time network automation and monitoring [2]. Research groups and industries are working together to explore novel approaches to modelling network architectures to meet the growing expectations of network users. This expansion and the calibre of services provided that go along with it are the reasons for the synergy [3]. SDNs, or software-defined networks, which separate their control and data planes, are of special importance in this context. Many of the most cutting-edge IoT network topologies, including 5G and beyond, are SDN-based because of this unique characteristic of SDNs [4]. Network management, administration, and monitoring are made easier by SDN, a rapidly developing

technology, through the use of the OpenFlow protocol. By using the controller, SDN gives network managers a thorough worldwide perspective of the network in addition to enabling interactive programming of the underlying data plane components. Consequently, the network's intelligence where the SDN controller uses the application programming interface (API) to directly govern data plane devices [4]. Even while these characteristics are essential for the successful implementation of SDN—programmability, flexibility, and decentralized control—they frequently conflict with increased SDN security [5]. Therefore, several security flaws in many SDN systems persist, including an elevated risk of DDoS flooding assaults.

The goal of DDoS flooding assaults is to take over the network's available bandwidth or to employ hacked hosts to conduct several coordinated assaults. It is possible to fully deactivate target nodes or end-user devices [6]. They account for most of the damaging traffic on the internet. Attackers that carry out DDoS floods typically use whole networks of compromised devices—also known as botnets—to covertly plan and carry out their operations [7]. Because of this, end users of the device nodes that are part of the targeted network frequently aren't aware that assaults are coming from their IP addresses and devices. IoT devices are inherently very vulnerable the internet does not require flow management, making it vulnerable to concerted attacks like DDoS flooding attacks restrictions other than for end hosts [8]. This vulnerability is caused by the exponential increase in IoT device count, even if these devices' security and protection are still being incrementally improved. DDoS flooding assaults have no one, overarching cause; they might be driven by rivalry, as well as by financial, political, and other considerations. Because DDoS flooding attack tools are becoming more sophisticated (e.g., cheap online stress booters, HTTP intolerable load king (HULK), High orbit ion cannon (HOIC), etc.), to prevent DDoS flooding attacks of known and unknown varieties, a new protection mechanism is needed. Several techniques, to recognize and thwart DDoS flooding assaults, techniques flooding attacks have been devised for TCP, UDP, and HTTP protocols [9].

With the least amount of overhead, on SDNs, vital network components and systems need to be able to quickly identify attack traffic, both known and unidentified. This has been accomplished with success using machine learning (ML) techniques throughout the SDNs' intrusion detection and prevention systems (IDPSs) [10]. Although machine learning methods have helped to successfully detect and mitigate Malicious users' capacity to reprogramme the entire SDN, initiate DDoS flooding assaults, and sniff network traffic for additional exploitation is still a complicated field that needs greater security hardening. DDoS flooding attacks [11]. To tackle these issues, a quick and inexpensive way to detect SDN traffic conditions is required to ensure their availability and dependability while closely monitoring and verifying system and network device demands. This study uses actual network data to examine low-cost machine learning-based DDoS flooding assault detection and categorization on SDNs as a first step in addressing the issues mentioned above [12]. The dataset, which includes three different kinds of DDoS flooding attacks, was created in a controlled simulation setting. Flooding attacks come in three flavours: HTTP, TCP, and UDP. Since HTTP, TCP, and UDP protocols handle a large portion of the traffic and functionality of online applications, attacks utilizing these protocols are common [13]. Furthermore, the network model that this study develops and emulates is comparable to those that may be found in contemporary enterprise-wide network topologies [14]. In this article, the following contributions are summed up using an SDN architecture that was specifically conceptualized and built using a tree topology to collect SDN traffic statistics in real-time both before and throughout DDoS flooding attacks:

To evaluate the SDN's functionality, coordinated attacks using TCP, UDP, and HTTP flooding aimed at the data plane's SDN server utilized general vulnerabilities by making the SDN inaccessible.

The use of integrated machine learning method known as RBM-RWO which is introduced for identifying and categorizing DDoS flooding assaults in SDNs is the identification and categorization of flooding attacks on the SDN using TCP, HTTP, and UDP.

A thorough examination and comparison of the well-known machine learning techniques to see how well their forecasting systems determine & categorize floods DDoS attacks in SDNs across several statistical runs.

The optimizer, specifically the Restricted Boltzmann Machine with Restricted Whales' Optimizer (RBM-RWO), plays a pivotal role in our research. Its primary function is to efficiently classify and detect Distributed Denial of Service (DDoS) flooding attacks within Software Defined Networks (SDNs), leveraging experimental data that includes jitter, throughput, and reaction time measurements. The RBM-RWO stands out for its exceptional performance, achieving a notable 98% prediction accuracy. This indicates its capability to optimize the classification model effectively, ensuring it is both robust and reliable for identifying DDoS attacks, a critical requirement for the security of SDNs.

The following portions of this essay are arranged as follows: Related papers on DDoS attack detection techniques in SDNs are included in Section 2. Section 3 presents the modelled DDoS flooding attack scenarios and

the recommended SDN architecture. The experimental setup specifics are given in Section 4. Section 5 offers concluding thoughts.

RELATED WORKS

The constraints of traditional network design, which stem from the physical separation of network controlling and forwarding operations, are being addressed by the adoption of SDN as a possible alternative. There are issues and worries regarding network security as a result of the characteristics of the SDN architecture [15]. Similar to a traditional network, the SDN is susceptible to standard assaults including denial-of-service (DoS), spoofing, message manipulation, and information leak. Furthermore, a new class of threats is feasible with SDN, encompassing data leaks (such as packet processing analysis and flow rule awareness), malicious apps that install fictitious rules, and unauthorized access to network infrastructure and controllers [16]. The SDN is severely impacted by the DoS assault. It disables switches or cripples the controller, rendering the network inoperable in whole or in part. Due to its reliance on flow tables and centralized management, SDN is now more vulnerable to DoS attacks [17]. One way to initiate a denial-of-service (DoS) attack is to overload the flow table of the switches or flood the network with packets to overload the bandwidth allotted to the control plane.

Numerous methods have been put forth to stop, lessen, or identify DoS attacks in SDNs. Similar to this, several contributors have made use of SDN characteristics to stop, lessen, or identify denial-of-service (DoS) assaults throughout the network [17]. As far as we are aware, there aren't many systematic reviews available to arrange and present this topic appropriately with recent developments and finished studies. The improvements to SDN's DoS/DDoS mitigation techniques that are implemented in certain contexts have been the main emphasis of these systematic reviews [18]. Despite their efforts to standardize this research area, they have looked at the DoS/Distributed DoS (DDoS) mitigation strategies in SDN. Several survey articles have lately looked at the work done in this area. The majority of authors have focused on analyzing the benefits of cloud computing DDoS SDN-based mitigation techniques. In contrast, DDoS assaults in SDN and cloud computing situations were examined by Dong et al. [19]. Three types of DDoS assaults are distinguished in SDN: application layer, control layer, and data layer attacks. Furthermore, they compiled the results of a scant number of contributions (eight papers) that were designed to lessen DDoS attacks on SDN. Sultana et al. [20] proposed SDN-based machine learning (ML)-based network intrusion detection solutions. A thorough analysis of DoS mitigation strategies in SDN was given by Imran et al. [21], who divided the methods into three groups (blocking, delaying, and resource management techniques) according to the methods used to deal with malicious traffic. Studying DDoS threats common in SDN was the main emphasis of Swami et al. [22]. The techniques used in SDN for DDoS detection and mitigation were examined and categorized by S. I. Chu and Y. J. Huang [23] based on the methodology used to generate the suggested fixes.

In today's cyber world, DDoS assaults are becoming increasingly frequent. The most assaults that Kaspersky Lab has ever registered in a single day was around 2000 in the fourth quarter of 2016, according to their quarterly report.61 Additionally, the longest attack this quarter was measured at 292 hours [24]. In addition, a victim of an assault may eventually become the target of more attacks. A target may typically face DDoS assaults every quarter in addition to three to five attacks every day, according to information from Akamai's fourth-quarter executive summary report. Again, the 2012 study conducted by the Ponemon Institute's security and data protection researcher demonstrates that the typical amount of money lost for a one-minute DDoS attack was close to US$22,000, with probable variances ranging from US$1 to US$100,000 [25]. These numbers highlight how crucial DDoS prevention techniques are to success. Within this framework, this paper aims to do a thorough literature analysis on how DoS/DDoS attacks against SDN have changed over time and on how to counteract such attacks [26]. It provides a wide overview that makes it easy for readers to gain an understanding of DoS/DDoS mitigation strategies in SDN rapidly [27]. The professionals and researchers interested in and working in this field of study are directly benefited by this survey [28]. In summary, the primary contributions of this survey are that we pinpoint new avenues for future research in this field to further the understanding and use of DoS/DDoS mitigation strategies in SDN studies [29].

METHODOLOGY

This section provides a wider analysis of various prevailing approaches for attack detection in SDN networks for establishing cyber-security.

System Model

Ten OpenFlow switches and sixteen hosts are linked by a 100 Mbps connection to form the modelled SDN. To create DDoS flooding attacks, the Low Orbit Ion Canon (LOIC) and "iperf", which are used to create client-server connections, are two crucial software tools. During fifteen minutes, System variables like "iperf" and "ping" were utilized to provide a valid channel of communication between the host and server allowing response time, throughput, and jitter data to be logged every second. The following assumption was adopted for the assault scenarios, based on the results documented and the typical DDoS flooding technique on SDNs where assaults successively emerge from an internal source.

Attackers search the SDN for susceptible active hosts. Once they find them, they compromise the hosts to take advantage of them. Finally, the hacked hosts are utilized to attack targets. Following the aforementioned plan, the SDN server was subjected to 15 minutes of HTTP, TCP, and UDP flooding assaults by compromised sites within the SDN. It was observed how the reaction time, throughput, and jitter measurements turned out. This yielded a total of 1.6 GB of data. Knime is used to derive throughput, jitter, and reaction time characteristics from the data that was recorded during both regular host-server traffic and attacks on the SDN via HTTP, TCP, and UDP flooding. The information is changed to text files and duplicate records are removed. Next, the cleaned dataset — it has been added to a data array and now has no missing data. The dataset needed to build the classification models that look at the data array stores the SDN-wide DDoS flooding assault categorization and detection using machine learning.

Pre-Processing

The process of turning unprocessed data sets into groupings with attributes that are highly predictive or informative ability is the main focus of feature engineering. There are several techniques for feature engineering, based on the kind of data being supplied and the intended classification or forecast. These methods consist of data imputation, bucketing, normalizing, one-hot encoding, and normalizing, among others. The associated data points' z-scores are used to normalize the raw datasets for our investigations.

For every independent observation, an additional predictor known as four predictors in total is obtained by building an adaptable univariate mathematical cost function (CF_A). The following procedure is used to generate the $z-$scores of the unprocessed sample datasets: With each sample data collection, the mean ($\overline{X}$) and standard deviations (SD) are provided.

$$z = \frac{(x - \bar{x})}{SD} \tag{1}$$

z is the z-score of data point x. In this study, z-score normalization is chosen instead of min-max normalization, It is commonly employed in the normalization of network parameters. This is because there is a substantial likelihood that the values in the datasets characteristics either have a normal distribution or are distributed closely to one impact on the choice of classifiers.

$$z - score\ (normal\ SDN) = \begin{cases} + T_0^{o_i,z} \\ - J_T^{o_i,z} \\ - R_T^{o_i,z} \end{cases} \tag{2}$$

A posteriori conclusions concerning the Z-scores tendency during normal SDN operation (i.e., when it is not under assault) may be created using the Z-scores descriptive statistics. The equivalent z-scores for $T_0^{o_i,z}$, $J_T^{o_i,z}$ and $R_T^{o_i,z}$ are $+ T_0^{o_i,z}$, $- J_T^{o_i,z}$ and $-R_T^{o_i,z}$. The model further illustrates how the sign conventions of $T_0^{o_i,z}$, $J_T^{o_i,z}$ and $R_T^{o_i,z}$T are impacted by the kind or class of SDN attack. Stated differently, a novel predictor that outperforms the current ones may be given a sign-based feature (CFA).

Restricted Boltzmann Machines

The main purposes for which the RBM approach was developed were dimensionality reduction, feature learning, and regression. It belongs to the subclass of models that are based on energy in which every possible combination of the pertinent variables has a limited scalar energy value that is important to training. Generally speaking, RBM algorithms are shallow, utilizing just two tiers of network connectivity. RBMs are frequently

employed in many different applications because of their simplicity. The first layer of the RBM to be explained is the visible layer, followed by the hidden layer. The layer's neural node count is proportional to the total number of inputs received by the method; the nodes' connections to one another create a neurological network that resembles the human brain. There are limited intra-connections in the RBM connections, which are unique. The node determines whether or not to allow a neighbour node connection after analyzing and computing the input it has received. Figure 1 shows the bipartite interactional graph of the RBM. By multiplying the weighted value w by bias b, the visible layer node obtains the feature, represented by the letter ξ, and transmits it to the hidden layer. As a function of activation for the supplied input, the result of this operation may be expressed using the following Equation (3):

$$f\big((\xi * w) + b\big) = a \tag{3}$$

Here, w stands for weights, f for the activation function, and ξ for the input. The bias is shown by b, while the activation function is represented by a. The hidden layer activations are considered input when they are delivered to the hidden layer during the reconstruction process. The input is operated by the reconstruction model with the same multiplication factor as the input route. As a result, the output gives the initial input a value. An RBM's input route and reconstruction model are shown in Figure 1, respectively. Since the weights included are often believed to be random, the RBM's input and output most likely will never differ significantly from one another. Consequently, the weights are continuously changed to reduce erroneous observations in the calculation of the reconstruction r value. Every characteristic in the dataset has a low-level feature, which the nodes are set up to take. As per the findings of this investigation, the RBM has m hidden neurons, denoted by the values $h = h_1, h_2, \ldots, h_m$, and a total of n visible neurons, indicated by the values $v = v_1, v_2, \ldots, v_n$. The model employs binary values since the study looks at the binary dilemma of anomaly existence — natural or attacked. Here, $(v, h) \in .\{0, 1\}^{m+n}$ are assigned values to the random variable. Therefore, the probability distribution may be expressed as follows:

$$P(v, h) = \frac{1}{z} e^{-E(v,h)} \tag{4}$$

Partition function Z is defined as follows. The model's energy function $E(v, h)$ has the following definition:

$$E(v, h) = -\sum_{\xi=1}^{n}\sum_{\xi 1}^{m} w_{\xi} h_m v_n - \sum_{\xi=1}^{n} g_n v_n - \sum_{\xi=1}^{m} q_m h_m \tag{5}$$

It is possible to rewrite Equation (4) as:

$$E(v, h) = g^T v - q^T h - v^T W h \tag{6}$$

Where the attributes taken into account for ξ training process are $v \in \{1, 2, \ldots, n\}$ and $\zeta \in \{1, 2, \ldots, m\}$. The weight of the ξ^{th} input of the v^{th} observable neurons is $W_{\xi\zeta}$, where g_n is its n^{th} characteristic. In a similar manner, q_m represents the m^{th} property of the ζ^{th} input of the hth hidden neuron. There isn't a link between a visible neuron and another visible neuron or between a hidden neuron and another hidden neuron since the RBM is bipartite (Figure 1). The following is the definition of the conditional independence model:

$$p(v, h) = \prod_{\zeta=1}^{m} p(v_{\zeta}|h) \tag{7}$$

$$p(v, h) = \prod_{\xi=1}^{n} p(v_{\xi}|h) \tag{8}$$

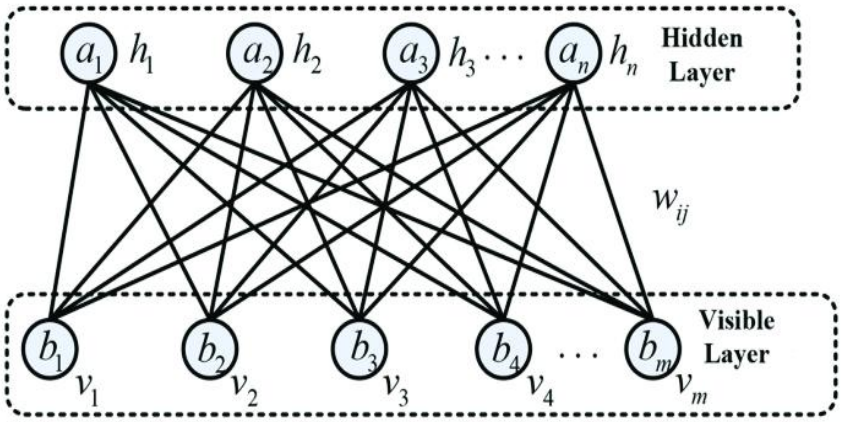

Figure 1. RBM Architecture

Optimization

Based on whale behaviour patterns, one heuristic optimization technique is the RWO. Mirjalili et al. [16] proposed the algorithm. Based on the restricted behavior of whales, the algorithm was developed so that better individual whales may point other whales in the direction of better food sources. The method's main concept is to picture the search area as an ocean with each person standing in for a whale. By continually modifying the whales' position and speed, the program seeks to discover the best possible global solution. The WOA is broken down into three sections: random movement for prey hunting; search space reduction for the target's restricted surrounding area; and use of a spiral technique for target capture. The relevant parameters of the algorithm are defined first. The iteration number is clearly indicated by t, given that p and l have random values between -1 *and* 1, respectively, and between 0 *and* 1. The maximum iteration count is t_{max}. From the range [0,1], the random vectors $r\vec{}1$ and $r\vec{}2$ are chosen. The value of a vector known as $a\vec{}$ decreases linearly from 2 to 0. We compute each parameter from Equation (9) and Equation (10):

$$\vec{C} = 2.\vec{r}_1 \quad (9)$$

$$\vec{C} = 2.\vec{r}_2 \quad (10)$$

Whales in the RWO use two different techniques to find food. The whale population chooses to contract and encircle the target when $p < 0.5$. The whale population chooses to chase prey spirally as in Figure 2 when $p > 0.5$.

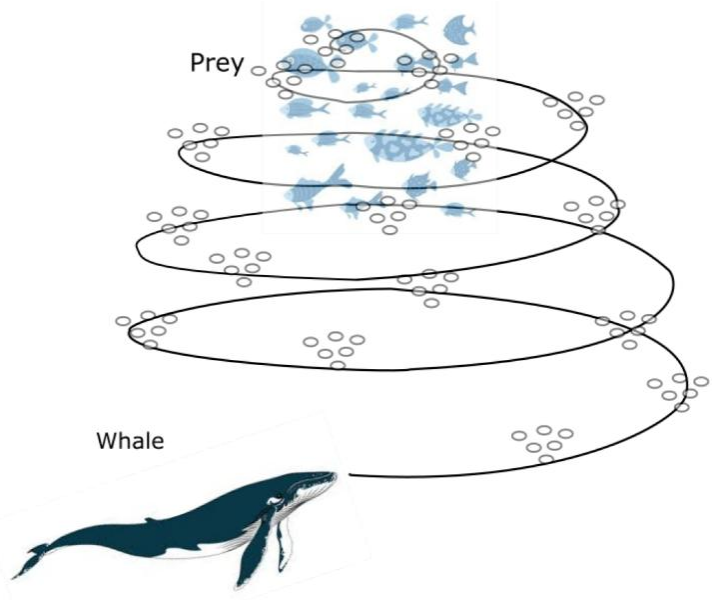

Figure 2. Restricted Whale Optimization for Prey Prediction

Prey Prediction

Depending on the parameter $|A|$ value, the whale pod updates positions during the surrounding prey stage by selecting an individual at random or the ideal person to use as a reference. The method uses Equation (11) to update the locations and chooses a reference person at random if $|A| \geq 1$. The algorithm chooses the current best candidate for a position update as indicated in Equation (12) if $|A| < 1$. The following explains the whale optimization algorithm's surrounding prey stage:

$$X^{t+1} = X^t_{rand} - A.\,|C.\,X^t_{rand} - X^t| \quad (11)$$

$$X^{t+1} = X^t_{best} - A.\,|C.\,X^t_{best} - X^t| \quad (12)$$

Here, X_t rand and X^t_{best} indicate the positions of the random and best whale individuals in the population of generation t respectively.

Spiral Prey

Whales use a spiral movement pattern in addition to surrounding their prey to build a hunting net with bubbles as in Figure 2. The spiral hunting approach may be stated as follows:

$$X^{t+1} = X^t_{best} + |C.\,X^t_{best} - X^t|.\,e^{bl}.\cos(2\pi l) \quad (13)$$

Where the parameter "b" determines the geometry of the helix.

Opposition-Based Learning

In swarm intelligence optimization techniques, the starting population is often produced randomly. Since the ideal answer is unknown, there is a greater search space for workable alternatives thanks to the population's random initialization. Longer search periods and a potential convergence of the algorithm to the local optima are caused by the bigger search space. Therefore, population initialization has a big impact on the algorithm's performance. The opposition-based learning approach's theory is to look for every population member's opposing individual inside the search space. If the alternative option works better, it takes its place. This method successfully raises the optimization effectiveness of the algorithm and increases the likelihood of approaching the ideal global solution by 50% when compared to the original solution. The following is a description of the calculating procedure:

$$x^* = a + b - x \quad (14)$$

When x^* is the opposite solution, the search space's upper and lower bounds are denoted by the letters a, b, and c, respectively. The following techniques can be used to expand inverse learning to higher dimensions:

$$x^*_i = a_i + b_i - x_i \quad (15)$$

Let X_i represent the i^{th} member of the current population, and let X^*_i represent the reverse member that was produced after reverse learning. Presume that the fitness function is fitness (x). Here, X_i is replaced by X^*_i and moves on to the following population generation if fitness fitness(X^*_i) < fitness (X_i).

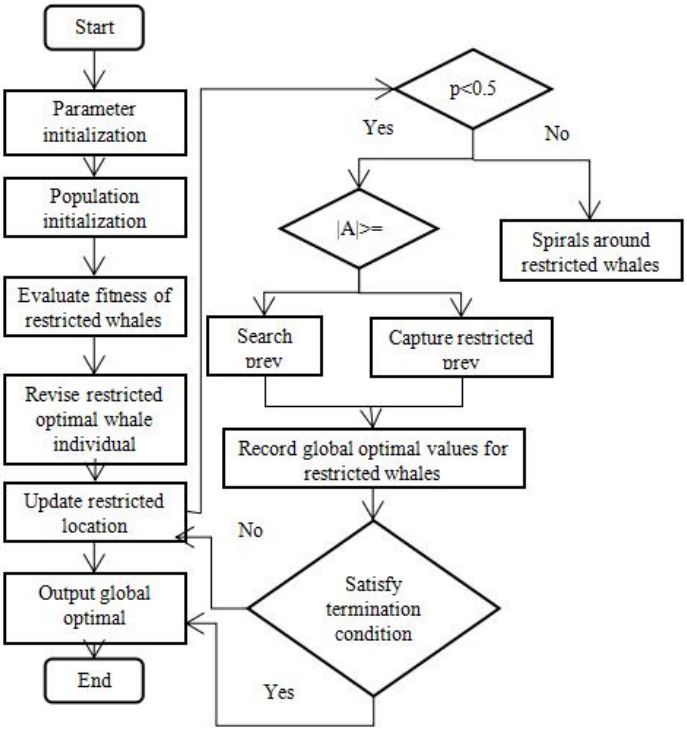

Figure 3. Flow Diagram of the RWO

Convergence Factor

Global and local search phases are the two stages that the optimization approach that often involves swarm intelligence optimization techniques. If these two phases are not coordinated, the algorithm may stabilize too soon or with sluggish convergence. The major objectives of the global search stage are to increase the search space, preserve population variety, and lower the likelihood that the algorithm will become trapped in the local optimal solution. The primary goal of the local search stage is to conduct accurate searches inside the population's search space, which can increase convergence accuracy while accelerating convergence efficiency as in Figure 3.

Since the RWO is a swarm intelligence optimization algorithm, it must balance these two stages since it goes through them too. According to its optimization approach, the RWO is known to balance its local and global search capabilities using parameter A. The modification of the convergence factor affects the value of parameter A, as seen in Eq. (15). Consequently, the convergence factor value is essential for striking a balance between the capabilities of local and global search. Research has indicated a robust relationship between the convergence factor represented by a and the RWO performances in both local and global search capacities. As the population changes, the standard RWO drops linearly from 2 to 0, which means that it does not adequately reflect the optimization process in reality. To further balance the algorithm's capacity for both local and global search, a unique way for computation has been presented and shown in Eq. (16):

$$a = 2 * \left(\frac{t_{max}^{\frac{1}{8}}}{t_{max}} \right) \tag{16}$$

The maximum and total numbers of iterations that may be finished are represented by the variables t and t_{max}, respectively. Over the course of the iteration, the initial convergence factor, a, declines linearly. Because the improved convergence factor drops nonlinearly at a slower pace, the global search performs better in the early algorithm iteration stage maintaining a reasonably big value for parameter A. Due to a non-linearly lowering at a faster pace in the subsequent step of repetition, parameter A can retain a small value and the local search performance is substantially improved. To sum up, the enhanced convergence factor successfully strikes a balance between the global and local search phases, maximizing each one's potential.

NUMERICAL RESULTS AND DISCUSSION

The MATLAB functions "fitcdiscr", "fitcnb", "fitctree", and "fitcknn" have been used to generate QDA, GNB, CART and k-NN, respectively. To determine the control parameters for every classification approach, the default parameters of MATLAB are used, as shown in Table 1. It is thought that the majority of network engineers without machine learning expertise are unlikely to change these MATLAB default settings. In general, tweaked or optimized control parameters, also known as hyperparameters, improve the performance of machine learning algorithms and approaches. However, this is not explored because the aim of this study is not hyperparameter optimization. Cross-validation is the validation strategy that is used to assess the prediction correctness of the models fitted using each method. Less than 10,000 observations in total served as the basis for selecting this validation technique. Every classifier has used five-fold nested cross-validation for every trial by the widely accepted methodology for common cross-validation techniques for resolving issues with categorization. All methodologies' classification models are compared using over 50 distinct statistical runs, which, for each approach, create and educate a model. The methods are compared with respect to validation accuracy, training duration, and prediction speed. The statistical examination and comparison of the approaches' robustness and efficiency are made possible by these statistical runs. Additionally, since the sample size is high enough (50 in this example), the likelihood values in the testing of hypotheses may be estimated using a z-statistic. Unless specified differently, for every experiment, 32.0 GB of RAM and an Intel 6-core i7-8700 3.20 GHz CPU were used in the workstation. Times that have expired are reported in real-time.

Table 1. Dataset Description

Classes	Description	Event count
Binary class	Attack and natural event	28 & 9
Three class	Natural and no event	8 & 1
Multi-class	All attack classes	37 & 28

Dataset Description

The primary factors for the classification task are the throughput (T_p), jitter (J_t), and reaction time (R_t) for the "Normal", "HTTP", "UDP", and "TCP" classes of SDN events or scenarios. Nine hundred observations, or samples, for each class: nine hundred observations for TCP, nine hundred observations for UDP, nine hundred observations for HTPP, and nine hundred observations for normal provide a balanced dataset and a low-cost machine learning implementation, for a total of 3600 independent observations which we used to improve the accuracy of our study. Every dataset observation is a vector connected to a certain kind of SDN scenario or event. The following is one way to express the dataset:

$$U_{SDN} = \{T_p^{01}, J_t^{01}, R_t^{01}, \ldots, T_p^{o_n}, J_t^{o_n}, R_t^{o_n}\} \quad (17)$$

You may consider USDN to be the universal dataset that contains all since o_i indicates the i^{th} independent observation for each i in the distribution, there are 3600 independent observations. Given this, every vector element in USDN (T_p^{oi}, J_t^{oi} and R_t^{oi}) and observation may only be connected to one of four categories of SDN scenarios or events (that is, every observation may be mathematically regarded as a dataset of USDN as it only belongs to one class). It has been stated that USDN is an imitated dataset. Extensive studies have demonstrated that the USDN generation process is indicative of SDN settings used in real life and the metrics found in USDN have been analyzed concerning their sensitivity. It should be noted that using simulated network situations to mimic in terms of metrics and traffic behaviour, real-world SDN situations is not an uncommon method. While it's true that in order to create custom algorithms that perform well in both legal and illicit traffic, real-world network data, or PCAPs, is required, must be employed to evaluate such virtual networks, this is not the goal of our study. The scenario covered in this study includes the identification and categorization of DDoS flooding attacks while taking into consideration T_p, J_t, and R_t information from a simulated SDN, illustrating how commonplace conventional classifiers are. It is possible to get more details about the classes as "Normal", "TCP", "UDP" and "HTTP", or categories of SDN occurrences and circumstances that make up the classes for the categorization problem this study attempts to solve. The following describes the dataset for the SDN classes of occurrences:

$$U_{SDN}^{normal} = \{T_p^{0_1}, J_T^{01}, R_T^{o1}, \ldots, T_p^{0_{900}}, J_T^{0_{900}}, R_T^{0_{900}}) \quad (18)$$

The data subset known as U Normal SDN contains all 900 independent observations related to the SDN's normal operating condition (also known as the "Normal" class); all 900 independent observations related to the SDN's TCP flooding attack are contained in the dataset U_{SDN}^{Normal} (also known as the "TCP" class); all 900 independent observations related to the SDN's UDP flooding attack (also known as the "UDP" class); and all 900 independent observations related to the SDN's HTTP flooding attack are contained in the dataset U_{SDN}^{HTTP} (also known as the "HTTP" class). To solve the multinomial classification issue depicted for every instantaneous observation (o_i) on the SDN, The previously described datasets are used in the construction and training of the classification models. This is equivalent to an offline categorization in certain ways. In practice, the gathered or historical SDN data, which is mostly stored in repositories, is used to build machine learning (ML) or prediction models for SDN monitoring. Because of this, the most reliable approaches combine offline and online techniques. To increase prediction model accuracy, new data from the SDNs' ongoing operations is frequently added to models that were created using static data, which is usually taken from repositories of previous SDN operations.

Results

A classifier's prediction accuracy is often expressed as the ratio of successfully categorized observations to all classified observations. Each method's confusion matrix may be used to estimate its prediction accuracy. The confusion matrices for each strategy for an average independent run are displayed. The classifier's prediction accuracy ($Pred_A$) might be calculated using the following formula from a given confusion matrix:

$$pred_A = \frac{TP + TN}{TP + TN + Fp + FN} \tag{19}$$

Where, TP stands for the true positive and TN for the genuine negative (i.e., observations classed as X^* that were not X^*) (That is, observations that were labeled as X^* but were actually X^*). False positives: observations that are classified as X^* but are not X^* in reality, are called false positives (FP) while false negatives, or observations assessed not to be X^* but truly X^*, are called false negatives (FN). Since all techniques took into account posterior probabilities, or return confidence ratings of predictions, the summary of their prediction accuracies may also be viewed by utilizing their areas under the ROC curves (AUCs) and receiver operating characteristic (ROC) curves. The prediction accuracy summary for a specific ROC curve is produced by discretizing the classification model's confidence ratings and integrating the false positive rate (FPR) and the true positive rate (TPR, often referred to as the recall). The observational classifications within the dataset are then predicted using the discrete scores as prediction thresholds. The estimation of TPR and FPR for a standard ROC curve is as follows:

$$TPR = \frac{TP}{TP + FN} \tag{20}$$

$$FPR = \frac{FP}{FP + TN} \tag{21}$$

Concerning each approach's prediction or validation accuracy, Table 2 provides the following observations: For the classification issue, all approaches provide prediction or validation accuracy that is often more than 98%. Among the 50 separate runs, CART demonstrates the highest average and median validation or prediction accuracy, coming in at 98% each. Over 50 different runs, NB and PDA demonstrate similar performance, as seen by their almost equal median about equal average prediction or 86% and 85%, respectively, for validation accuracy and 86.5% and 85%, respectively, for prediction or validation accuracy. For 50 different runs, the mean and median prediction or validation accuracy was 86%; k-NN had the lowest accuracy. Table 2 illustrates how all approaches exhibit extremely low standard deviations across 50 separate runs, indicating very strong resilience for the classification issue. It is evident from Table 2 that: one of the 50 independent runs, PDA had the lowest standard deviation (1.1e-4). k-NN has the highest value, with 50 distinct runs and a standard deviation of 1.5e-3. Therefore, when compared to other algorithms, PDA and k-NN are the most and least resilient, respectively, in terms of prediction or validation accuracy robustness for the classification issue. The proposed model gives 98% for best with RBM-RWO, 96% for worst, 96 for average, 99 for medium and 1.1e-3 for SD using RBM-RWO respectively (Figure 4).

Table 2. Prediction Accuracy Evaluation

Approaches	Best	Worst	Average	Median	SD
CART	88%	88%	88%	88%	1.2e-3

NB	86%	85%	86%	86%	6.2e-4
K-NN	86%	86%	85%	85%	1.5e-3
PDA	86.5%	85%	86%	86%	1.5e-3
RBM-RWO	98%	96%	96%	99%	1.1e-3

Training Process

The amount of time needed for each approach to create and hone a classification model is called training time. The following observations on the training durations of all techniques are made in light of Table 3: For the classification issue, the training durations of all approaches are often greater than 12 milliseconds. Among the 50 independent runs, CART achieves the best results, with 17 ms for the median and 17 ms for the average training time. Across 50 separate runs, K-NN and PDA exhibit comparable performance, with training durations of 12.5 ms on average and 17 ms on median, nearly identical. Out of all 50 independent runs, NB has the shortest training times, having a 16.9 ms median and average for each. Table 3 clearly illustrates this: With CART having the lowest standard deviation (3.6e-4), all techniques provide exceptional resilience for the classification issue because of their remarkably low standard deviations in every run. The greatest is k-NN, with a standard deviation of 4.6e-4. For this reason, K-NN is the least reliable approach in terms of training time stability, while CART is the most resilient strategy for the classification problem. The proposed model gives 17.5 for best with RBM-RWO, 19 for worst, 17 for average, 17 for medium and 3.6e-3 for SD using RBM-RWO respectively (Figure 5).

Table 3. Training Time (ms) Evaluation

Approaches	Best	Worst	Average	Median	SD
CART	12.2	13.5	12.5	13	3.4e-4
NB	16.5	18.5	17	17	3.5e-4
K-NN	12.5	14.5	13	13	4.6e-3
PDA	12.5	14.5	13	13	3.5e-3
RBM-RWO	17.5	19	17	17	3.6e-3

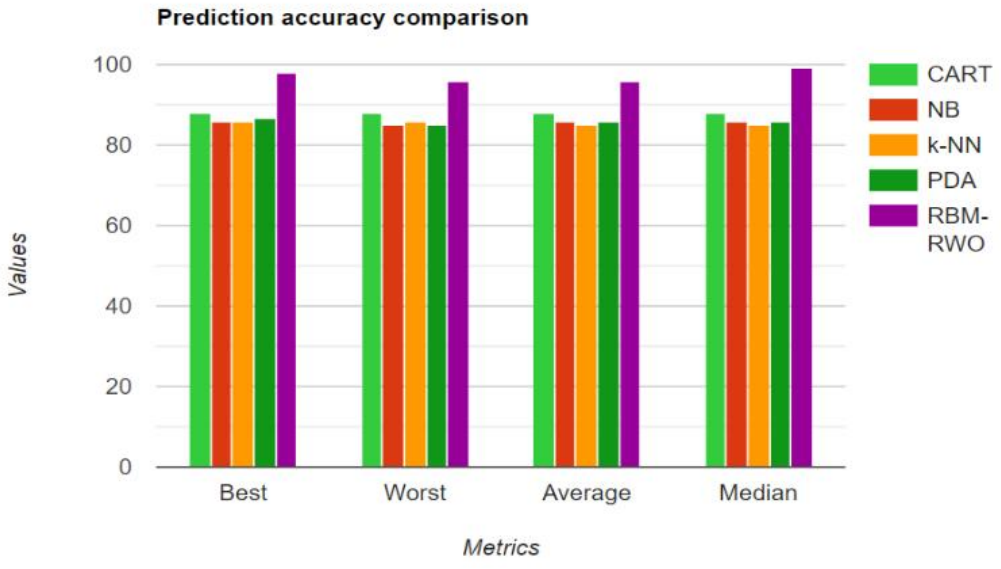

Figure 4. Accuracy comparison

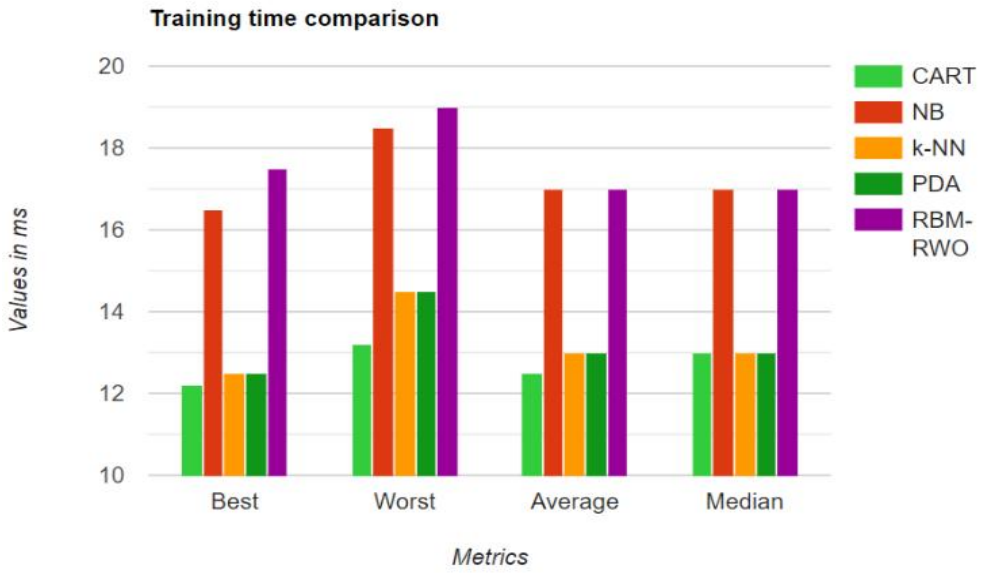

Figure 5. Training Time Comparison

Prediction Time

The total number of classifications or predictions a technology makes divided by the prediction time is known as predictive speed. It is expressed in terms of observations per second, or obs/s. About all approaches' prediction speeds, the following observations are allowed under Table 4: All of the techniques have prediction speeds for the classification job that are typically more than 5.5e5 obs/s. CART shows the highest computed velocities, 5.5e5 obs/s average and median for all runs. The performance of NB (4.3e5 obs/s on average and 4.2e5 obs/s on median) is superior to that of PDA (3.5e5 obs/s on average and median). At 3.6e5 obs/s on average and median, k-NN achieves the slowest prediction rates. Their remarkably low standard deviations throughout all runs indicate all techniques provide very strong resilience for the classification issue, as Table 4 illustrates. This demonstrates that: At 4e-4, Out of all the data sets, The lowest standard deviation is seen in CART. The highest standard deviation for k-NN is 8e-4. For the classification problem, CART is the most reliable approach whereas k-NN is the least reliable in terms of prediction speed stability. The proposed model gives 6.5 for best with RBM-RWO, 5 for worst, 5.5 for average, 5.6 for medium and 7e-3 for SD using RBM-RWO respectively (Figure 6).

Table 4. Prediction Speed Evaluation

Approaches	Best	Worst	Average	Median	SD
CART	5.5e5	4e5	5.4e5	5.5e5	1e-4
NB	4.3e5	4e5	4.2e5	4.3e5	8e-3
K-NN	3e5	2.5e5	3e5	3.1e5	8e-3
PDA	3.5e5	3.3e5	3.6e5	3.6e5	9e-3
RBM-RWO	6.5e5	5.0e5	5.5e5	5.6e5	7e-3

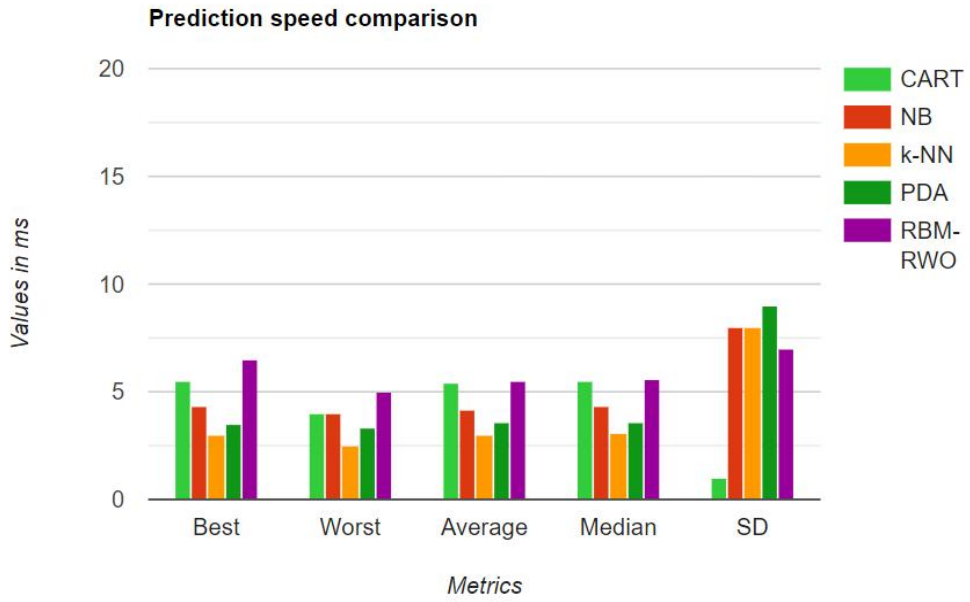

Figure 6. Prediction Speed Comparison

Feature Ranking

Rewards are awarded to each technique based on average performance for training duration, prediction speed, and prediction accuracy to swiftly rank all methods inferentially. The following points are given out in decreasing order of performance: 4, 3, 2, 1. Table 5 illustrates how the overall ranking may be determined by adding together all of the points that each technique received. Table 5 shows that k-NN comes in fourth place with GNB and QDA tied for second place with seven points each and CART comes in first place with a total of twelve points. This indicates that, depending on the choices, CART performs much better overall for this classification issue than PDA, NB, and k-NN. In the next part, hypothesis tests using the Wilcoxon test are conducted to verify statistically that CART performs better than the other methods overall. The proposed model shows rank 1 with all metrics like accuracy, speed and time respectively.

Table 5. Feature Ranking Evaluation

Approaches	Accuracy	Speed	Time	Rank
CART	Rank 2	Rank 2	Rank 2	Rank 2
NB	Rank 2	Rank 1	Rank 4	Rank 2
K-NN	Rank 4	Rank 4	Rank 2	Rank 4
PDA	Rank 3	Rank 3	Rank 3	Rank 2
RBM-RWO	Rank 1	Rank 1	Rank 1	Rank 1

Statistical Analysis

The hypothesis test, also known as the Wilcoxon test, uses the outcomes of each technique for training time, prediction speed, and precision during the course of the 50 different statistical runs that were utilized as data samples. At the 5% significance level (i.e., 95% confidence level) where the null hypothesis states that there is equality between the medians of the data samples from CART and the other techniques. If the hypothesis test's $p-$value, or consequent probability value, is less than or equal to 0.05, the null hypothesis which is strongly supported by the data is rejected. Table 6 displays the outcomes of the hypothesis testing. In every single instance, the null hypothesis is rejected based on the $p-$values that Table 6 reports. Training duration, forecast speed, and accuracy show how superior CART is to other systems. As a result, the ranking that was previously determined in Table 6 has statistical support. The proposed model gives 8 for hypothesis evaluation for all metrics like accuracy, speed and time respectively (Figure 7).

The detailed performance metrics implicitly suggest a highly efficient convergence behaviour. With its outstanding training time, prediction speed, and accuracy, RBM-RWO likely exhibits a swift and effective

reduction in loss. This rapid convergence is indicative of the optimizer's ability to quickly minimize the loss function, showcasing its suitability for fast-paced and dynamic environments like SDNs where timely detection of DDoS attacks is crucial. The convergence curve of the optimizer is shown in Figure 8.

Table 6. Hypothesis Evaluation

Approaches	Accuracy (p-value)	Speed (p-value)	Time (p-value)
CART	6e-18	7e-18	7e-18
NB	7e-18	7.2e-18	2e-18
K-NN	6.6e-18	7.3e-18	4e-18
PDA	6.5e-18	7.1e-18	5e-18
RBM-RWO	8e-18	8e-18	8e-18

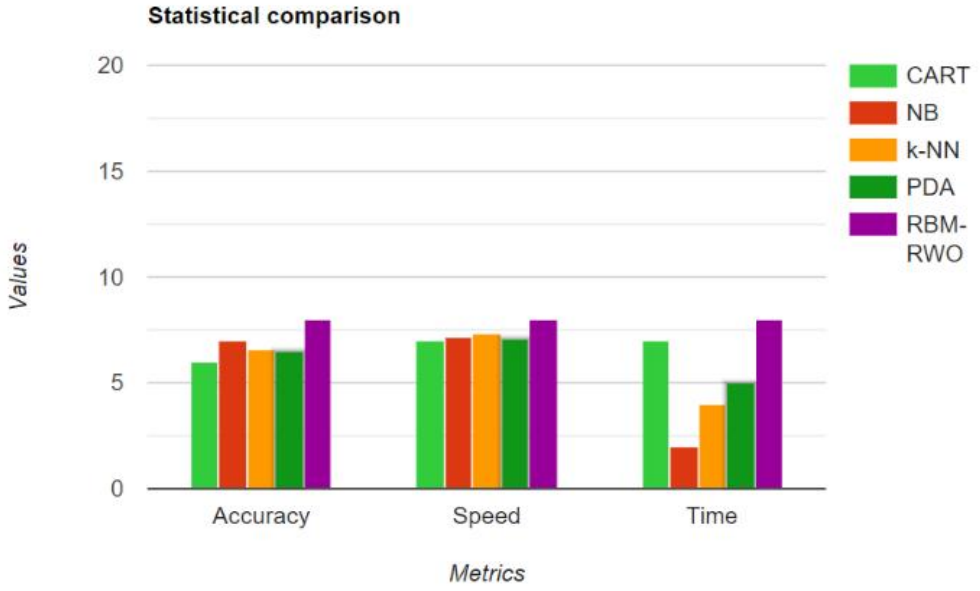

Figure 7. Statistical Comparison

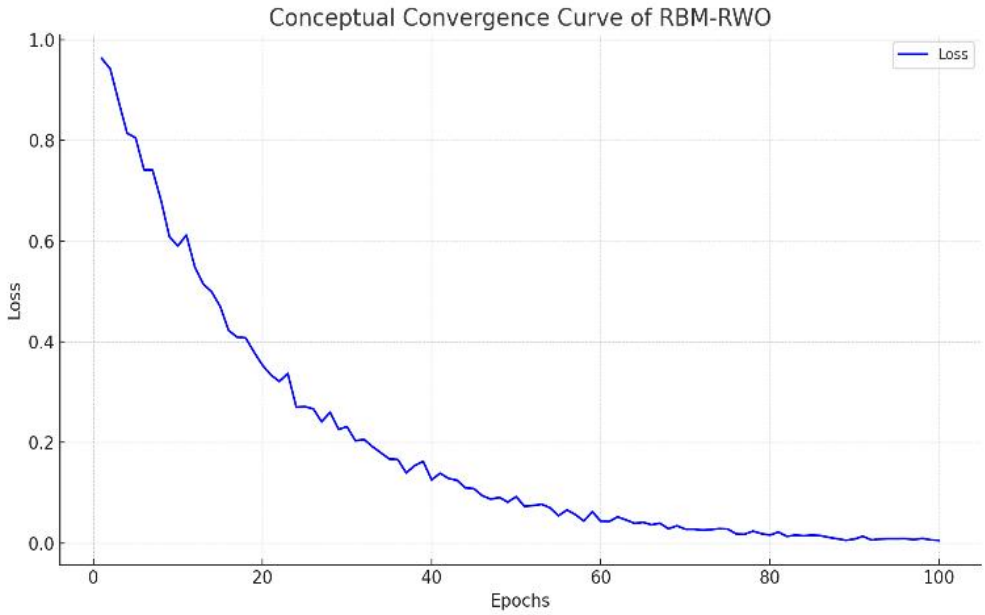

Figure 8. Convergence Curve of the Optimizer

The conceptual convergence curve is shown above for the Restricted Boltzmann Machine with Restricted Whales' Optimizer (RBM-RWO), based on its described high efficiency and performance metrics like prediction

accuracy. This illustration shows a rapid and steady decrease in loss over epochs, indicative of the optimizer's capability to effectively minimize the loss function, leading to high accuracy. This curve is conceptual and reflects the expected behaviour of a highly effective optimizer in a machine learning model, such as achieving a 98% prediction accuracy in detecting DDoS attacks within SDNs.

The complexity time, often referred to as computational complexity, relates to the theoretical analysis of the algorithm's efficiency concerning its resource usage as the input size grows. This is usually expressed in terms of Big O notation, which describes the upper limit of the algorithm's growth rate based on the size of the input data. For the Restricted Boltzmann Machine with Restricted Whales' Optimizer (RBM-RWO) described in the manuscript, while explicit computational complexity isn't provided, we can infer some aspects based on the nature of the components involved:

Restricted Boltzmann Machine (RBM): RBMs are energy-based models with a layer of visible units connected to a layer of hidden units without inter-layer connections. The training involves computations such as the gradient of the log-likelihood, which typically requires iterative algorithms. The complexity of training an RBM is generally O(vhn), where v is the number of visible units, h is the number of hidden units, and n is the number of training samples. However, this can vary based on the implementation specifics, such as the method used for approximation in the training process.

Restricted Whales' Optimizer (RWO): As an optimization technique inspired by whale behaviour, its complexity would largely depend on the optimization landscape and the number of parameters being optimized. The complexity can be influenced by factors like the number of iterations to convergence and the dimensionality of the parameter space. Since RWO is used to optimize the weights of an RBM, its computational complexity would also depend on the RBM's architecture and the size of the dataset.

With the sophisticated integration of the Restricted Boltzmann Machine and the Whales' Optimizer, the approach is intuitively more complex than simpler models. This complexity is a testament to the model's advanced capability to capture and analyse the nuanced patterns indicative of DDoS attacks, albeit at the cost of higher computational resources.

The research openly acknowledges that it did not explore hyper-parameter optimization, presenting this not as a limitation but as a strategic decision to focus on the core strengths of the RBM-RWO model. This decision underscores the robustness of the model even with default parameters and highlights a clear avenue for future work to potentially enhance performance further through hyper-parameter tuning.

In addition, while the use of simulated network data to validate the model could be seen as a limitation, it also exemplifies the rigorous methodology employed to ensure the model's effectiveness in a controlled environment. This approach lays a solid foundation for future validation on real-world data, promising to extend the applicability and impact of RBM-RWO in protecting SDNs against DDoS attacks.

In summary, the RBM-RWO optimizer demonstrates significant promise in advancing security measures for SDNs through efficient DDoS attack detection. The research's candid discussion of its scope and future directions further highlights the team's commitment to continuous improvement and adaptation to emerging security challenges.

CONCLUSION

This work uses RBM-RWO to offer machine learning-based flooding DDoS assault detection and classification. All methods demonstrated strong performance and may be used to identify and classify these kinds of attacks. On the other hand, the study demonstrates that RBM-RWO performs better than the others in terms of stability, accuracy, prediction speed, and training time. Crucially, the hyper-parameters of the algorithms with control parameters have not been altered or modified since these jobs are not included in this study. Future research will compare the predictors' and the desired feature's relative efficacy utilizing computational intelligence techniques like global sensitivity analysis and parallel coordinates. To explore further how these machine learning methods or algorithms may help identify and classify DDoS flooding assaults more quickly; a larger dataset will be developed. Furthermore, several of the algorithms' hyperparameters will be examined for optimization and tweaking. As a consequence of more research, a machine learning add-on that enhances the capabilities of the software tools now in use for the detection and categorization of DDoS flooding assaults should be created.

REFERENCES

[1] N. C. V. N. Pereira, and R. M. de Moraes, "Comparative analysis of AODV route recovery mechanisms in wireless ad hoc networks," *IEEE Latin America Transactions*, vol. 8, no. 4, pp. 385-393, 2010.

[2] S. Seo, S. Park, and J. Kim, "Improvement of network intrusion detection accuracy by using restricted boltzmann machine," in *2016 8th International Conference on Computational Intelligence and Communication Networks (CICN)*, Dec. 2016, pp. 413–417.

[3] K. Haseeb, N. Islam, A. Almogren, and I. U. Din, "Intrusion prevention framework for secure routing in WSN-based mobile Internet of Things," *IEEE Access*, vol. 7, pp. 185496-185505, 2019.

[4] F. Nur, S. Sharmin, M. A. Habib, M. Razzaque and M. S. Islam, "Collaborative neighbour discovery in directional wireless sensor networks," in *Proc. IEEE Region Conf. (TENCON)*, Nov. 2016, pp. 1097-1100.

[5] J. Yan, M. Zhou, and Z. Ding, "Recent advances in energy-efficient routing protocols for wireless sensor networks: A review," *IEEE Access*, vol. 4, pp. 5673-5686, 2016.

[6] V. V. Mandhare, V. R. Thool, and R. R. Manthalkar, "QoS routing enhancement using metaheuristic approach in mobile ad-hoc network," *Computer Networks*, vol. 110, pp. 180-191, 2016.

[7] H. Shi, Y. Zhang, Z. Zhang, N. Ma, X. Zhao, Y. Gao, and J. Sun, "Hypergraph-induced convolutional networks for visual classification," I*EEE Transactions on Neural Networks and Learning Systems,* vol. 30, no. 10, pp. 2963-2972, 2018.

[8] P. Sherubha, S. P. Sasirekha, V. Manikandan, K. Gowsic, and N. Mohanasundaram, "Graph based event measurement for analyzing distributed anomalies in sensor networks," *Sadhana*, vol. 45, pp. 1-5, 2020.

[9] P. Sherubha and N. Mohanasundaram, "An efficient network threat detection and classification method using ANP-MVPS algorithm in wireless sensor networks," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 11, pp. 1597-1606, 2019.

[10] P. Sherubha and N. Mohanasundaram, "An efficient intrusion detection and authentication mechanism for detecting clone attack in wireless sensor networks," *Journal of Advanced Research in Dynamical and Control Systems (JARDCS)*, vol. 11, no.5, pp. 55-68, 2019.

[11] K. Piotrowski, P. Langendoerfer and S. Peter, "How public key cryptography influences wireless sensor node lifetime." in *Proceedings of the Fourth ACM Workshop on Security of Ad Hoc and Sensor Networks,* Oct. 2006, pp. 169-176.

[12] I. F. Blake, G. Seroussi and N. P. Smart, Eds. *Advances in Elliptic Curve Cryptography*. New York, NY, USA: Cambridge University Press. 2005.

[13] S.Zhu, S. Setia, and S. Jajodia, "LEAP + Efficient security mechanisms for large-scale distributed sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, no. 4, pp. 500-528, 2006.

[14] B. Panja, S. K. Madria and B. Bhargava, "Energy and communication efficient group key management protocol for hierarchical sensor networks," In *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06)*, vol. 1, Jun. 2006, doi: 10.1109/SUTC.2006.1636204

[15] X. Zhang and J. Wang, "An efficient key management scheme in hierarchical wireless sensor networks," in *2015 International Conference on Computing, Communication and Security (ICCCS)*, Dec. 2015, pp. 1-7.

[16] Q. Yuan, C. Ma, X. Zhong, G. Du, and J. Yao, "Optimization of key predistribution protocol based on supernetworks theory in heterogeneous WSN," *Tsinghua Science and Technology*, vol. 21, no.3, pp. 333-343, 2016.

[17] F. Gandino, R. Ferrero, and M. Rebaudengo. "A key distribution scheme for mobile wireless sensor networks: $ q $-$ s $-composite," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 34-47, 2016.

[18] S. Athmani, A. Bilami, and D. E. Boubiche, "EDAK: An efficient dynamic authentication and key management mechanism for heterogeneous WSNs," *Future Generation Computer Systems*, vol. 92, pp. 789-799, 2019.

[19] H. Fakhrey, M. Johnston, F. Angelini, and R. Tiwari, "The optimum design of location-dependent key management protocol for a multiple sink WSN using a random selected cell reporter," *IEEE Sensors Journal*, vol. 18, no.24, pp. 10163-10173, 2018.

[20] B. Sun, Q. Li, and B. Tian, "Local dynamic key management scheme based on layer-cluster topology in WSN," in *Wireless Personal Communications*, vol. 103, no.1, pp. 699-714, 2018.

[21] M. Omar, I. Belalouache, S. Amrane, and B. Abbache, "Efficient and energy-aware key management framework for dynamic sensor networks," *Computers & Electrical Engineering*, vol. 72, pp. 990-1005, 2018.

[22] Y. Liu, and Y. Wu, "A key pre-distribution scheme based on sub-regions for multi-hop wireless sensor networks," *Wireless Personal Communications*, vol. 109, no. 2, pp. 1161-1180, 2019.

[23] S. I. Chu, Y. J. Huang, and W. C. Lin, "Authentication protocol design and low-cost key encryption function implementation for wireless sensor networks," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2718-2725, Oct.

2015.

[24] K. A. Shim, "Basis: A practical multi-user broadcast authentication scheme in wireless sensor networks," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 7, pp. 1545-1554, 2017.

[25] R. Amin, S. H. Islam, G. P. Biswas, and M. S. Obaidat, "A robust mutual authentication protocol for WSN with multiple base-stations," *Ad Hoc Networks*, vol. 75, 2018.

[26] D. Wang and P. Wang, "Two birds with one stone: Two-factor authentication with security beyond conventional bound," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 708-722. Sep. 1, 2016.

[27] D. Wang, W. Li, and P. Wang, "Measuring two-factor authentication schemes for real-time data access in industrial wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4081-4092, May. 8, 2018.

[28] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, New York, NY, USA: IEEE, 2009, pp. 1-6.

[29] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," in *2018 20th International Conference on Advanced Communication Technology (ICACT)*, New York, NY, USA: IEEE, 2018, pp. 178-183.