electronics

Article

Fuzzy Local Information and Bhattacharya-Based C-Means Clustering and Optimized Deep Learning in Spark Framework for Intrusion Detection

Brunel Elvire Bouya-Moko, Edward Kwadwo Boahen and Changda Wang *

Faculty of School of Computer Science & Communication Engineering, Jiangsu University, Zhenjiang 212013, China; bouyabrunel@gmail.com (B.E.B.-M.); eboahen@gctu.edu.gh (E.K.B.)
* Correspondence: changda@ujs.edu.cn

Abstract: Strong network connections make the risk of malicious activities emerge faster while dealing with big data. An intrusion detection system (IDS) can be utilized for alerting suitable entities when hazardous actions are occurring. Most of the techniques used to classify intrusions lack the techniques executed with big data. This paper devised an optimization-driven deep learning technique for detecting the intrusion using the Spark model. The input data is fed to the data partitioning phase wherein the partitioning of data is done using the proposed fuzzy local information and Bhattacharya-based C-means (FLIBCM). The proposed FLIBCM was devised by combining Bhattacharya distance and fuzzy local information C-Means (FLICM). The feature selection was achieved with classwise info gained to select imperative features. The data augmentation was done with oversampling to make it apposite for further processing. The detection of intrusion was done using a deep Maxout network (DMN), which was trained using the proposed student psychology water cycle caviar (SPWCC) obtained by combining the water cycle algorithm (WCA), the conditional autoregressive value at risk by regression quantiles (CAViaR), and the student psychology-based optimization algorithm (SPBO). The proposed SPWCC-based DMN offered enhanced performance with the highest accuracy of 97.6%, sensitivity of 98%, and specificity of 97%.

Keywords: big data; Apache Spark; intrusion detection; bhattacharya-based C-means clustering; data augmentation

Citation: Bouya-Moko, B.E.; Boahen, E.K.; Wang, C. Fuzzy Local Information and Bhattacharya-Based C-Means Clustering and Optimized Deep Learning in Spark Framework for Intrusion Detection. *Electronics* **2022**, *11*, 1675. https://doi.org/10.3390/electronics11111675

Academic Editor: Amir Mosavi

Received: 18 April 2022
Accepted: 20 May 2022
Published: 25 May 2022

1. Introduction

Nowadays, a large amount of data flows every second, making intrusion detection a complex task. Thus, the intrusion detection system needs an effective and enhanced detection technique that could discover the activities of intrusion and severe threats to the security of the network. Day to day, the networks are becoming computer technology-reliant which elevates the requirement for secure networks. Therefore, computer network security is essential for establishing confidentiality, integrity, and availability, but these factors cannot avert the need for detection of intrusion. Susceptible computer models and networks need to be made secure for preventing the hazard of illegal admittance. IDS scans all network packets and attempts to categorize traffic as invasive or genuine. The discovery of intrusion is a procedure that starts where the firewall ends [1]. The utilization of network-based services is escalating in a rapid manner, so cyber security issues are rising. Cyber security detection has recently become a challenging research task in network communications. IDS is a system that monitors the traffic of the network for detecting malevolent tasks and produces alert messages to the control station [2]. The network traffic data size is becoming more and more complicated, which is a very complex task for processing with classical processing tools, because of increasing internet-based services. Quick and effective cyber security IDS is a complex issue, because of huge and complicated traffic [3,4]. A pragmatic cyber security IDS must be capable of practicing

across a huge network size as quickly as possible for determining the malevolent traffic as early as possible [5].

Offering privacy and protection to big data is a most imperative issue faced by developers for managing the security, particularly with a huge growth of the networks and quick growth of data produced with various bases [6]. The growth and development present more liberty to hackers for launching suspicious attacks and utilizing these progress methods for detecting the intrusion [7,8]. Mastering several kinds of internet protocol formats is the foundation for the exploration of cyberspace search engines. Meanwhile, developers utilize the IDS for detecting intrusions in order to elevate the competence of malevolent attack detection and its predictions for determining the attacks earlier [9]. For ingesting and processing large data, big data tools like Apache Spark [2] and Apache Hadoop are utilized. Hadoop is an emerging tool that is considered a leader in the big data world. It utilizes Map-Reduce for processing large data and stores it in HDFS. Spark proved to be faster in contrast to Hadoop as it utilizes in memory computations. It handles streaming data, SQL queries, graph processing, and machine learning, and can execute on Hadoop or utilize the HDFS for storing data [10]. Due to changes in time, various types of attacks are produced, or the configuration of existing attacks is altered, and thus a smart IDS is needed [11]. Adaptive and rapid discovery is considered an imperative aspect of any IDS method as huge attacks are produced on a daily basis. A technique must be adaptive to discover novel attacks [12].

In recent days, several ML techniques were adapted for detecting intrusion [13]. IDS utilizes ML techniques and classifiers for detecting intrusions. They utilized the KDD99Cup dataset and devised several attributes for ML classifiers to discover several attacks. Various methods utilized KNN to detect and classify intrusions. In [14], SVMs were utilized for detecting intrusion. In [15], a hybrid learning technique was utilized, which combined naïve Bayes and K-means clustering for detecting the intrusion. For enhancing the accuracy of the technique, some works combined optimization techniques with ML techniques for enhancing the performance of detection. In [16], the IDS was devised using PSO, then the KNN technique was added for discovering the intrusion. The techniques demonstrated that there are methods for improving the precision of the KNN approach. In [17], a hybrid technique was utilized, which combined GA and SVM for detecting intrusion. In [18], a combination of KNN, SVM, and PSO was devised to produce weights, which were further utilized to create a group of classifiers with improved accuracy for detecting intrusion [19]. In [20], the assessment of classification methods was elaborated, like the RBF Network and deep models for detecting the intrusion. In [21], a comparative assessment of classification models was analyzed for detecting the intrusion, and they utilized several techniques, such as logistic, random forest, and random tree for discovering intrusions in the network [12].

The first contribution of this research work was to devise an optimization-driven deep model for detecting intrusion using the spark framework. The series of steps followed for intrusion detection in the spark framework were data partitioning, feature selection, data augmentation, and intrusion detection. The input data were initially passed to the data partitioning module in order to partition the data for further processing. The second contribution was that the data partitioning was carried out based on the proposed FLIBCM where the Bhattacharya distance was incorporated with FLICM. Once the data partitioning was done, the selection of features was accomplished with classwise info gain. Once the feature selection was done, the data augmentation was performed. After the completion of these processes, the intrusions in the network were detected using the DMN and were trained with developed SPWCC. The developed SPWCC was devised by integrating the WCA, the CAViaR model, and SPBO, which becomes the third contribution of the research. Finally, the experimental analysis stated that the proposed SPWCC-based DMN improved performance with the highest accuracy of 97.6%, sensitivity of 98%, and specificity of 97%.

The organization of the paper is structured as follows: Section 2 analyzes the literature review. Section 3 portrays the materials and methods. Sections 4 and 5 portray the results and discussions. Section 6 depicts the conclusions.

2. Literature Review

The eight classical techniques on the basis of detecting intrusion with big data are examined. Ashfaq Khan and Juntae Kim [22] devised a spark for detecting intrusion based on deep architecture. Here, the Conv-AE was used for determining the misuse attack. However, the technique failed to use real-time streaming data. Valerio Morfino and Salvatore Rampone [23] used the ML method for determining cyber-attacks. Here, the supervised ML technique was adapted in the MLlib library using Apache Spark for processing big data. The cloud model was used to discover the implication of elasticity and scalability. However, the time adapted for training was increased due to huge instances. Ramin Atefinia and Mahmood Ahmadi [24] devised a multiarchitectural modular deep neural network model for minimizing the rate of false positives to detect the intrusion. The technique consists of the feed-forward module with two recurrent modules and the output weights of these methods were fed to the aggregator module to generate the solution of the model. However, the technique failed to utilize a custom feature extractor for improving the performance. Ahmed Mahfouz et al. [25] developed a technique to discover intrusion in the network. The method also produced a new feasible dataset called Game Theory and Cyber Security, which matched real-world criteria for analyzing the performance of the technique. However, the speed of processing was slow. Hassan, M.M et al. [26] developed a hybrid deep learning technique for effectively determining intrusion of the network using WDLSTM network and CNN. The technique utilized deep CNN for extracting the meaningful features from IDS big data and WDLSTM for retaining the long-term dependencies amongst the mined features for preventing overfitting. However, the technique did not analyze bigger datasets for obtaining real-time IDS. Zhong, W et al. [27] developed a BDHDLS for detecting the intrusion. Here, the BDHDLS utilized content features and behavioral features for understanding the network traffic features and information accumulated in the payload. However, the technique was unable to minimize the computational resources. Su, T et al. [28] utilized BLSTM and attention mechanism with the Bat algorithm for detecting intrusion. The attention technique was utilized for screening the network flow vector consisting of packet vectors produced by the BLSTM model that can generate the key features for classifying the network traffic. Moreover, multiple convolutional layers were employed for capturing the local features. Here, the multiple convolutional layers were utilized for processing the data. However, the technique produced less accuracy. Haggag, M et al. [29] developed the IDS model, namely the DLS-IDS, on the basis of deep learning for detecting intrusion. Here, the NSL-KDD dataset was utilized for training and testing. However, the technique was unable to handle other datasets. Pooja TS and Purohit Shrinivas Acharya developed an automated method for network intrusion detection. The technique was modeled by exploiting the long short-term memory approach.

3. Materials and Methods

3.1. Methods

These days, a large amount of data flows each second, and thus the task of detecting intrusions is becoming so complex [30,31]. Hence, the intrusion detection model needs an effective and enhanced detection technique that can discover intrusive tasks and severe threats that occur in the network. This research devised a technique for detecting intrusion using big data in the spark model, which comprises master and slave nodes. At first, the input data is fed to the data partitioning phase. The data partitioning is carried out in the master node. The data partitioning is done using the proposed FLIBCM, where the FLICM [32] is modified based on Bhattacharya distance. After, data partitioning, the selection of features is done with classwise info gain [33]. Once the features are selected, the data augmentation is done using an oversampling process. After data augmentation, the detection of intrusion is done using the DMN [34] and is trained with the developed SPWCC. The developed SPWCC algorithm is devised by integrating the WCA [35], CAViaR [36],

and SPBO [37]. Figure 1 shows the structure of intrusion detection using the proposed SPWCC-based DMN with the spark model.

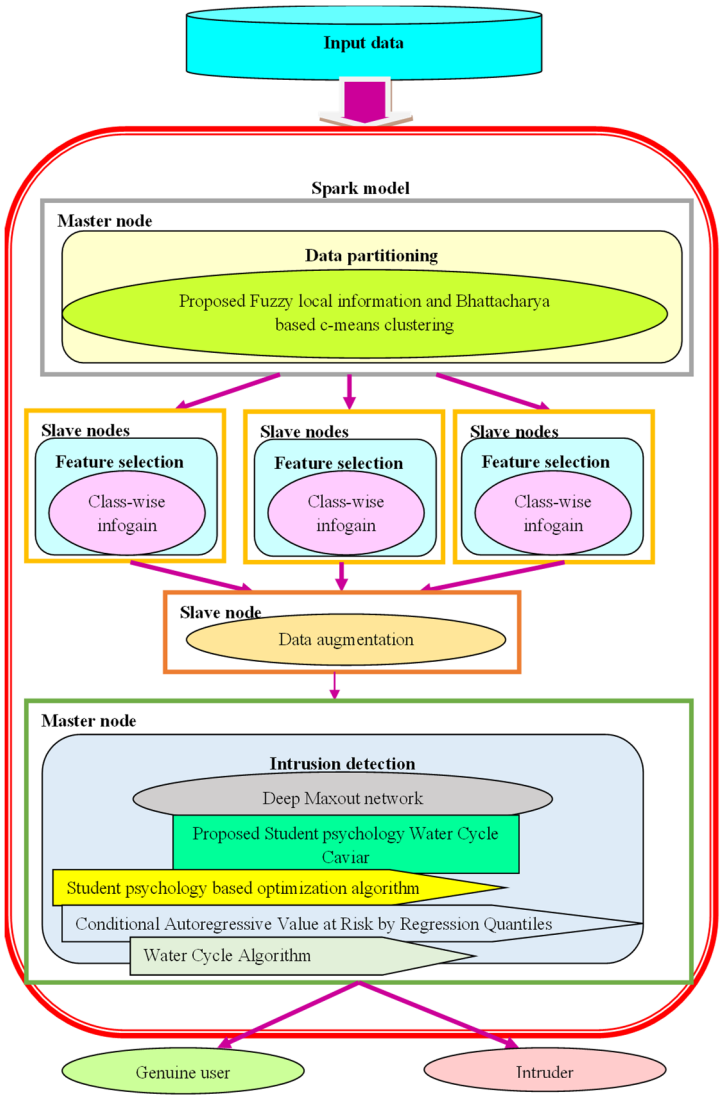


Figure 1. The architecture of intrusion detection using the proposed SPWCC-based DMN with spark model.

Considering the input big data is expressed as M with various attributes and stated in Equation (1).

$$M = \left\{G_{g,h}\right\}; (1 \leq g \leq F); (1 \leq h \leq H) \tag{1}$$

where $G_{g,h}$ represents g^{th} data with h^{th} attribute, F denotes total data, and H denotes the total attributes.

The data partitioning is made into small units that provide the effectual processing of data. Here, the spark model generates high power for processing as the slave node server comprises the ability to execute in parallel. Here, the input data $G_{g,h}$ is uploaded in the spark that considers the master node wherein data is partitioned into several subsets. Here, the partitioning of data is done in the master node using the proposed FLIBCM, which is devised by combining FLICM [32] and Bhattacharyya distance. The proposed FLIBCM is utilized for combining local spatial and local gray level data in a fuzzy manner for preserving the noise insensitiveness. It is also utilized for controlling the impact of neighborhood pixels based on distance from the central pixel. The FLIBCM is effective as it

is free from parameter selection. Thus, this technique assists to minimize the overall time taken for processing.

Master nodes in spark: The master nodes are liable to split the generated input big data and send distributed data to various slave nodes for feature selection. Here, the novel fuzzy factor, represented in Equation (2).

$$G_{ki} = \sum_{\substack{j \in N_i \\ i \neq j}} \frac{1}{b_{ij}+1}\left(1+u_{kj}\right)^m b_{jk}^2 \tag{2}$$

where, b_{ij} signifies Bhattacharyya distance between pixels i and j, and b_{jk}^2 symbolizes Bhattacharyya distance between pixels k^{th} cluster and j^{th} pixel, u_{kj} symbolizes membership degree of j^{th} pixel in k^{th} cluster, and m represents the weighting unit on each fuzzy membership, and N_i is the neighbors falling into a window around the i^{th} pixel.

With $G_{k,i}$, a robust FCM model was made for clustering the data, which was obtained by combining local spatial and gray level data and is represented in Equation (3).

$$J_m = \sum_{i=1}^{N}\sum_{k=1}^{c}\left[u_{ki}^m b_{ik}^2 + G_{ki}\right] \tag{3}$$

The two imperative criterions for J_m at its local minimal extreme, based on u_{ki} and v_k, are represented in Equations (4) and (5).

$$u_{ki} = \frac{1}{\sum_{j=1}^{c}\left(\frac{b_{ik}^2+G_{ki}}{b_{ij}^2+G_{ji}}\right)^{\frac{1}{m-1}}} \tag{4}$$

$$v_k = \frac{\sum_{i=1}^{N} u_{ki}^m x_i}{\sum_{i=1}^{N} u_{ki}^m} \tag{5}$$

The steps of the proposed FLIBCM are provided in Figure 2.

Here, the data are split into various data subsets using the proposed FLIBCM and are given as

$$G_{g,h} = \{E_l\}; (1 \leq l \leq L) \tag{6}$$

where L signifies total subsets generated from inputted data. It should be noted that the total data subsets is equal to the total slave nodes, as each set of data is given to the slave node set for acquiring definite features.

The procedure assists to generate the number of input variables while developing a predictive model. It is effective to reduce the number of input variables for minimizing the cost of computation. The choice of feature is an effective and important step for high dimensionality data mining tools and is termed as an essential part of processing huge data. The info gain evaluates the attributes by computing the information gain based on class.

Slave nodes in spark: Each slave node attains a subgroup of data with the master node and carries out feature selection. The input to l^{th} slave node is represented in Equation (7).

$$E_l = \{L_{s,t}\}; (1 \leq s \leq J); (1 \leq t \leq N) \tag{7}$$

where $L_{s,t}$ signifies l^{th} subset data with t^{th} attribute of s^{th} data. From each data subset $L_{s,t}$, the features are chosen with information gain.

Consider K representing a set that comprises p data samples with q distinct classes. The training dataset consists of a sample of the class R. The information gain evaluates the attributes by computing their info gain with class and is represented in Equation (8).

$$IG(F) = R\left(p_1, \ldots, p_q\right) - \varepsilon(F) \tag{8}$$

where $R(p_1, \ldots, p_q)$ represents expected data, and $\varepsilon(F)$ signifies entropy. Here, the expected information needed to categorize the offered sample is represented in Equation (9).

$$R(p_1, \ldots, p_q) = -\sum_{i=1}^{q} \frac{p_i}{p} \log_2(p) \tag{9}$$

where $\frac{p_i}{p}$ signifies the probability that a random sample belongs to a class K_i. Consider that feature F contains v distinct values $\{f_1, f_2, \ldots, f_v\}$, which can split the training set into v subsets $\{K_1 K_2, \ldots, K_v\}$ such that K_i signifies a subset that poses value f_i for the feature F. Consider K_j to consist of K_{ij} samples of class i.

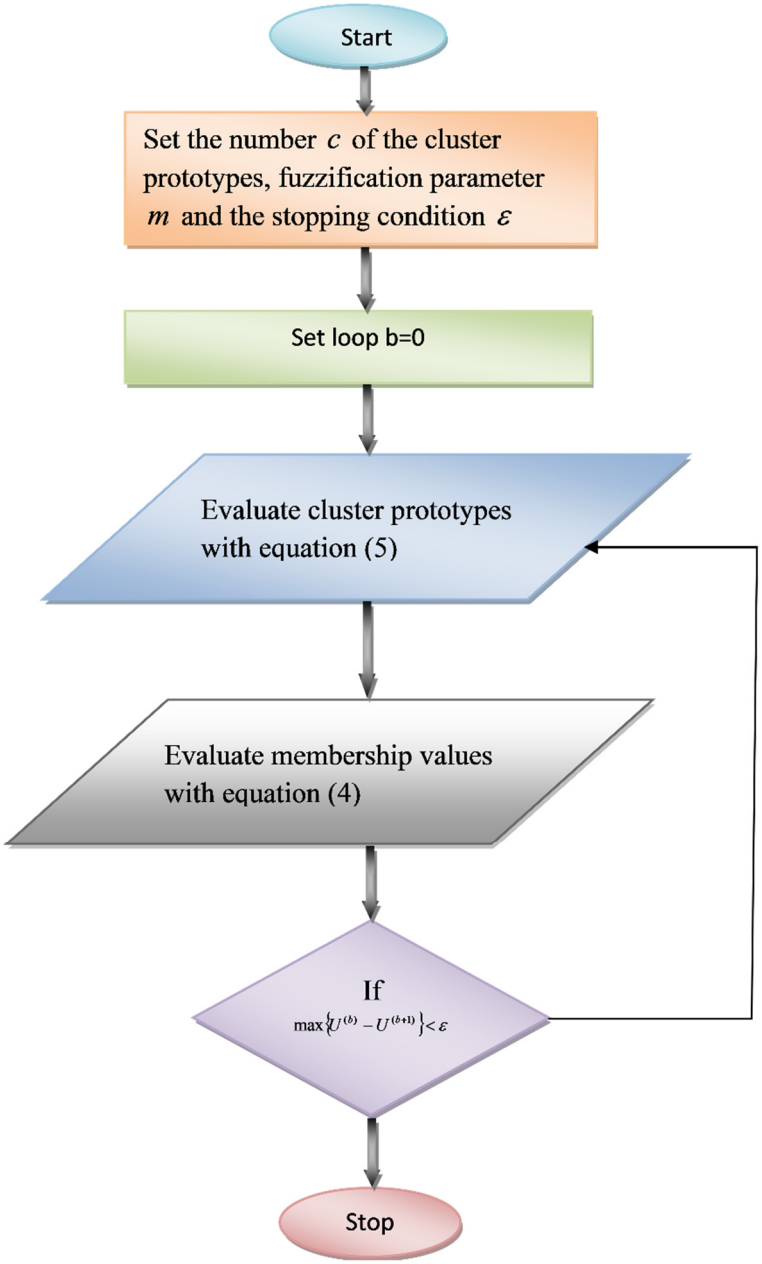


Figure 2. Steps of developed FLIBCM algorithm.

The entropy of the feature F is represented in Equation (10).

$$R\varepsilon(F) = \sum_{j=1}^{v} \frac{p_{ij} + \ldots + p_{mj}}{p} \times R(p_{ij}, \ldots, p_{qj}) \tag{10}$$

Hence, the slave node set-1 carries out feature selection with info gain, the selected features are expressed as D and are subjected to the data augmentation, which is performed in a slave node.

Data augmentation is a method that provides the user with a significant increase in the data diversity, which assists to minimize the overfitting problem, and then is effective in the data generation process. Here, the augmentation of data was employed with the oversampling method. After the selection of features is performed, the data augmentation is carried out on the selected features D, which is carried out in a slave node. The augmented data C are subjected to master nodes for detecting the intrusion, which is carried out using the DMN and the training of the DMN is done using the proposed SPWCC algorithm.

3.2. Proposed SPWCC-Based DMN for Intrusion Detection

The detection of intrusion is a technique which helps to observe the activities of malicious behavior and is accumulated using security data. It is beneficial for determining harmful behaviors or privacy violations. Here, the proposed SPWCC-based DMN is employed to classify the intrusions that occurred in the network, and the process is carried out in the master node. Here, the SPWCC-based DMN is devised by integrating SPWCC into the DMN [34] for choosing optimum weights contained in the DMN. The DMN is able to address several optimization issues and possesses the ability to deal with real-time data. It has the ability to handle a large amount of data and is easy to implement. The structure of the DMN and the process for training the DMN are portrayed below.

The DMN [34] is a structured Maxout, which utilizes various benefits of the activation function. The Maxout network helps to enhance the accuracy and facilitate optimization using dropout and is termed a fast model. It provides easy computation and possesses the ability to handle huge data. The Maxout offers an optimization process by averting the hidden units from inactive to transiting. Meanwhile, the Maxout unit represents a trainable activation function. Here, the input given to the DMN is augmented data C. Figure 3 depicts the structure of the DMN.

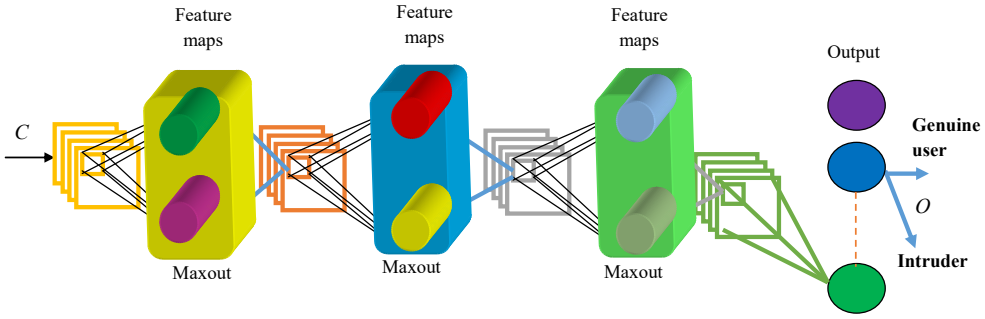


Figure 3. Structure of the DMN.

The DMN is a kind of trainable activation function that has a multilayer structure. Assume an input $b \in I^d$ wherein x represents the raw input vector or state vector of a hidden layer, and the activation of a hidden unit is given as follows:

$$f_{i,j}^{1} = \max_{j \in [1,\, k_1]} x^T \omega_{ij} + h_{ij} \tag{11}$$

$$f_{2,j}^{2} = \max_{j \in [1,\, k_2]} {f_{i,j}^{1}}^T \omega_{ij} + h_{ij} \tag{12}$$

$$f_{i,j}^{m} = \max_{j \in [1,\, k_m]} {f_{i,j}^{m-1}}^T \omega_{ij} + h_{ij} \tag{13}$$

$$f_{i,j}^{n} = \max_{j \in [1,\, k_n]} {f_{i,j}^{n-1}}^T \omega_{ij} + h_{ij} \tag{14}$$

$$f_i = \max_{j \in [1, k_n]} f_{i,j}^n \tag{15}$$

Here, n represents the total number of layers in the DMN, and $\omega_{i,j}$ denotes weights, and h signifies bias. The output obtained from the DMN is the detection result denoted as O.

The training of the DMN is performed using the proposed SPWCC algorithm, which inherits the benefits of both the SPBO and WCC algorithms for detecting intrusion. Here, the WCC is devised by combining the WCA and CAViaR, wherein the WCA [35] contains the scalability to address several engineering designs and constrained optimization issues and is effective in providing qualitative solutions with improved computational efficiency. Meanwhile, the CAViaR [36] offers satisfactory solutions. Thus, the combination of the WCA and CAViaR in the WCC algorithm offers a globally optimal solution with improved performance. Meanwhile, the SPBO [37] is motivated by a student's psychology who attempted to provide more effort in enhancing the performance of examinations to become the best student. The SPBO generates an optimal solution with rapid convergence mobility. It overwhelms local optima to generate the global best solution. Hence, the combination of the SPBO in the WCC generates an optimum solution that helps to tune the optimal weights of the DMN for detecting intrusions. The steps of the developed SPWCC are expressed as follows:

(Step 1) Initialization:

The first step is the initiation of the solution, which is given in Equation (16).

$$T = \left\{ T_1, T_2, \cdots, T_\vartheta, \cdots T_\rho \right\}; 1 \le \vartheta \le \rho \tag{16}$$

where ρ signifies the total count of solutions and T_ϑ indicates the ϑ^{th} solution.

(Step 2) Evaluation of error:

The best solution is determined with error and is termed a minimization issue, and thus the solution having less mean square error (MSE) is chosen as the best solution and represented in Equation (17).

$$MS_{err} = \frac{1}{v} \sum_{u=1}^{v} \left[\xi_f - O \right]^2 \tag{17}$$

where ξ_f denotes expected output and O signifies the predicted output, while v denotes the count of data, where $1 < u \le v$..

(Step 3) Discovery of update equation:

For generating a global optimum solution, the updated equation of WCC is represented in Equation (18).

$$T_{River}^{l+1} = \left(\alpha_0 + \alpha_1 T_{River}^{l-1} + \alpha_2 T_{River}^{l-2} + \alpha_1 f\left(T_{River}^{l-1} \right) + \alpha_2 f\left(T_{River}^{l-2} \right) \right) (1 - RandI) + RandI T_{sea}^{l} \tag{18}$$

where, $Rand$ is uniformly distributed random number, I is an arbitrary number between 1 and 2, the position of the river at iteration l is denoted as T_{River}^{l}, and the position of the sea at iteration l is expressed as T_{sea}^{l}, α expresses p vector of the unknown parameter, and $f(T)$ expresses the fitness function of T.

Form SBPO [37], wherein the students try to provide more effort in the study as compared to the effort provided by an average student in the class and try to follow the effort of the best student. This is mathematically expressed in Equation (19).

$$T^{l+1} = T_{best} + rand \times \left(T_{best} - T^l \right) \tag{19}$$

where $rand$ signifies a random number between 0 and 1, T_{best} represents the marks obtained by the best student, and T^l is the marks obtained by l^{th} student.

$$T^{l+1} = T_{best}(1 + rand) - rand \times T^l \tag{20}$$

$$T_{best} = \frac{T^{l+1} + rand \times T^{l}}{1 + rand} \tag{21}$$

Substitute the best solution of the SPBO with the optimal solution of the WCC. Here, we substitute Equation (21) in Equation (18):

$$\begin{aligned} T_{River}^{l+1} &= \left(\alpha_0 + \alpha_1 T_{River}^{l-1} + \alpha_2 T_{River}^{l-2} + \alpha_1 f\left(T_{River}^{l-1}\right) + \alpha_2 f\left(T_{River}^{l-2}\right)\right) \\ &(1 - RandI) + RandI\left(\tfrac{T^{l+1} + rand \times T^{l}}{1 + rand}\right) \end{aligned} \tag{22}$$

$$\begin{aligned} T_{River}^{l+1} &= \left(\alpha_0 + \alpha_1 T_{River}^{l-1} + \alpha_2 T_{River}^{l-2} + \alpha_1 f\left(T_{River}^{l-1}\right) + \alpha_2 f\left(T_{River}^{l-2}\right)\right)(1 - RandI) \\ &+ \tfrac{RandIT_{River}^{l+1}}{1+rand} + \tfrac{RandI.rand \times T_{River}^{l}}{1+rand} \end{aligned} \tag{23}$$

$$\begin{aligned} T_{River}^{l+1} &- \tfrac{RandIT_{River}^{l+1}}{1+rand}\left(\alpha_0 + \alpha_1 T_{River}^{l-1} + \alpha_2 T_{River}^{l-2} + \alpha_1 f\left(T_{River}^{l-1}\right) + \alpha_2 f\left(T_{River}^{l-2}\right)\right) \\ &(1 - RandI) + \tfrac{RandI.rand \times T_{River}^{l}}{1+rand} \end{aligned} \tag{24}$$

$$\begin{aligned} T_{River}^{l+1}\left[1 - \tfrac{RandI}{1+rand}\right] &= \left(\alpha_0 + \alpha_1 T_{River}^{l-1} + \alpha_2 T_{River}^{l-2} + \alpha_1 f\left(T_{River}^{l-1}\right) + \alpha_2 f\left(T_{River}^{l-2}\right)\right) \\ &(1 - RandI) + \tfrac{RandI.rand \times T_{River}^{l}}{1+rand} \end{aligned} \tag{25}$$

$$\begin{aligned} T_{River}^{l+1}\left[\tfrac{1+rand-RandI}{1+rand}\right] &= \left(\alpha_0 + \alpha_1 T_{River}^{l-1} + \alpha_2 T_{River}^{l-2} + \alpha_1 f\left(T_{River}^{l-1}\right) + \alpha_2 f\left(T_{River}^{l-2}\right)\right) \\ &(1 - RandI) + \tfrac{RandI.rand \times T_{River}^{l}}{1+rand} \end{aligned} \tag{26}$$

$$\begin{aligned} T_{River}^{l+1}\left[1 - \tfrac{RandI}{1+rand}\right] &= \left(\alpha_0 + \alpha_1 T_{River}^{l-1} + \alpha_2 T_{River}^{l-2} + \alpha_1 f\left(T_{River}^{l-1}\right) + \alpha_2 f\left(T_{River}^{l-2}\right)\right) \\ &(1 - RandI) + \tfrac{RandI.rand \times T_{River}^{l}}{1+rand} \end{aligned} \tag{27}$$

(Step 4) Re-evaluation of the error:

After completing the process of updating, the error of each solution is re-evaluated using equation (17).

(Step 5) Termination:

The steps are iterated repeatedly until the maximal time l_{max} is reached, wherein an optimal solution is determined. The proposed SPWCC-based DMN provides the detected output O, which is either an intruder or a genuine user.

4. Results

The effectiveness of the SPWCC-based DMN was evaluated by altering training data considering specificity, sensitivity, and accuracy. The assessment was done considering without attack, with DoS attack, with probe attack, and with other attacks.

The execution of the developed SPWCC-based DMN was performed in the PYTHON tool with the pySpark package installed in Windows 10 OS, 4GB RAM, and Intel processor. The analysis of techniques was done using the NSL-KDD dataset. The NSL-KDD dataset [38] is a dataset used for discovering the intrusion by adapting a complete set rather than selecting a small portion. It consists of various data files, like KDDTrain+.ARFF, KDDTrain+.TXT, KDDTrain+20Percent.ARFF, KDDTrain+_20Percent.TXT, KDDTest-21.ARFF and KDDTest-21.TXT.

4.1. Evaluation Metrics

The efficiency of the developed SPWCC-based DMN was evaluated by employing the metrics given as follows:

4.1.1. Accuracy

It is defined as a measure of data, which is precisely preserved and is expressed as

$$Acc = \frac{P + Q}{P + Q + H + F} \tag{28}$$

where P signifies true positive, Q symbolizes true negative, F denotes true false positive, and H is a false negative.

4.1.2. Sensitivity

It refers to the ratio of the number of true positives with respect to the total number of positives.

$$Sen = \frac{P}{P + H} \tag{29}$$

where P refers to true positives, H is the false negatives.

4.1.3. Specificity

It refers to the ratio of negatives which are correctly detected.

$$Spec = \frac{Q}{Q + F} \tag{30}$$

where Q is true negative and F signifies false positive.

5. Discussion

5.1. Comparative Analysis

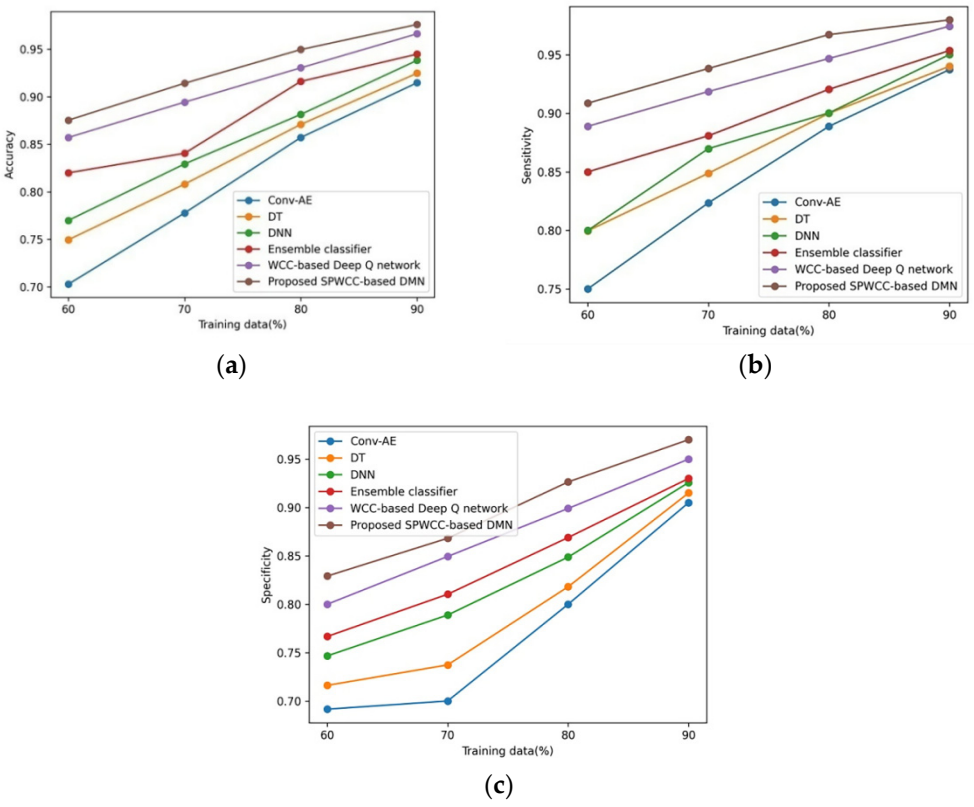
The assessment was done with techniques, like Conv-AE [22], DT [38], DNN [3], Ensemble classifier [4], WCC-based Deep Q network, and the proposed SPWCC-based DMN. Here, the assessment was carried out using without attack scenarios, DoS attack, probe attack, and other attack scenarios.

5.1.1. Assessment without Attack

Figure 4 presents the assessment of techniques without an attack scenario. The assessment of techniques with accuracy is displayed in Figure 4a. For 90% data, the accuracy evaluated by Conv-AE, DT, DNN, Ensemble classifier, the WCC-based Deep Q network, and the proposed SPWCC-based DMN was 0.915, 0.925, 0.938, 0.945, 0.966, and 0.976. The performance improvements of the Conv-AE, DT, DNN, Ensemble classifier, the WCC-based Deep Q network with respect to the proposed SPWCC-based DMN considering accuracy were 6.25%, 5.225%, 3.893%, 3.176%, and 1.024%. The assessment of techniques with sensitivity is displayed in Figure 4b. For 90% of data, the sensitivity evaluated by the proposed SPWCC-based DMN was 0.980, which is more than other comparative methods. The sensitivity was increased when increasing the training data percentage. The assessment of techniques with specificity is displayed in Figure 4c. The specificity evaluated by the proposed SPWCC-based DMN was 0.970, which is 6.701%, 5.670%, 4.536%, 4.123%, and 2.061% better than the Conv-AE, DT, DNN, Ensemble classifier, and the WCC-based Deep Q network, respectively, for 90% training data. The specificity was improved when increasing the training data percentage and the maximum specificity was obtained at 90% of training data.

5.1.2. Assessment with DoS Attack

Figure 5 presents the assessment of techniques with the DoS attack scenario. The assessment of techniques with accuracy, sensitivity, and specificity is displayed in Figure 5a–c, respectively. The training data varied from 60% to 90%, and the maximum performance results were obtained at 90% of training data. The accuracy, sensitivity, and specificity of the proposed SPWCC-based DMN were 0.973, 0.980, and 0.970, respectively, which is higher than the other comparative methods, such as Conv-AE, DT, and DNN, Ensemble classifier, and the WCC-based Deep Q network.



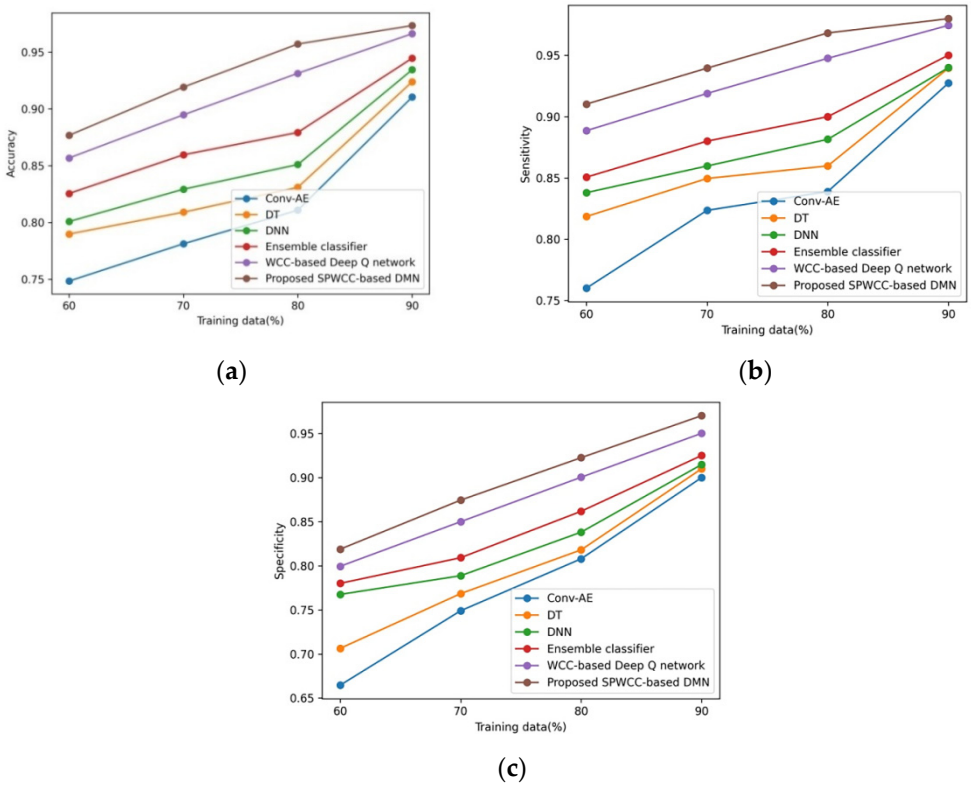
(**a**) (**b**)

(**c**)

Figure 4. Assessment of techniques without attack considering (**a**) accuracy, (**b**) sensitivity, and (**c**) specificity.

5.1.3. Assessment with Probe Attack

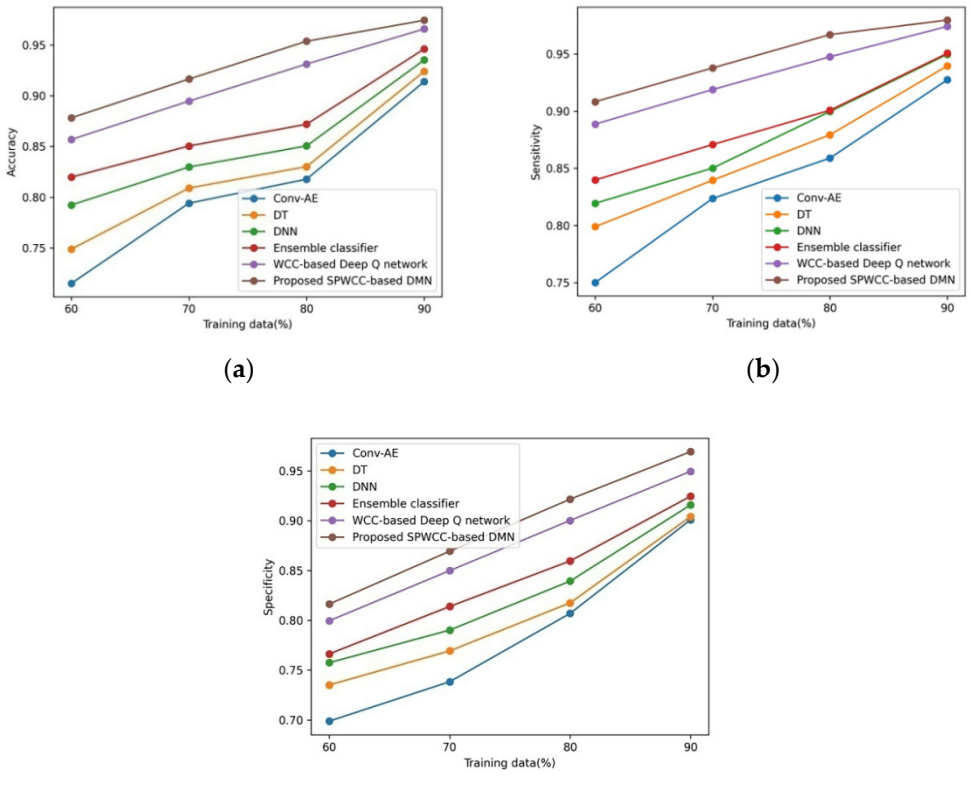
Figure 6 presents the assessment of techniques with a probe attack scenario. The assessment of techniques with accuracy is displayed in Figure 6a. For 90% of data, the accuracy measured by the proposed SPWCC-based DMN was 0.974, which is 6.160%, 5.133%, 4.004%, 2.874%, and 0.821% better than the other existing methods. The assessment of techniques with sensitivity is displayed in Figure 6b. For 60% data, the sensitivity measured by Conv-AE, DT, DNN, Ensemble classifier, and the WCC-based Deep Q network, with respect to the proposed SPWCC-based DMN were 0.750, 0.799, 0.819, 0.840, 0.889, and 0.908. The assessment of techniques with specificity is displayed in Figure 6c. For 90% data, the proposed system had a maximum specificity of 0.969.



(**a**) (**b**)

(**c**)

Figure 5. Assessment of techniques with DoS attack considering (**a**) accuracy, (**b**) sensitivity, and (**c**) specificity.



(**a**) (**b**)

(**c**)

Figure 6. Assessment of techniques with probe attack considering (**a**) accuracy, (**b**) sensitivity, and (**c**) specificity.

5.1.4. Assessment of Other Attacks

Figure 7 presents the assessment of techniques with a probe attack scenario. The assessment of techniques with accuracy is displayed in Figure 7a. For 60% data, the accuracy measured by Conv-AE, DT, DNN, Ensemble classifier, WCC-based Deep Q network, and the proposed SPWCC-based DMN were 0.723, 0.759, 0.810, 0.831, 0.858, and 0.879. The assessment of techniques with sensitivity is displayed in Figure 7b. For 90% data, the proposed method had a maximum sensitivity of 0.979. The assessment of techniques with specificity is displayed in Figure 7c. For 90% data, the proposed SPWCC-based DMN had the maximum specificity of 0.971, which is 11.019%, 7.983%, 6.797%, 5.252%, and 2.162% better than the existing methods, such as Conv-AE, DT, DNN, Ensemble classifier, and the WCC-based Deep Q network.

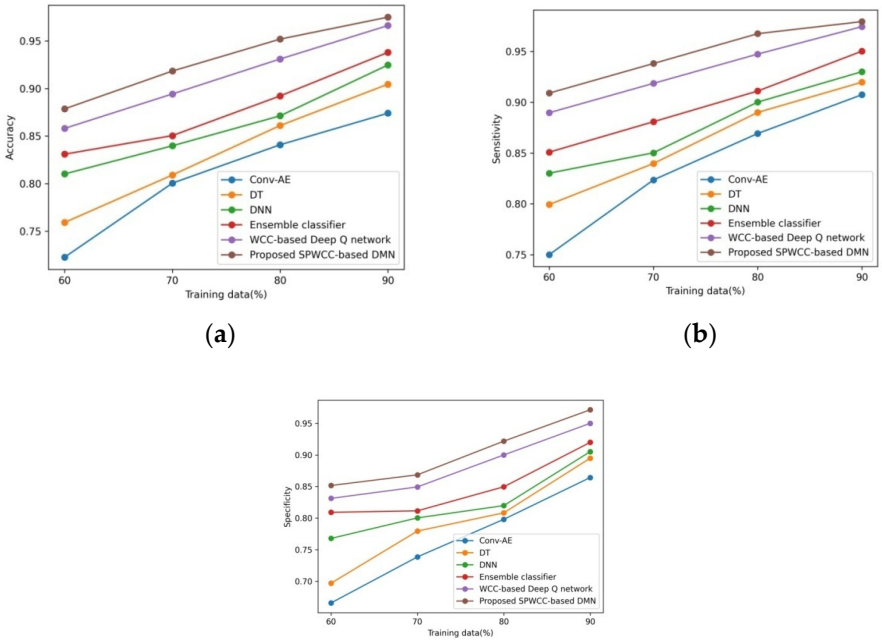


Figure 7. Assessment of techniques with other attacks considering (**a**) accuracy, (**b**) sensitivity, and (**c**) specificity.

5.2. Discussion Based on the Best Performance

Table 1 demonstrates a comparative assessment of techniques using specificity, accuracy, and sensitivity by considering without attack, with DoS attack, with probe attack, and with other attack scenarios. The proposed system obtained maximum accuracy, sensitivity, and specificity at 90% of training data. Considering without attack, the highest accuracy of 0.976 was measured by the proposed SPWCC-based DMN. The highest sensitivity of 0.980 was measured by the proposed SPWCC-based DMN, while the sensitivity evaluated by Conv-AE, DT, DNN, Ensemble classifier, and the WCC-based Deep Q network were 0.937, 0.940, 0.950, 0.954, and 0.974. The highest specificity of 0.970 was measured by the proposed SPWCC-based DMN, which is 6.701%, 5.670%, 4.536%, 4.123%, and 2.061% better than the Conv-AE, DT, DNN, Ensemble classifier, and WCC-based Deep Q network, respectively. Considering with DoS attack, the highest accuracy of 0.973, highest sensitivity of 0.980, and highest specificity of 0.970 were measured by the proposed SPWCC-based DMN. Considering with probe attack, the highest accuracy of 0.974, highest sensitivity of 0.980, and highest specificity of 0.969 were measured by the proposed SPWCC-based DMN. Considering with other attack, the highest accuracy of 0.975, highest sensitivity of 0.979, and highest specificity of 0.971 were measured by the proposed SPWCC-based DMN.

Table 1. Comparative discussion.

Attacks	Metrics	Conv-AE	DT	DNN	Ensemble Classifier	WCC-Based Deep Q Network	Proposed SPWCC-Based DMN
Without attack	Accuracy	0.915	0.925	0.938	0.945	0.966	**0.976**
	Sensitivity	0.937	0.940	0.950	0.954	0.974	**0.980**
	Specificity	0.905	0.915	0.926	0.930	0.950	**0.970**
With DoS attack	Accuracy	0.910	0.924	0.934	0.945	0.966	**0.973**
	Sensitivity	0.927	0.940	0.940	0.950	0.974	**0.980**
	Specificity	0.900	0.910	0.915	0.925	0.950	**0.970**
With probe attack	Accuracy	0.914	0.924	0.935	0.946	0.966	**0.974**
	Sensitivity	0.927	0.940	0.950	0.951	0.974	**0.980**
	Specificity	0.901	0.904	0.916	0.924	0.949	**0.969**
With other attack	Accuracy	0.874	0.904	0.925	0.938	0.966	**0.975**
	Sensitivity	0.907	0.920	0.930	0.950	0.974	**0.979**
	Specificity	0.864	0.895	0.905	0.920	0.950	**0.971**

6. Conclusions

This paper devised a technique for detecting intrusions using the Apache Spark framework. Here, the steps considered for intrusion detection involve data partitioning, feature selection, data augmentation, and intrusion detection. Here, the input data was fed to data partitioning, which was done in the master node. The data partitioning was carried out with the proposed FLIBCM, which was devised by combining the FLICM and Bhattacharya distance. After data partitioning, the selection of features was performed for choosing imperative features. Here, the features were selected using classwise info gain in slave nodes. After feature selection, the augmentation of data was performed to make the data suitable for further processing. Here, data augmentation was performed using the oversampling process in the slave nodes. Thereafter, the network intrusion was detected using the DMN in the master node. The training of the DMN was done using the proposed SPWCC algorithm to select the optimal weights for tuning the DMN. Here, the proposed SPWCC algorithm was devised by combining the WCA, CAViaR model, and SPBO. The proposed SPWCC-based DMN provided improved performance with the highest accuracy of 97.6%, sensitivity of 98%, and specificity of 97%. In the future, other databases can be adapted for checking the feasibility of the model.

Abbreviation

The following table describes the significance of various abbreviations used throughout the article.

HDFS	Hadoop Distributed File System
WDLSTM	Weight-Dropped, Long Short-Term Memory
CNN	Convolutional Neural Network
SQL	Structured Query language
ML	Machine Learning
KNN	K-nearest neighbors
Conv-AE	convolutional-auto encoder
SVMs	Support Vector Machines
BDHDLS	Big Data-based Hierarchical Deep Learning System
PSO	Particle Swarm Optimization
GA	Genetic Algorithm
FLIBCM	fuzzy local information and Bhattacharya-based C-Means clustering
DLS-IDS	Deep Learning Spark Intrusion Detection System
BLSTM	Bidirectional Long Short-term memory
RBF	Radial Basis function

Author Contributions: B.E.B.-M.: Conceptualization, Methodology, Software, Validation, Resources, Formal analysis, writing—Original draft, Writing—Review and Editing. C.W.: Conceptualization, Methodology, Formal analysis, Writing—Review and Editing, Supervision, Project administration. E.K.B.: Software, Validation, Formal analysis. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are openly available in https://www.unb.ca/cic/datasets/nsl.html (accessed on 22 June 2021).

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

References

1. Dahiya, P.; Srivastava, D.K. Network intrusion detection in big dataset using spark. *Procedia Comput. Sci.* **2018**, *132*, 253–262. [CrossRef]
2. Azeroual, O.; Nikiforova, A. Apache spark and MLlib-based intrusion detection system or how the big data technologies can secure the data. *Information* **2022**, *13*, 58. [CrossRef]
3. Abushwereb, M.; Alkasassbeh, M.; Almseidin, M.; Mustafa, M. An accurate IoT intrusion detection framework using Apache Spark. *arXiv* **2022**, arXiv:2203.04347.
4. Ramkumar, M.P.; Bhaskar Reddy, P.V.; Thirukrishna, J.T.; Vidyadhari, C. Intrusion detection in big data using hybrid feature fusion and optimization enabled deep learning based on spark architecture. *Comput. Secur.* **2022**, *116*, 102668.
5. Gupta, G.P.; Kulariya, M. A framework for fast and efficient cyber security network intrusion detection using Apache Spark. *Procedia Comput. Sci.* **2016**, *93*, 824–831. [CrossRef]
6. Mahdy, A.M.S.; Lotfy, K.; El-Bary, A.A. Use of optimal control in studying the dynamical behaviors of fractional financial awareness models. *Soft Comput.* **2022**, *26*, 3401–3409. [CrossRef]
7. Li, R.; Shen, M.; Yu, H.; Li, C.; Duan, P.; Zhu, L. A survey on cyberspace search engines. In *CNCERT: Cyber Security, Proceedings of the China Cyber Security Annual Conference, Beijing, China, 12 August 2020*; Springer: Singapore, 2020; pp. 206–214.
8. Daskevics, A.; Nikiforova, A. IoTSE-based open database vulnerability inspection in three Baltic countries: ShoBEVODSDT sees you. In Proceedings of the 8th international Conference on Internet of Things: Systems, Management and Security (IOTSMS), Gandia, Spain, 6–9 December 2021; pp. 1–8.
9. Faker, O.; Dogdu, E. Intrusion detection using big data and deep learning techniques. In Proceedings of the 2019 ACM Southeast Conference, Kennesaw, GA, USA, 18–20 April 2019; pp. 86–93. [CrossRef]
10. Hafsa, M.; Jemili, F. Comparative study between big data analysis techniques in intrusion detection. *Big Data Cogn. Comput.* **2019**, *3*, 1. [CrossRef]
11. Mahdy, A.M.S.; Youssef, E.S.M. Numerical solution technique for solving isoperimetric variational problems. *Int. J. Mod. Phys. C* **2021**, *32*, 2150002. [CrossRef]
12. Kulariya, M.; Saraf, P.; Ranjan, R.; Gupta, G.P. Performance analysis of network intrusion detection schemes using Apache Spark. In Proceedings of the 2016 International Conference on Communication and Signal Processing (ICCSP), Melmaruvathur, India, 6–8 April 2016; pp. 1973–1977.
13. Veeraiah, N.; Krishna, B.T. Intrusion detection based on piecewise fuzzy C-means clustering and fuzzy naive bayes rule. *Multimed. Res.* **2018**, *1*, 27–32.
14. Yi, Y.; Wu, J.; Xu, W. Incremental SVM based on reserved set for network intrusion detection. *Expert Syst. Appl.* **2011**, *38*, 7698–7707. [CrossRef]
15. Muda, Z.; Yassin, W.; Sulaiman, M.N.; Udzir, N.I. Intrusion detection based on K-Means clustering and Naïve Bayes classification. In Proceedings of the 2011 7th International Conference on Information Technology in Asia, Sarawak, Malaysia, 12–13 July 2011; pp. 1–6.
16. Syarif, A.R.; Gata, W. Intrusion detection system using hybrid binary PSO and K-nearest neighborhood algorithm. In Proceedings of the 11th International Conference on Information & Communication Technology and System (ICTS), Surabaya, Indonesia, 31 October 2017; pp. 181–186.
17. Aslahi-Shahri, B.M.; Rahmani, R.; Chizari, M.; Maralani, A.; Eslami, M.; Golkar, M.J.; Ebrahimi, A. A hybrid method consisting of GA and SVM for intrusion detection system. *Neural Comput. Appl.* **2016**, *27*, 1669–1676. [CrossRef]
18. Aburomman, A.A.; Reaz, M.B.I. A novel SVM-KNN-PSO ensemble method for intrusion detection system. *Appl. Soft Comput.* **2016**, *38*, 360–372. [CrossRef]
19. Zhang, H.; Dai, S.; Li, Y.; Zhang, W. Real-time distributed-random-forest-based network intrusion detection system using Apache Spark. In Proceedings of the 37th IEEE international performance computing and communications conference (IPCCC), Orlando, FL, USA, 17–19 November 2018; pp. 1–7.
20. Kalyani, G.; Lakshmi, A.J. Performance assessment of different classification techniques for intrusion detection. *IOSR J. Comput. Eng. (IOSRJCE)* **2012**, *7*, 25–29. [CrossRef]

21. Chauhan, H.; Kumar, V.; Pundir, S.; Pilli, E.S. A comparative study of classification techniques for intrusion detection. In Proceedings of the International Symposium on Computer and Business Intelligent, New Delhi, India, 24–26 August 2013; pp. 40–43.
22. Khan, M.A.; Kim, J. Toward developing efficient conv-AE-based intrusion detection system using heterogeneous dataset. *Electronics* **2020**, *9*, 1771. [CrossRef]
23. Morfino, V.; Rampone, S. Towards near-real-time intrusion detection for IoT devices using supervised learning and Apache Spark. *Electronics* **2020**, *9*, 444. [CrossRef]
24. Atefinia, R.; Ahmadi, M. Network intrusion detection using multi-architectural modular deep neural network. *J. Supercomput.* **2020**, *77*, 3571–3593. [CrossRef]
25. Mahfouz, A.; Abuhussein, A.; Venugopal, D.; Shiva, S. Ensemble classifiers for network intrusion detection using a novel network attack dataset. *Future Internet* **2020**, *12*, 180. [CrossRef]
26. Hassan, M.M.; Gumaei, A.; Alsanad, A.; Alrubaian, M.; Fortino, G. A hybrid deep learning model for efficient intrusion detection in big data environment. *Inf. Sci.* **2020**, *513*, 386–396. [CrossRef]
27. Zhong, W.; Yu, N.; Ai, C. Applying big data based deep learning system to intrusion detection. *Big Data Min. Anal.* **2020**, *3*, 181–195. [CrossRef]
28. Su, T.; Sun, H.; Zhu, J.; Wang, S.; Li, Y. BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset. *IEEE Access* **2020**, *8*, 29575–29585. [CrossRef]
29. Haggag, M.; Tantawy, M.M.; El-Soudani, M.M. Implementing a deep learning model for intrusion detection on Apache Spark platform. *IEEE Access* **2020**, *8*, 163660–163672. [CrossRef]
30. Ayyagari, M.R.; Kesswani, N.; Kumar, M.; Kumar, K. Intrusion detection techniques in network environment: A systematic review. *Wirel. Netw.* **2021**, *27*, 1269–1285. [CrossRef]
31. Ahmad, Z.; Khan, A.S.; Shiang, C.W.; Abdullah, J.; Ahmad, F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Trans. Emerg. Telecommun. Technol.* **2020**, *32*, e4150. [CrossRef]
32. Krinidis, S.; Chatzis, V. A robust fuzzy local information C-means clustering algorithm. *IEEE Trans. Image Process.* **2010**, *19*, 1328–1337. [CrossRef] [PubMed]
33. Mukherjee, S.; Sharma, N. Intrusion detection using naive bayes classifier with feature reduction. *Procedia Technol.* **2012**, *4*, 119–128. [CrossRef]
34. Sun, W.; Su, F.; Wang, L. Improving deep neural networks with multi-layer maxout networks and a novel initialization method. *Neurocomputing* **2018**, *278*, 34–40. [CrossRef]
35. Eskandar, H.; Sadollah, A.; Bahreininejad, A.; Hamdi, M. Water cycle algorithm—A novel metaheuristic optimization method for solving constrained engineering optimization problems. *Comput. Struct.* **2012**, *110*, 151–166.
36. Engle, R.F.; Manganelli, S. CAViaR: Conditional autoregressive value at risk by regression quantiles. *J. Bus. Econ. Stat.* **2004**, *22*, 367–381. [CrossRef]
37. Das, B.; Mukherjee, V.; Das, D. Student psychology based optimization algorithm: A new population based optimization algorithm for solving optimization problems. *Adv. Eng. Softw.* **2020**, *146*, 102804. [CrossRef]
38. The NSL-KDD Dataset. Available online: https://www.unb.ca/cic/datasets/nsl.html (accessed on 22 June 2021).