electronics

Article

Filter-Based Ensemble Feature Selection and Deep Learning Model for Intrusion Detection in Cloud Computing

C. Kavitha [1], Saravanan M. [2], Thippa Reddy Gadekallu [3,4], Nimala K. [5], Balasubramanian Prabhu Kavin [6,*] and Wen-Cheng Lai [7,8,*]

1 Department of Computer Science and Engineering, Sathyabama Institute of Science and Technology, Chennai 600083, Tamil Nadu, India
2 Department of Networking and Communications, College of Engineering and Technology, SRM Institute of Science and Technology, SRM Nagar, Kattankulathur 603203, Tamil Nadu, India
3 School of Information Technology and Engineering, Vellore Institute of Technology, Vellore 632014, Tamil Nadu, India
4 Department of Electrical and Computer Engineering, Lebanese American University, Byblos 10150, Lebanon
5 Department of Networking and Communications, School of Computing, SRM Institute of Science and Technology, SRM Nagar, Kattankulathur 603203, Tamil Nadu, India
6 Department of Data Science and Business System, SRM Institute of Science and Technology, SRM Nagar, Kattankulathur 603203, Tamil Nadu, India
7 Bachelor Program in Industrial Projects, National Yunlin University of Science and Technology, Douliu 640301, Taiwan
8 Department Electronic Engineering, National Yunlin University of Science and Technology, Douliu 640301, Taiwan
* Correspondence: ceaserkavin@gmail.com (B.P.K.); wenlai@yuntech.edu.tw (W.-C.L.)

Abstract: In recent years, the high improvement in communication, Internet of Things (IoT) and cloud computing have begun complex questioning in security. Based on the development, cyberattacks can be increased since the present security techniques do not give optimal solutions. As a result, the authors of this paper created filter-based ensemble feature selection (FEFS) and employed a deep learning model (DLM) for cloud computing intrusion detection. Initially, the intrusion data were collected from the global datasets of KDDCup-99 and NSL-KDD. The data were utilized for validation of the proposed methodology. The collected database was utilized for feature selection to empower the intrusion prediction. The FEFS is a combination of three feature extraction processes: filter, wrapper and embedded algorithms. Based on the above feature extraction process, the essential features were selected for enabling the training process in the DLM. Finally, the classifier received the chosen features. The DLM is a combination of a recurrent neural network (RNN) and Tasmanian devil optimization (TDO). In the RNN, the optimal weighting parameter is selected with the assistance of the TDO. The proposed technique was implemented in MATLAB, and its effectiveness was assessed using performance metrics including sensitivity, F measure, precision, sensitivity, recall and accuracy. The proposed method was compared with the conventional techniques such as an RNN and deep neural network (DNN) and RNN–genetic algorithm (RNN-GA), respectively.

Keywords: intrusion detection; recurrent neural network; deep learning model; filter-based ensemble feature selection; cloud computing

Citation: Kavitha, C.; M., S.; Gadekallu, T.R.; K., N.; Kavin, B.P.; Lai, W.-C. Filter-Based Ensemble Feature Selection and Deep Learning Model for Intrusion Detection in Cloud Computing. *Electronics* **2023**, *12*, 556. https://doi.org/10.3390/electronics12030556

Academic Editor: Yue Wu

Received: 24 November 2022
Revised: 7 January 2023
Accepted: 16 January 2023
Published: 21 January 2023

1. Introduction

Due to advancements in discrete derivation, many users share similar cloud assets, and the use of the cloud grows as it begins with a distributed ledger. Network traffic may rise as harmful network attacks become more sophisticated and widespread. The primary benefit of distributed computing, which enables efforts to spend investment funds by using benefits on demand through the Internet, is that it enjoys tremendous benefits such as adaptability, speed flexibility and estimated administrations [1]. The cloud's resources and

data correspondence are extremely difficult operations as the number of intrusions steadily rises [2].

As shown in Figure 1, numerous security breaches have recently been discovered in the virtual enterprise layer of distributed computing. To reveal attacks on the cloud foundation, various intrusion detection architecture devices and methods are accessed at the cloud stage in the current scenario. Inadequacy and waste are the consequences of conventional or existing IDS [3]. In most cases, IDS works with the firewall to create a mandatory security setup. A conceptual approach to enhancing the architecture of distributed computing's interrupt detection has emerged as a result of these considerations. It is able to monitor system activity and actually detect network interference [4].

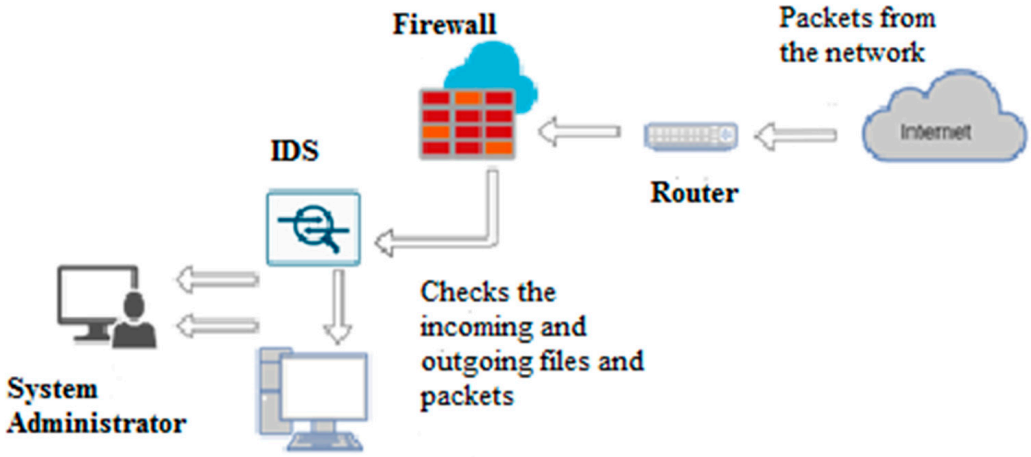

Figure 1. Analytical Model for Network Intrusion Detection System with Deep learning.

This exploration work brings together security attacks related to the public cloud climate and proposed arrangements and ideas in the network [5]. The literature reveals the amount of incalculable insurance against network attacks. Despite the impressive efforts of conservation experts in the last twenty years, the issue of structural conservation [6] remains untested. Concerns about the dependability of information assurance components in this environment have been highlighted in light of attacks on distributed system administrations and frameworks. The system formulated in this review [7,8] (cloud computing, attack, and intrusion authentication: CCAID) is aimed to work on the presentation of an intrusion identification framework that works in distributed computing conditions. Extraordinary advances in correspondence, distributed computing and IoT have opened up fundamental difficulties in security [9]. With these turns of events, cyberattacks are developing extra quickly because current security components do not provide effective arrangements. For a variety of security applications, such as intrusion recognition, several artificial intelligence (AI)-based solutions have been presented [10].

The main contributions and methodology of the research are as follows:

- Development of FEFS and DLM for intrusion detection in the cloud computing environment. Initially, the worldwide datasets of KDDCup-99 and NSL-KDD are used to gather the incursion data.
- The data are utilized for validation of the proposed methodology. The collected database is utilized for feature selection to empower the intrusion prediction. The FEFS is a combination of three feature extraction processes: filter, wrapper and embedded algorithms. Based on the above feature extraction process, the essential features are selected for enabling the training process in the DLM.
- Finally, the classifier receives the chosen features. The DLM is a combination of RNN and TDO. In the RNN, the optimal weighting parameter is selected with the assistance of the TDO.

The additional material for the article is organized as follows. A detailed explanation of the literature review portion is provided in Section 2. Section 3 provides an explanation of the entire process model. In Section 4, the projected technique's results are described. Section 5 of the paper contains a summary.

2. Literature Review

Various techniques are developed by researchers for optimal intrusion detection in cloud computing environments. A few techniques are analyzed and presented in this section.

Xuan-Ha Nguyen et al. [11] have introduced a real guard, a DNN-based network intrusion detection framework (NIDS) that worked directly at adjacent gates to protect IoT gadgets within the enterprise. The strength of the concept was its ability, with a little computational know-how, to precisely and sequentially identify various digital threats. This was accomplished through a lightweight organ extraction component and a productive attack discovery model controlled by deep brain systems.

Adeel Abbas et al. [12] have introduced an ensemble-based intrusion detection model to recognize retaliatory exercises based on predefined attack designs. In the proposed model, a decision tree, naïve Bayes, and logistic regression are informed by offering a ballot classifier in the context of distinguishing the model's exhibition from some current major state-of-the-art techniques. Additionally, the CICIDS2017 dataset was used to examine the suitability of the suggested model. The results represent a huge betterment in precision compared to existing models for parallel and multi-class characterization situations.

Wai Weng Lo et al. [13] have introduced an E-GraphSAGE that allows graph neural networks (GNNs) to acquire some the edge attributes of a graph and topological data for network interference identification in IoT enterprises. For better information, the proposal was the first efficient, effective and widely evaluated approach to apply GNNs to the problem of enterprise intrusion recognition for IoT using stream-based information.

Vinayakumar Ravi et al. [14] have proposed a deep learning-based interval model for network attack location and corporate attack characterization. In addition to using kernel-based principal component analysis (KPCA), which includes a method to identify the best features, the suggested model integrates components from layers of discontinuous models. The best parts of the intermediate models were eventually integrated, and an ensemble meta-classifier was used to finish the characterization. When compared to current methods and widely used AI and deep learning models, experimental research and findings of the proposed technique on multiple benchmark network interference datasets demonstrate that it outperformed them.

Sumegh Tharewal et al. [15] have introduced a near-end process optimization strategy for the industrial Internet of Things of an intrusion detection system in light of a deep reinforcement learning (DRL) computation. This technique integrates the sensing capability of deep learning to effectively detect various types of cyberattacks on the industrial Internet. In this system, the DRL-IDS interference identification framework is based on a component determination technique from the LightGPM perspective, which selects the most attractive list of capabilities from modern Internet of Things information. When deep learning is matched with computations, it differentiates between interruptions.

Pham et al. [16] gave a wrapper FS method that used gain ratio and naïve Bayes to find relevant features. Utilizing two optimization techniques, boosting, and bagging the ensemble, the -e technique, which utilized tree-based approaches as the fundamental classifier, was evaluated. Using the NSL-KDD dataset, the model's classification precision and false alarm (false positive) rate were evaluated. The intrusion model for ML recognition was also developed collaboratively.

Besharati [17] suggested using a host-based intrusion detection system (IDS) for a CC setup that used logistic regression to eliminate variables that did not help find attacks. The ML intrusion detection model used DT, linear discriminant analysis and an ANN as part of ensemble bagging. The F-score, which is the harmonic mean of precision and recall,

classification accuracy, precision (the ratio of the total number of harmful network traffic packets successfully "detected" to the number of harmful network traffic packets accurately "detected" as malignant) and classification accuracy were used to validate it.

Belouch [18] made an effort to improve the FS mechanism by selecting various sets of characteristics based on the type of traffic or attack [18]. To find characteristics unrelated to the desired outcome, the FS method employed an evolutionary search strategy. The two-stage classification procedure was implemented by the ML intrusion detection model using the reduced error pruning tree (REPTree) approach. On the NSL-KDD and UNSW-NB15 datasets, the -e model was evaluated and approved based on classification accuracy as a performance indicator.

According to Vijayanand et al. [19], whereas the FS utilized a genetic algorithm, the ML intrusion detection system made use of multiple SVM classifiers. Identification accuracy, false positive rate and negative predictive value (for each kind of attack) were the -e performance assessment criteria. On the ADFA-LD and CICIDS2017 datasets, the -e models were compared to each other.

According to Aljawarneh et al. [20], network output and a variety of classifiers, such as naïve Bayes, J48, AdaBoost and RT, were also suggested as ways to enhance the FS method. It was decided to go with the classification result with the highest accuracy. The -e ML intrusion detection system was trained and validated using the NSL-KDD dataset.

A wrapper and filter were also utilized by Moustafa and Slay [21]. Using a mining technique, the two datasets (NS KDD and UNSW-NB15) were compressed to produce central points for input variables. Unspoken assumption (EM) grouping, computed relapse and naïve Bayes were the classifications that were communicated in the organization IDS. Evaluation measurements were taken using order accushocking and deceive rate, which is the mathematical normal of the false positive rate and false negative rate.

According to Mogal et al. [22], attributes that were insignificant in light of the false alarm rate were eliminated by making use of the apriori method and the center points of attribute values. Naïve Bayes and logistic regression served as classifiers in the ML model. Using the KDDCup-99 and UNSW-NB15 datasets, classification accuracy and processing time were used as metrics to evaluate the IDS recognition engine.

3. Proposed Intrusion Detection Model

Cloud computing offers customers a novel and maybe simpler way to manage, save and access their information. Perfect resource allocation can be key for cloud computing design to empower the condition of efficient computing origin, affordability and scalability on demand with a quality of service (QoS) assurance. As assaults become more sophisticated in the cloud computing environment, it can be crucial to manage their prevention and to strengthen the safety of the data that are kept and transferred there. Cloud management may be disturbed, and data loss may occur as a result of different attacks such as distributed denial of service (DDoS) and denial of service (DoS). As a result, in this work, FEFS and DLM are built for intrusion detection in the cloud computing environment. The whole design of the intrusion detection technique is depicted in Figure 2.

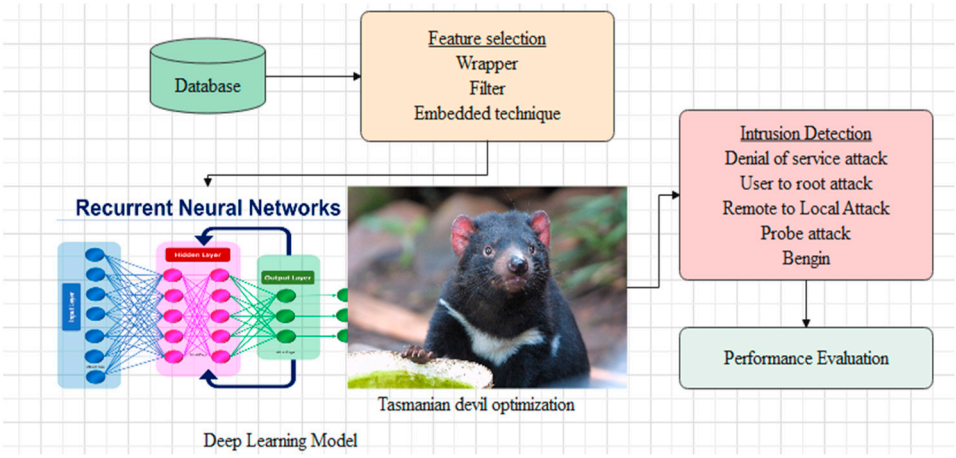

Figure 2. Design of intrusion detection.

3.1. Model Training and Testing Dataset

This section explains how to distinguish training and testing datasets for the examination using the database's stored data. The database is initially gathered from the data source. The wrapper, filter, and embedded technique of FEFS are then taken into consideration when selecting the essential features. In the cloud computing setting, the classifier receives the selected features for intrusion detection. The intruder is identified, and the performance is evaluated on the basis of the training and testing phase. The following section provides a brief explanation of the approach.

3.2. Dataset Description

This intrusion detection model was utilized with datasets of KDDCup-99 [23] and NSL-KDD [24] that were collected from the New Brunswick University public repository. These data were utilized for training and testing of intrusion detection in a cloud computing environment. The various parameters of each feature that define the characteristics of network packets are contained in this database. Numerous features are shown as numerical parameters, while three parameters are shown as non-numeric ones.

It has been extremely challenging to acquire assessment datasets due to the considerable effort security professionals have put into developing data security measures for real-world network settings and computer systems, such as encryption, data anonymization and various data privacy regulations. Consequently, it is difficult to obtain actual network traffic and computer system data. However, in recent decades, a variety of simulated datasets have been developed to address the aforementioned issues. Nearly all of these datasets replicate the most important aspects of actual network traffic.

As a result, by replacing the qualitative feature thread parameters with numerical parameters, the input dataset is improved. In order to select all of the features, the feature selection method, which is applicable to network traffic in cloud environments, used five of them from the KDDCup-99 and NSL-KDD databases. This database contains 148517 packets of information labeled as attack packets or normal packets. The four types of well-known attack packets are as follows:

- **Probe attack:** In this category of outbreak, host ports can be checked for exposed docks that can be secondhand to identify probable vulnerabilities in the cloud computing organization.
- **Denial of service attack:** A type of assault that causes resources or services produced by cloud computing system users to become unavailable.
- **User to root attack:** An attempt to strengthen the base account hijacking those results from a hijacked user explanation.
- **Remote to local attack (R2L):** A system package is sent to a system to target a user explanation and gain access to the computer's contents.

Some of the features of KDDCup-99 and NSL-KDD are given in Table 1.

Table 1. Characteristics of the database.

Feature Count	Feature Name	Description	Type
Feature 1	Hot	The total number of the containers which are hot indicators	Numeric
Feature 2	Urgent	The quantity of the urgent packages as a whole	Numeric
Feature 3	Incorrect break	The total number of incorrect connections' fragments	Numeric
Feature 4	Land	To validate the connection as from a similar host or not	Numeric
Feature 5	Dst-bytes	The number of information bytes sent from the source to the destination	Numeric
Feature 6	Src-bytes	The number of bytes of data transmitted from source to destination	Numeric
Feature 7	Flag	The error or normal status of the connection	String
Feature 8	Service	The kind of network service at the destination	String
Feature 9	Type of the protocol	A packet's leading connection protocol	String
Feature 10	Duration	The length of the connection procedure	Numeric

3.3. Feature Extraction

Feature selection computes the information that will be extracted from the network structure for validating the intrusion discovery classically. The major goal of this research was to improve the intrusion detection model's overall presentation by managing an ideal combination of attributes that effectively influence activity in the original packet flow. To reduce information redundancy and improve performance, feature selection techniques that are semi-supervised, unsupervised and supervised were used. The embedded, wrapper and filter techniques are these feature selection methods. Wrapper and filter techniques that are learned from network data were used in the intrusion detection model with a combination of various methods for selecting features that may have an impact on the accuracy of predicting entities, and a useful set of relevant features that are mostly connected to the information in the target output. The combined feature selection technique, which consists of an embedded technique, a wrapper and a filter, was developed in this study to identify the effective features for identifying an attack in the cloud computing system. The correlation between all characteristics and classes was calculated by a subset evaluator. During feature evaluation, the attribute–class association with the highest correlation is more likely to be chosen. The genetic algorithm determines the significance of each property using this feature evaluation. The rule evaluation stage returns the feature subset with the fewest features if two-element subsets share the same wellness equation. Finally, the model is generated, and attacks are classified using the selected characteristics.

3.3.1. Embedded Algorithms

The ensemble algorithm of AdaBoost [24] was utilized as the feature selection technique for choosing the essential features. This feature extraction technique empowers the classifier training behaviors. To obtain the final results, at the conclusion of each cycle, the boosting framework incorporates the output of inferior classifiers into a stacked total.

3.3.2. Wrapper

In the wrapper method, the random forest algorithm (RF) [25] is a technique which is designed by a combination of outcomes of a large number of decision trees learned with various training clusters. This is developed based on various sub learning clusters that can be generated. In the expansion of the trees, the characteristics can be chosen at random. In the technique, every node can be divided into branches with the consideration of the optimal parameter among the randomly chosen parameter from every node. The derived trees can be achieved by randomly chosen variables. The features are chosen from the input data using the RF algorithm.

3.3.3. Filter

A filer information gain ratio [26] that takes into account the entropy of a particular feature of a session can be used to complete the feature selection. Physics may be used to calculate entropy, which is a measure of how disorderly a parameter is. Entropy is a low-valued parameter for describing features that belong to a particular class. When characteristics link to more than one class, the entropy parameter is larger. Quantifying the value of each feature in identifying the session is made easier with the help of information gain.

3.3.4. Parameter Derivation

The entropy parameter for each feature is taken into account when it is computed. Information gain is calculated as the reduction in the characteristics of the complete pair of impure data items weighted regular entropy. As a result, we were able to determine which characteristics with high data gain parameters are the most essential for detecting assembly data. A parameter that can be used to evaluate a feature's relevance in relation to other characteristics in a dataset is the information gain ratio (IGR). IGR was used as a parameter rather than data gain in this study due to the tendency of information gain to favor features with a large number of distinct parameters.

$$IGR\ (EA, F) = \frac{GAIN(EA, F)}{SPLIT\ INFO\ (EA, F)} \tag{1}$$

Here, F can be defined as the specific features and EA is defined as the dataset feature information. The feature selection process is formulated as follows:

$$GAIN\ (EA, F) = Entropy\ (EA) - \sum_{Ve\ Parameters\ (F)} \frac{|cEA, V|}{|EA|^{*} Entropy}(EA) \tag{2}$$

$$E, V = \left\{ \frac{A \epsilon E}{Value(A, F)} \right\} = V \tag{3}$$

The entropy formula based on Shannon's entropy is presented as follows:

$$EX = -\sum P_I Log_2(P_I) \tag{4}$$

$$Split\ info\ (EA, F) = -|EA, F||EX|V \epsilon Values\ (F) Log_2|EA, F||EA| \tag{5}$$

Based on this formulation, the features are selected from the input data.

3.4. Deep Learning Neural Network (DNN)

For the purpose of identifying an intruder from the cloud computing environment, the selected features are sent to the DNN. RNN and TDO are combined into the DNN. With the help of the TDO, the training process in the RNN is improved. By choosing the best weighting parameter for the network structure in the RNN, the training process is made easier. The suggested classifier is explained in detail in the following section.

An indicator's calculation relies on a constant known as an "input." It is possible for indicators to have one, multiple or none at all. A distinct name is given to each input.

There are true/false, string and numerical input options. To alter the indicators for each classification, the input values are altered. The fact that fundamental modifications to the study's logic could be made without having to update the study's actual code is one of the primary advantages of using inputs. Input values are used by most indicators, but not all of them; please refer to default studies' inputs for additional details. For multiple classifiers, the inputs can be changed before or after the indicator is installed. Figure 3 shows the architecture of deep learning neural network.

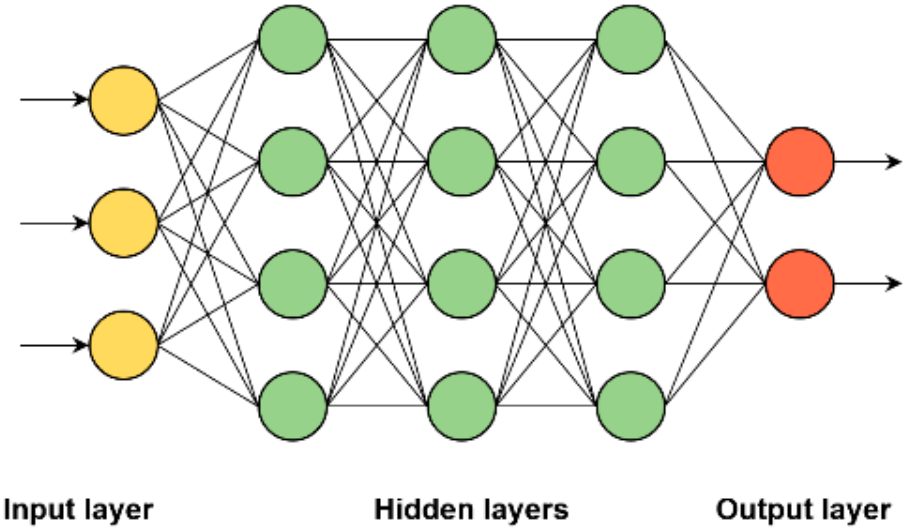

Figure 3. Architecture of Deep Learning Neural Network.

3.4.1. Recurrent Neural Network

Recurrent neural networks, sometimes known as RNNs [27], are a class of neural network used for processing sequence data. Sequence information is described as a close relationship between before and after information that might show the degree of a phenomenon or entity. An RNN has three layers: input, output and hidden. From these layers, the concealed layer performs the effective operation; it is an output layer of the final second and does not contain the output layer at the same time. The RNN processes the sequence information, which is presented in the memory and applied to the output and data information characteristics due to the RNN's depth provide the output and input process, defined as a depth network. The model development of the RNN is presented in the below figures.

Conventional neural networks generate connections only among layers and RNNs connect layers and neurons. Here, the hidden layer is considered for memory storage purposes in the complete network and the RNN can be unfolded, which is utilized for supervised classification training of the recurrent neural network, to develop a directional neural network which saves the previous data and applies them, distinguishing it from conventional feedforward neural networks. For conventional neural networks, learning of samples is highly efficient, with high credibility and more realistic processing of complex feature relationships. RNNs utilize the reverse propagation technique to compute the forward propagation technique, related to their order and time of network propagation. Moreover, in general problem solving, an RNN is not an efficient method because it has limited capability and the hidden layer does not manage the complete model. To solve these problems, in the RNN, the heaviness is improved with the deliberation of TDO. This deep recurrent neural network structure is presented in the below figure.

The proposed RNN has the ability to encode a lot of previous data with the operation of temporal space. In the training process of the RNN, the temporal backpropagation technique is trained to manage the recurrent layer. The error values are generated with the backpropagation technique and are reduced with the assistance of the TDO and this

method propagates in two different ways for the previous period: it transmits and the former layer is forwarded to the current layer [28].

The input layer is connected with the hidden layer which is presented as follows:

$$S_J(T) = F\big(Net_J(T)\big) \tag{6}$$

$$S_J(T) = F\big(Net_J(T)\big) \tag{7}$$

The hidden layer is connected with the output layer, which is presented as follows:

$$Y_k(T) = G(Net_K(T)) \tag{8}$$

$$Net_K(T) = \sum_{J}^{M} S_J(T) W_{KJ} + B_K \tag{9}$$

Here, B_K and B_J can be defined as deviations, G and F can be defined as the output and hidden layer commencement function. The initiation meaning empowers the non-linearity of the complete neural network that empowers the expression capability.

Choosing the Loss and Activation Function

The loss function is used in this article as the network's learning sum function and is shown as follows:

$$C = \frac{1}{2} \sum_{p}^{n} \sum_{k}^{0} (D_{PK} - Y_{PK})^2 \tag{10}$$

The Fitness Function for Training the Network

A neural system was created with more than two layers to compute a large amount of historical data that can be transmitted with an error parameter. The error function is therefore defined as follows:

$$\delta = -\frac{\partial C}{\partial (Net)} \tag{11}$$

$$\delta_{PJ}(T-1) = \sum_{H}^{M} \delta_{PH}(T) U_{HI} F^{'}\big(\delta_{PJ}(T-1)\big) \tag{12}$$

Here, J can be defined as the hidden layer node index with the period of time $T-1$, H is the hidden layer node index with the period of time T.

Weight Matrix Update Computation

In the analysis, the index P of the learning sample cannot be combined. The error parameter of output layer can be presented as follows:

$$E_o(T) = D(T) - Y(T) \tag{13}$$

The weight of output layer is presented as follows:

$$w(T+1) = w(T) + \eta S(T) E_o(T)^t \tag{14}$$

The error function of the output layer to the hidden layer is presented as follows:

$$E_H(T) = D_H\big(E_o(T)^t V, T\big) \tag{15}$$

The equation below is used to find the error vector:

$$d_{HJ}(X, T) = X F^{'}\big(Net_J\big) \tag{16}$$

The weight matrix can be computed based on the below equation:

$$V(T+1) = V(T) + \eta X(T)H(T)^t \tag{17}$$

The cyclic weight matrix can be computed based on the below equation:

$$U(T+1) = U(T) + \eta S(T)H(T)^t \tag{18}$$

The research procedure is utilized to predict intruders, the input for the previous user information and output is a prediction of intruders from the system. In the RNN, the mass structure is optimized with the deliberation of TDO. A detailed explanation of the TDO is presented in the below section.

3.4.2. Tasmanian Devil Optimization

In the RNN, weight optimization is achieved with the assistance of the TDO algorithm. A brief explanation of the TDO is given in this section. The Tasmanian devil is a marsupial, carnivorous member of the Dasyuridae family that dwells in Tasmania, an island state. It is an opportunistic mammal that hunts for prey and eats carrion. It has two feeding strategies. The algorithm computes carrion and feeds on it in the initial technique. Using its second hunting and feeding strategy, it attacks its target [29–31]. In the section below, the mathematical modeling of the feeding method is described.

In this section, the procedure and process of the natural characteristics of Tasmanian devil feeding are presented. The optimization procedure can attain an efficient answer for an optimization problem. The devil can gain access to food due to this behavior, method and personality. It is also an optimization technique with the goal of computing the most effective solution, and it is a nutritional procedure with the possibility of computing the food source. The exploration and exploitation of the problem-solving space as well as the management of the best solution are two key objectives of the optimization process. It defines the exploitation index in the process to find the search space's most effective position. The devil search features are used in computing food sources in diverse spaces. Additionally, the pursuit behavior between the prey and the devil in a constrained area can be equivalent to the exploiting indices in geotargeting with the purpose of convergence to the ideal response. This outlines which mathematical modeling of the devil technique to produce the food supply can be susceptible to creating an optimizer to achieve effective answers to optimization problems.

Stage 1: Initialization

A population approach could be used with the suggested TDO. The first generation of the search agent may be generated at random based on the issue's requirements. Members of TDO's population manage candidate parameters for issue variables associated with their location in the search location. They are problem solvers who are searching for solutions to issues. Therefore, each individual within a population may be represented mathematically as a vector with a parameter count equal to the number of issue variables. On the basis of the following matrix, the set of TDO members is lastly designed.

$$S = \begin{bmatrix} S_1 \\ \cdots \\ S_I \\ \cdots \\ S_N \end{bmatrix}_{N\times M} = \begin{bmatrix} S_{1,1} & \cdots & S_{1,J} & \cdots & S_{1,M} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ S_{I,1} & \cdots & S_{1,J} & \cdots & S_{I,M} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ S_{N,1} & \cdots & S_{N,J} & \cdots & S_{N,M} \end{bmatrix}_{N\times M} \tag{19}$$

where M can be described as a count of variables of specified issues, N can be described as the count of searching devils, $S_{1,J}$ can be defined as the candidate parameter, S_I can be defined as the candidate solution and S can be defined as the population of devils. The primary goal of the problem is calculated by putting each candidate solution into

the parameters of the objective function's variables. Using the vector that is presented as follows, the parameters for the objective function [24] are designed in the results:

$$f = \begin{bmatrix} f_1 \\ \cdots \\ f_I \\ \cdots \\ f_N \end{bmatrix}_{N\times 1} = \begin{bmatrix} f(S_1) \\ \cdots \\ f(S_I) \\ \cdots \\ f(S_N) \end{bmatrix}_{N\times 1} \tag{20}$$

Here, f_I can be defined as the parameter of the objective function achieved through the candidate solution, f can be defined as the vector of parameters of the objective function. In order for the objective function to display the calibre of the potential solutions, the parameters must first be validated. Considered to be the best member of the population, the candidate solution enhances the computation of the objective function's ideal parameter. In each iteration, the population's best individual can be improved in relation to the best parameters. Two methods of Tasmanian devil feeding can be created for the population upgrading process in TDO. For many devils, feeding on carrion or hunting prey may be necessary. In TDO, the probability of selecting either of these techniques is equivalent to 50%. Related to this theory, in every iteration of TDO, every devil can be upgraded based on two techniques.

Method 1: Exploration stage (eating carrion as food)

The devil chooses to use a hunting technique to feed on carrion in the search area. Different kinds of animals live near a devil that hunts many prey, therefore it is unable to eat all prey. Following that, this animal still succeeds in obtaining adequate food from its target. Due to this, the devil decides to attack carrion. In order to calculate carrion, a devil must scan the area of the habitat, using features similar to those used in a technique search to solve problems. This method typically strengthens TDO exploration's capacity for sifting through many search locations to identify the typical ideal spot. The technique for consuming carcasses is currently being developed. Each devil in the TDO design has a spot that can be offered as a carrion space for any remaining population members in the search location. Random selection of the simulation is presented as follows:

$$D_I = S_K,\ I = 1, 2, \ldots, N,\ K\epsilon\{1, 2, \ldots, N | K \neq I\} \tag{21}$$

Here, D_I is carrion chosen by the devil.

A new location for the devil can be computed in the search location based on the selected carrion. The optimal objective parameter for the carrion determines whether the devil will migrate toward the carrion or away from it in this strategy. Simulating the devil's movement style is possible. After calculating the devil's new location in the final stage of the first technique, the location can be accepted, and the goal function's parameter can be set to be as optimal as possible there. The updating process is designed and presented as follows:

$$S_{I,J}^{NEW,\ T1} = \begin{cases} S_{I,J} + U.\left(D_{I,J} - I.S_{I,J}\right) & f_{S_I} < f_I; \\ S_{I,J} + U.\left(D_{I,J} - I.S_{I,J}\right) & Otherwise \end{cases} \tag{22}$$

$$S_I = \begin{cases} S_{I,J}^{NEW,\ T1} & f_I^{NEW,T1} < f_I; \\ S_I & Otherwise \end{cases} \tag{23}$$

Here, I can be defined as a random variable that is 1 or 2, U can be defined as the random variable in interval [0, 1], f_{S_I} can be defined as the objective function parameter of the chosen carrion, $f_I^{NEW,T1}$ is the objective function structure, $S_{I,J}^{NEW,\ T1}$ can be defined as the parameter of the variable, $S_I^{NEW,\ T1}$ can be defined as the novel status of the devil related to the initial technique.

Method 2: Exploitation technique (eating via consuming prey)

The devil's second strategy is to chase and consume prey. There are two phases to the devil's attack qualities. It selects the target and assaults it during the early phase after scanning the area. The second stage comes next, where after controlling the victim, it attacks it to kill it in addition to the original devouring. Designing the beginning stage is comparable to modeling the starting procedure of the carcass that is selected. As a result, the prey is chosen in the early phase and attacked. In the second method, the devil's updating procedure, the position of the surviving population members can be taken as the position of the prey. The following is how the prey selection process is portrayed.

$$P_I = S_K,\ I = 1, 2, \ldots, N,\ K\epsilon\{1, 2, \ldots, N | K \neq I\} \tag{24}$$

Here, P_I is the prey chosen by the devil.

After computing the prey location, a novel location can be computed for the devil. The devil advances toward the chosen prey while computing this unique position, and away from it if the objective function parameter is different. The new location computed for the devil changes the last location and it enhances the parameter of the final function. This last phase of the second technique is designed as follows:

$$S_{I,J}^{NEW,\ T2} = \begin{cases} S_{I,J} + U.\left(D_{I,J} - I.S_{I,J}\right) & f_{P_I} < f_I; \\ S_{I,J} + U.\left(D_{I,J} - I.S_{I,J}\right) & Otherwise \end{cases} \tag{25}$$

$$S_I = \begin{cases} S_{I,J}^{NEW,\ T2} & f_I^{NEW,T2} < f_I; \\ S_I & Otherwise \end{cases} \tag{26}$$

Here, f_{P_I} is the objective function parameter of the chosen carrion, $f_I^{NEW,T1}$ is the objective function limitation, $S_{I,J}^{NEW,\ T1}$ can be defined as the parameter of the variable, $S_I^{NEW,\ T1}$ can be defined as the novel status of the devil related to the initial technique.

The additional phase and the replication of pursuing prey are the key differences between this strategy and the initial one. The pursuit of prey close to the attack site may be comparable to the local search at the search site. The TDO's capacity to converge on the best candidate solutions is typically designed with these devil features. To achieve this chasing procedure, the devil manages the prey and is present in the attack place. In this phase, devil location can be defined as the neighborhood center here, and the prey chasing procedure takes place. The neighborhood radius presents the periods in which the devil manages the prey that is computed. Hence, a novel location based on the chasing procedure in the neighborhood can be computed for the devil. The novel computed location can be acceptable to the devil, and it is given a better parameter for the objective function than its last location. The location updating process is simulated for the devil as follows:

$$U = 0.01\left(1 - \frac{T}{t}\right) \tag{27}$$

$$S_{I,J}^{NEW} = S_{I,J} + (2U - 1).u.S_{I,J} \tag{28}$$

$$S_I = \begin{cases} S_I^{NEW} & f_I^{NEW} < f_I \\ S_I & otherwise \end{cases} \tag{29}$$

Here, f_I^{NEW} is the impartial meaning parameter, $S_{I,J}^{NEW}$ is a parameter, S_I^{NEW} can be defined as the new status of the devil in the neighborhood, t can be defined as the maximum amount of iterations, T can be defined as the iteration number and U can be defined as the neighborhood region of the constituent of operation position. Algorithm 1 is the pseudocode of the Tasmanian Devil Optimization (TDO) which in-volves two different methods, one is Exploration phase, and the other is Exploitation state.

Algorithm 1: Pseudocode of TDO

Initiate TDO
Input the optimization issue data
Initiate the population's size and the amount of iterations
Computation of the objective function and setting up the devil position
For I = 1:N
For T = 1:t
Probability<0.5, If probability = RAND
Method 1: Exploration phase
Choose carrion
Compute new status of devil
Update the devil
Else
Method 2: Exploitation state
Phase 1: Choosing a target and assaulting
Choose the devil's prey
Compute new status
Update the devil
Phase 2: Prey chasing
Update neighborhood radius
Compute new status
Update the devil
End if
End for I = 1:N
Store the optimal solution
End for T = 1:t
Save the optimal solution achieved by TDO
End TDO

Computational Complexity

This section validates the computational simplicity of TDO. This initialization is equivalent to $o(n.M)$. Here, M can be defined as the number of problem variables, n can be defined as the number of devils. The RDO contains an issue-solving procedure with the number of iterations t. The computational complexity of the process of updating population variables on their route to the prey or carcass is similar to $o(n.M.t)$. The second technique's prey chasing process has computing complexity that is similar to $o(N_{x2}.M.t)$. Here, N_{x2} is defined as the number of devils who have utilized the 2nd feeding technique. The total computational complexity of TDO is therefore equal to $o((n.M)((1+t)+(t.N_{x2})))$.

Genetic Algorithm for Optimizing Recurrent Neural Network

Recent efforts have focused on outsourcing machine learning, from Bayesian statistics through feature selection and optimizer tuning. Several programs are available to let users effectively run hundreds of trials. Deep neural network design, similarly, is frequently constructed by professionals by trial and error. This approach, although producing cutting-edge models in a number of disciplines, is night before going to bed. Recent developments in processing capacity have enabled scientists to employ supervised learning and aspects to search for optimal neural networks automatically. We look at how a genetic algorithm (GA) may be used to determine the appropriate window size and number of units and a recurrent neural network with long short-term memory (LSTM) [32,33]. For this goal, we train and evaluate time series prediction models. The GA is implemented using DEAP, a Python module. The methodology's major purpose is to familiarize the reader with utilizing a GA to select relevant ideal settings; hence, just two factors are explored. In addition, practical and theoretical RNN experience is assumed.

4. Performance Evaluation

The effectiveness of the suggested strategy is assessed and supported in this section using the findings from the simulation. A couple of performance measures, including F measure, specificity, sensitivity, and accuracy, can be used to demonstrate the predicted technique's capacity to detect intrusions in the cloud computing environment. In MATLAB, effective assessing of the proposed technique is put into practice. The projected method is put up against the traditional methods, such as RNN, DNN and RNN-GA, in turn. With FEFS and DLM taken into account, the proposed technique is created for identification of intrusion detection in the cloud computing environment. RNN with TDO was created in the DLM for effective intrusion detection. With the help of the TDO, the weight parameter of the RNN is chosen. In the cloud computing environment, the intrusion is finally detected, reducing data theft, and boosting system security. Table 2 lists the technique's implementation parameters.

Table 2. Implementation variables.

S. No	Methods	Description	Parameters
1	Recurrent neural network	Learning rate	0.001
2		Minibatch size	10
3		Loss function	Tanh
4		Type of neurons	Bidirectional LSTM
5		Learning rate	0.01
6		Activation function output layer	Softmax
7	Tasmanian devil optimization	Number of iterations	100
8		Number of populations	50
9		Constant number	0.5
10		Convergence parameter	2
11		Upper limit	10
12		Lower limit	−10

To validate the technique, a confusion matrix is utilized in the system. Two datasets are considered for checking the performance of the system: the KDDCup-99 database and NSL-KDD dataset. This structure with a confusion matrix is given in Figures 4 and 5.

The proposed feature selection and ensemble method's cross-validation performance are depicted by the confusion matrices in Figures 6 and 7. The performance is described in detail in the following section. When compared to other approaches, the findings show that the proposed method significantly improved performance. The table summarizes the findings of a comparison of datasets with full features and other feature selection measures. The suggested ensemble approach outperformed all other independent FS measures, according to the findings, with a high accuracy rate of 0.995, a detection rate of 0.975 and a lower false alarm rate.

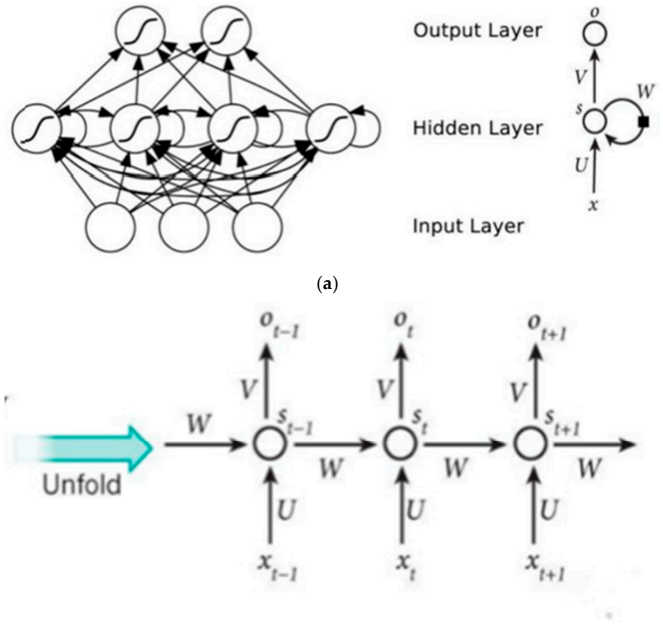

Figure 4. Analysis of (**a**) RNN structure and (**b**) RNN proc ess.

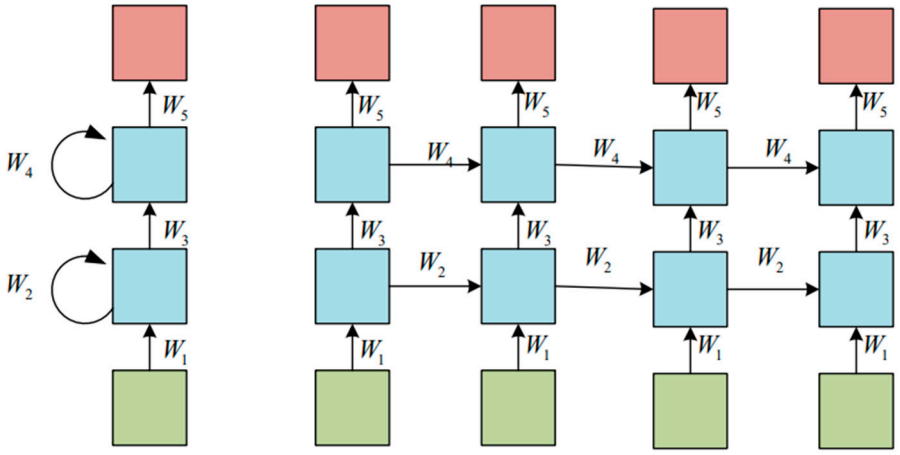

Figure 5. Structure of deep recurrent neural network.

The comparison analysis of accuracy is given in Figure 8. The proposed procedure was compared with the traditional practices such as RNN-GA, RNN and DNN. The proposed technique achieved 0.95% accuracy in intrusion detection in the cloud computing environment. Similarly, the RNN-GA, RNN and DNN achieved 0.92%, 0.89% and 0.87% accuracy in intrusion detection of cloud computing. The proposed system achieved the best parameters of accuracy. The comparison analysis of the F measure is given in Figure 9. The proposed technique was compared with the traditional techniques such as RNN-GA, RNN and DNN. The proposed technique achieved 0.92% F measure in intrusion detection in the cloud computing environment. Similarly, the RNN-GA, RNN and DNN achieved 0.91%, 0.88% and 0.85% F measure in intrusion detection of cloud computing. The proposed technique achieved the best parameters of the F measure. The comparison analysis of precision is given in Figure 10. The proposed technique achieved 0.92% precision in intrusion detection in the cloud computing environment. Similarly, the RNN-GA, RNN and DNN achieved 0.89%, 0.89% and 0.84% precision in intrusion detection of cloud computing. The proposed technique achieved the best parameters of precision.

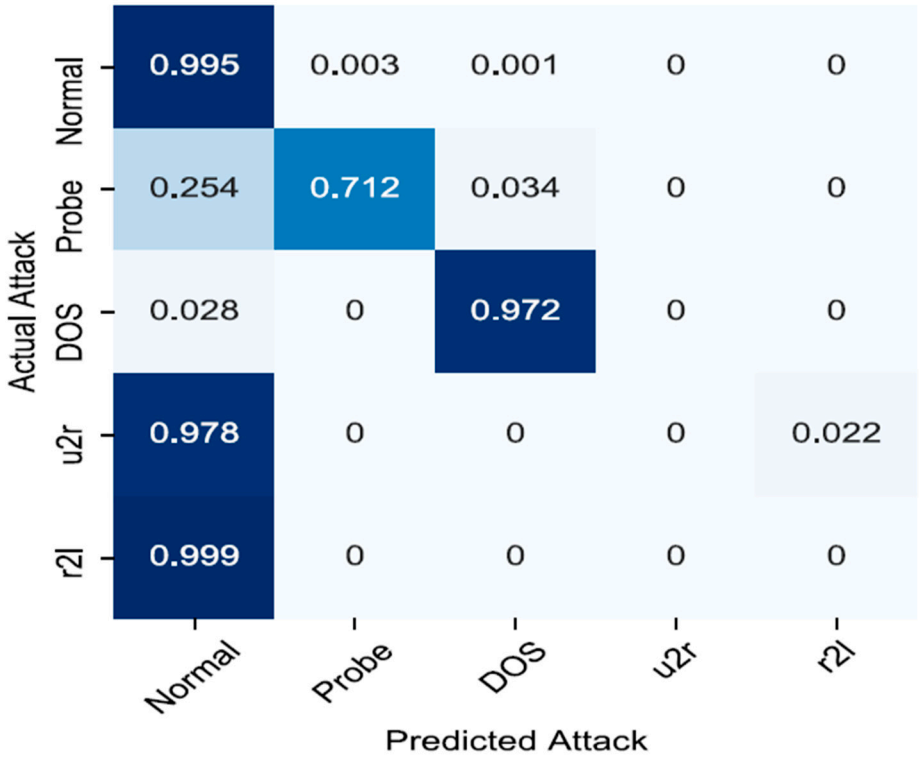

Figure 6. The performance analysis using Confusion Matrix on KDDCup-99 database.

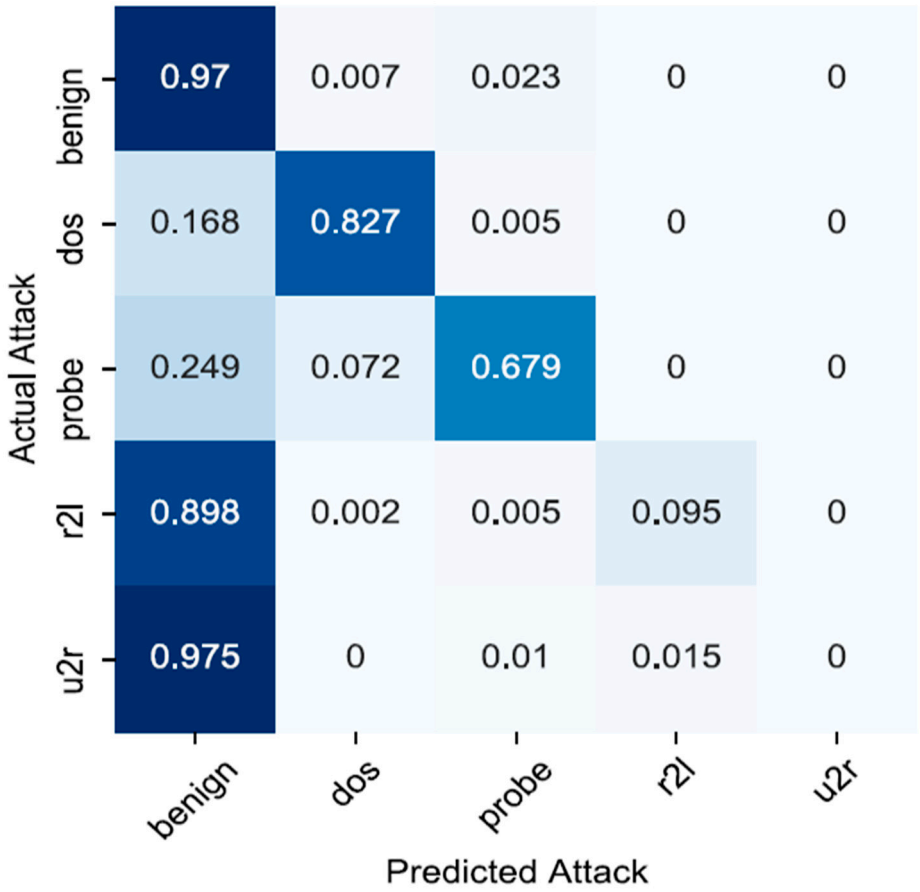

Figure 7. The performance analysis using Confusion Matrix on NSL-KDD dataset.

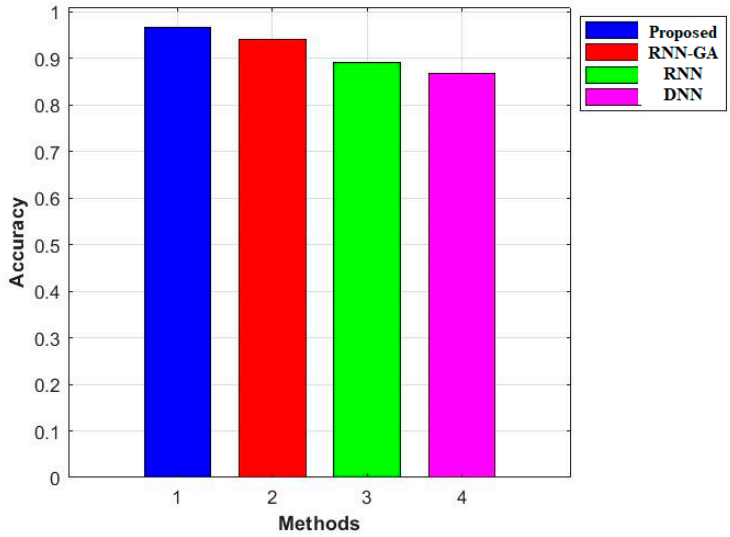

Figure 8. The Accuracy performance of confusion matrix.

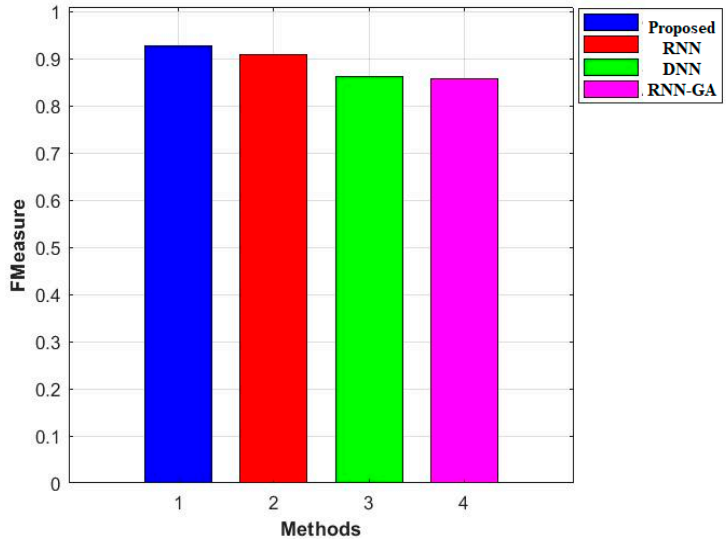

Figure 9. F measure performance of confusion matrix.

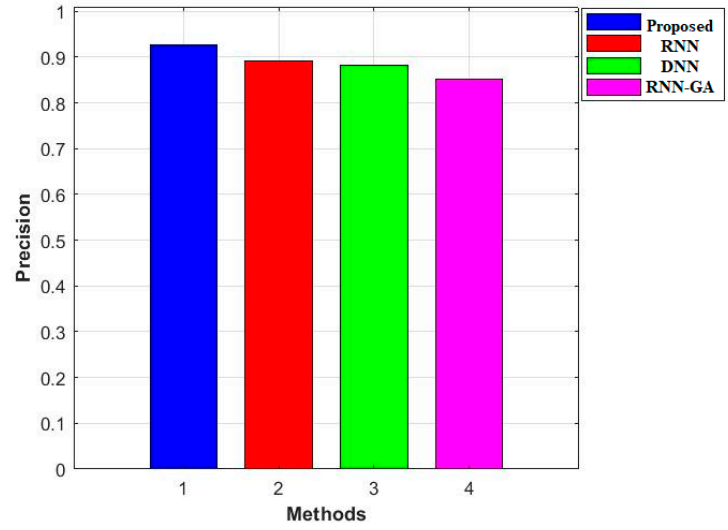

Figure 10. Precision performance of confusion matrix.

Table 3 compares the proposed combination of a recurrent neural network (RNN) and Tasmanian devil optimization (TDO) to that of the current DNN, RNN and RNN-GA.

Table 3. Performance metrics comparison.

	Accuracy	F Measure	Precision	Recall	Sensitivity	Specificity
DNN	0.87	0.85	0.84	0.83	0.81	0.82
RNN	0.89	0.88	0.89	0.88	0.85	0.87
RNN-GA	0.92	0.91	0.89	0.91	0.90	0.92
Proposed	0.95	0.92	0.92	0.93	0.91	0.93

Figure 11 shows the comparison analysis of recall. The proposed technique was compared with the traditional techniques such as RNN-GA, RNN and DNN. The proposed technique achieved 0.93% recall in intrusion detection in the cloud computing environment. Similarly, the RNN-GA, RNN and DNN achieved 0.91%, 0.88% and 0.83% recall in intrusion detection of cloud computing. The proposed technique achieved the best parameters of recall.

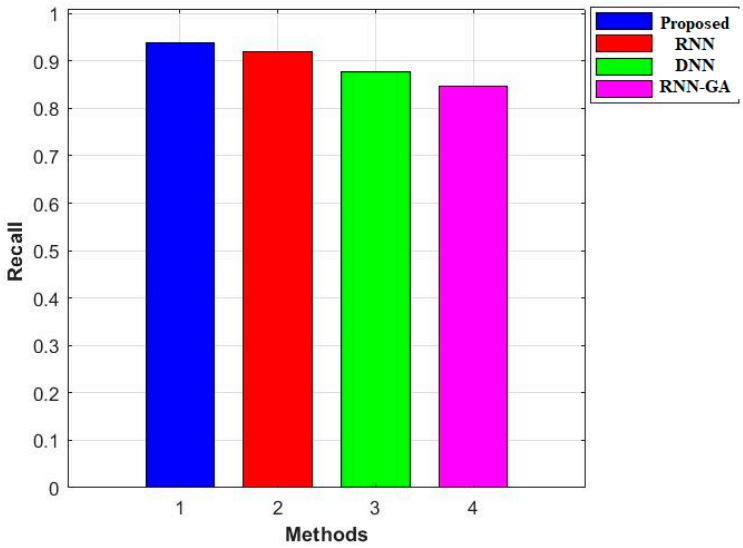

Figure 11. Recall performance of confusion matrices.

Figure 12 shows the comparison analysis of sensitivity. The proposed technique was compared with the traditional techniques such as RNN-GA, RNN and DNN, which showed 0.90%, 0.85% and 0.81% sensitivity in intrusion detection of cloud computing. The proposed technique achieved the best parameters of sensitivity.

Figure 13 shows the comparison analysis of specificity. The proposed technique achieved 0.93% specificity in intrusion detection in the cloud computing environment. Similarly, the RNN-GA, RNN and DNN achieved 0.92%, 0.87% and 0.83% sensitivity in intrusion detection of cloud computing. The proposed technique achieved the best parameters of specificity.

Results of comparing the chosen combination rules' accuracy to the assaults in the NSL-KDD dataset were calculated as in Table 4. For instance, all the combination rules performed well on the benign samples. The average probability, which was 0.95% for routine situations and 0.94% for probe assaults, was used to indicate the highest level of performance accuracy. However, the other attacks, especially U2R, did not have the same remarkable performance accuracy. This may be due to the low number of instances of this attack in the dataset.

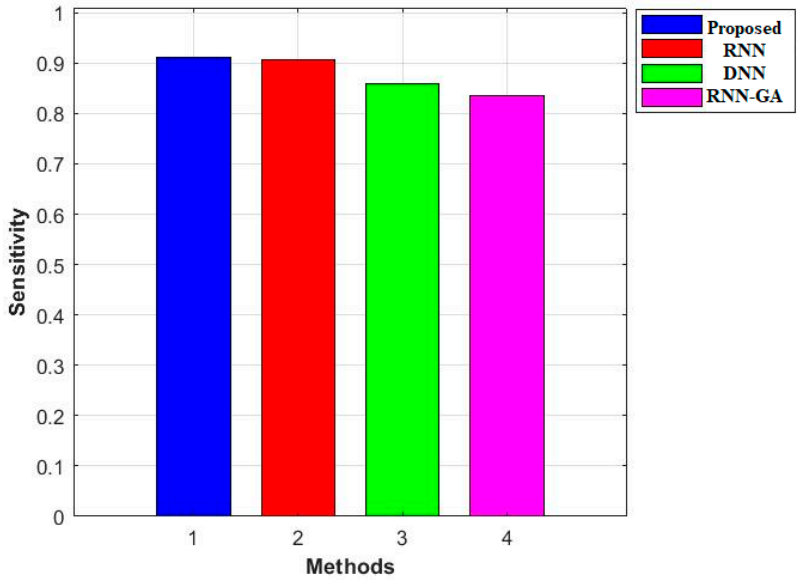

Figure 12. Sensitivity performance of confusion matrix.

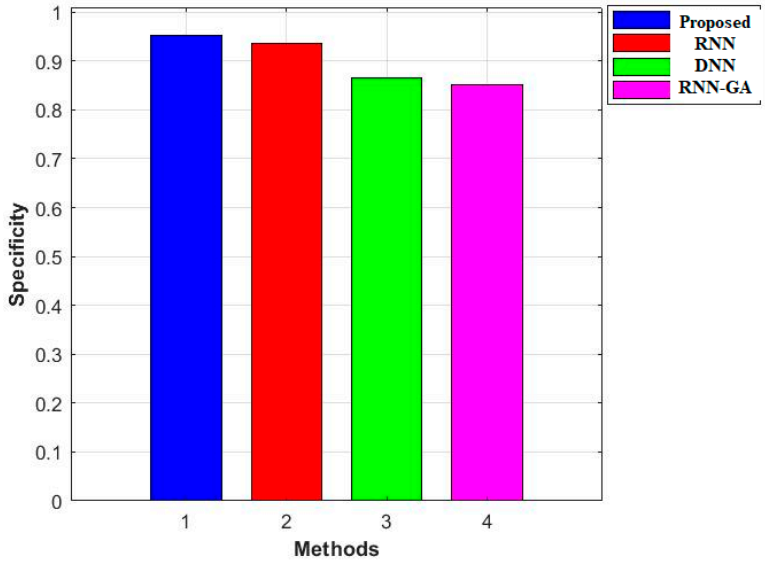

Figure 13. Specificity performance of confusion matrix.

Table 4. Accuracy comparison of different combination rules based on the NSL-KDD test dataset.

Attack Type	Average of Probabilities	Majority Voting	Product of Probability	Minimum Probability	Maximum Probability
Normal	0.95	0.95	0.93	0.92	0.93
DoS	0.94	0.94	0.92	0.91	0.92
Probe	0.94	0.92	0.91	0.89	0.87
R2L	0.93	0.91	0.89	0.87	0.84
U2R	0.69	0.87	0.86	0.76	0.77

Despite the limited number of R2L and U2R instances available, the majority voting rule unexpectedly outperforms the average probabilities on the NSL-KDD dataset. It has outstanding performance accuracy of 0.95%, 0.92% and 0.86% for DoS, R2L and U2R, respectively. Lastly, of all the combination rules, the U2R and R2L attacks had the

lowest accuracy ratings. We established a convincing connection between this and the few instances of these attacks that were included in the aforementioned dataset. Lastly, in contrast to the selected studies and numerous other approaches, by achieving greater overall accuracy, our proposed method has outperformed the other chosen methods. Therefore, it is reasonable to assert that the proposed method outperforms several of the systems used for this assessment in terms of performance accuracy, precision, F measure, recall, sensitivity, and specificity.

5. Conclusions

We developed FEFS and DLM for intrusion detection in a cloud computing environment. Initially, the worldwide datasets of KDDCup-99 and NSL-KDD were used to gather the incursion data. The data were utilized for validation of the proposed methodology. The collected database was utilized for feature selection to empower the intrusion prediction. The main aim of the technology is to prevent cyberattacks that may become more frequent since current security measures might not provide the best answers. The FEFS is a combination of three feature extraction processes: filter, wrapper, and embedded algorithms. Based on the above feature extraction process, the essential features were selected for enabling the training process in the DLM. Finally, the selected features were sent to the classifier. The DLM is a combination of RNN and TDO. In the RNN, the optimal weighting parameter is selected with the assistance of the TDO. Performance metrics including precision, recall, accuracy, sensitivity, specificity, and F measure are used to measure the effectiveness of the suggested method, which is implemented in MATLAB. The proposed approach is contrasted with traditional methods such as RNN-GA, RNN and DNN. According to performance metrics, the suggested strategy produces effective results. Future improvements to accuracy will focus on intrusion detection models in cloud computing environments.

Author Contributions: C.K.: Research concept and methodology, writing—original draft preparation. S.M.: Resources. T.R.G.: Data acquisition. N.K.: Visualization, investigation. B.P.K.: Review, revision and editing. W.-C.L.: Validation and funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been funded by the National Yunlin University of Science and Technology, Douliu.

Data Availability Statement: The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

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

References

1. Shamshirband, S.; Fathi, M.; Chronopoulos, A.T.; Montieri, A.; Palumbo, F.; Pescapè, A. Computational intelligence intrusion detection techniques in mobile cloud computing environments: Review, taxonomy, and open research issues. *J. Inf. Secur. Appl.* **2020**, *55*, 102582. [CrossRef]
2. Jaber, A.N.; Rehman, S.U. FCM–SVM based intrusion detection system for cloud computing environment. *Clust. Comput.* **2020**, *23*, 3221–3231. [CrossRef]
3. Wang, W.; Du, X.; Shan, D.; Qin, R.; Wang, N. Cloud Intrusion Detection Method Based on Stacked Contractive Auto-Encoder and Support Vector Machine. *IEEE Trans. Cloud Comput.* **2020**, *10*, 1634–1646. [CrossRef]
4. Shin, S.; Gu, G. CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds? In Proceedings of the IEEE International Conference on Network Protocols (ICNP), Austin, TX, USA, 30 October–2 November 2012; pp. 1–6. [CrossRef]
5. Lin, F.; Zhou, Y.; An, X.; You, I.; Choo, K.-K.R. Fair Resource Allocation in an Intrusion-Detection System for Edge Computing: Ensuring the Security of Internet of Things Devices. *IEEE Consum. Electron. Mag.* **2018**, *7*, 45–50. [CrossRef]
6. Deshpande, P.; Sharma, S.C.; Peddoju, S.K.; Junaid, S. HIDS: A host-based intrusion detection system for cloud computing environment. *Int. J. Syst. Assur. Eng. Manag.* **2018**, *9*, 567–576. [CrossRef]
7. Nathiya, T.; Suseendran, G. An Effective Hybrid Intrusion Detection System for Use in Security Monitoring in the Virtual Network Layer of Cloud Computing Technology. In *Data Management, Analytics, and Innovation*; Springer: Singapore, 2019; pp. 483–497.
8. Patil, R.; Dudeja, H.; Modi, C. Designing an efficient security framework for detecting intrusions in virtual network of cloud computing. *Comput. Secur.* **2019**, *85*, 402–422. [CrossRef]

9. Truong, T.C.; Zelinka, I.; Plucar, J.; Čandík, M.; Šulc, V. Artificial intelligence and cybersecurity: Past, presence, and future. In *Artificial Intelligence and Evolutionary Computations in Engineering Systems*; Springer: Singapore, 2020; pp. 351–363.
10. Ghosh, P.; Karmakar, A.; Sharma, J.; Phadikar, S. CS-PSO based intrusion detection system in cloud environment. In *Emerging Technologies in Data Mining and Information Security*; Springer: Singapore, 2019; pp. 261–269.
11. Nguyen, X.-H.; Nguyen, X.-D.; Huynh, H.-H.; Le, K.-H. Realguard: A Lightweight Network Intrusion Detection System for IoT Gateways. *Sensors* **2022**, *22*, 432. [CrossRef]
12. Abbas, A.; Khan, M.A.; Latif, S.; Ajaz, M.; Shah, A.A.; Ahmad, J. A New Ensemble-Based Intrusion Detection System for Internet of Things. *Arab. J. Sci. Eng.* **2022**, *47*, 1805–1819. [CrossRef]
13. Lo, W.W.; Layeghy, S.; Sarhan, M.; Gallagher, M.; Portmann, M. E-GraphSAGE: A Graph Neural Network based Intrusion Detection System for IoT. In Proceedings of the NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 25–29 April 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–9.
14. Ravi, V.; Chaganti, R.; Alazab, M. Recurrent deep learning-based feature fusion ensemble meta-classifier approach for intelligent network intrusion detection system. *Comput. Electr. Eng.* **2022**, *102*, 108156. [CrossRef]
15. Derhab, A.; Aldweesh, A.; Emam, A.Z.; Khan, F.A. Intrusion Detection System for Internet of Things Based on Temporal Convolution Neural Network and Efficient Feature Engineering. *Wirel. Commun. Mob. Comput.* **2020**, *2020*, 6689134. [CrossRef]
16. Pham, N.T.; Foo, E.; Suriadi, S.; Jeffrey, H.; Lahza, H.F.M. Improving performance of intrusion detection system using ensemble methods and feature selection. In Proceedings of the Australasian Computer Science Week Multiconference, Brisband, QLD, Australia, 29–31 January 2018; pp. 1–6.
17. Besharati, E.; Naderan, M.; Namjoo, E. LR-HIDS: Logistic regression host-based intrusion detection system for cloud environments. *J. Ambient. Intell. Humaniz. Comput.* **2018**, *10*, 3669–3692. [CrossRef]
18. Belouch, M.; El Hadaj, S.; Idhammad, M. A Two-Stage Classifier Approach using RepTree Algorithm for Network Intrusion Detection. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*, 389–394. [CrossRef]
19. Vijayanand, R.; Devaraj, D.; Kannapiran, B. Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithmbased feature selection. *Comput. Secur.* **2018**, *77*, 304–314. [CrossRef]
20. Aljawarneh, S.; Aldwairi, M.; Yassein, M.B. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *J. Comput. Sci.* **2018**, *25*, 152–160. [CrossRef]
21. Moustafa, N.; Slay, J. A hybrid feature selection for network intrusion detection systems: Central points. *arXiv* **2017**, arXiv:1707.05505.
22. Mogal, D.G.; Ghungrad, S.; Bhusare, B.B. Nids using machine learning classifiers on unsw-nb15 and kddcup99 datasets. *Int. J. Adv. Res. Comput. Commun. Eng.* **2017**, *6*, 533–537. [CrossRef]
23. Available online: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed on 20 November 2022).
24. Available online: https://www.unb.ca/cic/datasets/nsl.html (accessed on 20 November 2022).
25. Aljawarneh, S.; Aldwairi, M.; Yassein, M.B. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *J. Comput. Sci.* **2018**, *25*, 152–160. [CrossRef]
26. Çavuşoğlu, Ü. A new hybrid approach for intrusion detection using machine learning methods. *Appl. Intell.* **2019**, *49*, 2735–2761. [CrossRef]
27. Krishnaveni, S.; Vigneshwar, P.; Kishore, S.; Jothi, B.; Sivamohan, S. Anomaly-Based Intrusion Detection System Using Support Vector Machine. In *Advances in Intelligent Systems and Computing*; Springer: Berlin, Germany, 2020; pp. 723–731. [CrossRef]
28. Shang, K.; Chen, Z.; Liu, Z.; Song, L.; Zheng, W.; Yang, B.; Liu, S.; Yin, L. Haze Prediction Model Using Deep Recurrent Neural Network. *Atmosphere* **2021**, *12*, 1625. [CrossRef]
29. Fan, C.; Wang, J.; Gang, W.; Li, S. Assessment of deep recurrent neural network-based strategies for short-term building energy predictions. *Appl. Energy* **2019**, *236*, 700–710. [CrossRef]
30. Dehghani, M.; Hubalovsky, S.; Trojovsky, P. Tasmanian Devil Optimization: A New Bio-Inspired Optimization Algorithm for Solving Optimization Algorithm. *IEEE Access* **2022**, *10*, 19599–19620. [CrossRef]
31. Rout, T.M.; Baker, C.M.; Huxtable, S.; Wintle, B.A. Monitoring, imperfect detection, and risk optimization of a Tasmanian devil insurance population. *Conserv. Biol.* **2018**, *32*, 267–275. [CrossRef] [PubMed]
32. Rani, S.; Babbar, H.; Srivastava, G.; Gadekallu, T.R.; Dhiman, G. Security Framework for Internet of Things based Software Defined Networks using Blockchain. *IEEE Internet Things J.* **2022**. [CrossRef]
33. Han, Z.; Yang, Y.; Wang, W.; Zhou, L.; Gadekallu, T.R.; Alazab, M.; Gope, P.; Su, C. RSSI Map-Based Trajectory Design for UGV Against Malicious Radio Source: A Reinforcement Learning Approach. *IEEE Trans. Intell. Transp. Syst.* **2022**. [CrossRef]