

A novel intrusion detection system in cloud infrastructure using deep learning technique

Dharani Kumar Talapula *

Adarsh Kumar †

School of Computer Science

University of Petroleum & Energy Studies (UPES)

Bidholi

Dehradun 248007

Uttarakhand

India

Kiran Kumar Ravulakollu §

School of Technology

Woxsen University

Hyderabad 502345

Telangana

India

Manoj Kumar ‡

Faculty of Engineering and Information Sciences

University of Wollongong in Dubai

Dubai

U.A.E.

Abstract

One of the business strategies for selling computer resources with services and technology for better use of computing infrastructures is Cloud computing (CC). Nowadays, every IT company prefers cloud computing because it provides consumers with flexible, pay-per-use services. Due to its open and distributed structure, which is susceptible to attackers, thereby, privacy and security is a key obstacle to its sustainability. The most prevalent approach for detecting assaults on the cloud is known to be Intrusion Detection System (IDS). This article aims to propose a novel intrusion pattern detection system (IPDS)

* E-mail: tdharani@rediff.com (Corresponding Author)

† E-mail: adarsh.kumar@ddn.upes.ac.in

§ E-mail: kiran007.r@gmail.com

‡ E-mail: manojkumar@uowdubai.ac.ae

in cloud computing that includes three stages: (1) pre-processing, (2) feature extraction, and (3) classification. At first, pre-processing is performed on the input data via Z-score normalization and then feature extraction is performed along with statistical and higher-order statistical features. Subsequently, the extracted features are given to the classification phases that use the Optimized Quantum Neural Network (QNN) classifier. The hidden neuron optimization is performed by Cubic Chaotic Map integrated Cat and Mouse Based Optimization (CC-CMBO) Algorithm to make the classification more exact. Finally, the results of the proposed work are assessed to those of standard systems with respect to various measures.

Subject Classification: Primary 93A30, Secondary 49K15.

Keywords: Cloud computing, Classification, QNN, Intrusion detection, Optimization.

1. Introduction

Cloud is a specialized IT infrastructure for supplying scalable & measurable IT resources remotely. Depending on the client's requirements and usage, cloud computing [1] may deliver platform, software, & infrastructure-as-a-service approaches. Cloud computing necessitates virtualizing storage resources and related applications. A hypervisor inspector controls the sharing of hardware platforms and the general operation of the OS in the cloud environment. An intrusion describes an effort to compromise the CIA of a computer, network, or broader environment such as the cloud. IDS comprises a number of components that are used to detect and identify malicious activity. Signature-based & anomaly-based [4] [5] are the two types of IDSs. Signature-based IDS identifies attacks & compares captured patterns to original patterns, while anomaly-based IDS discovers novel forms of intrusions [6]. This cloud environment includes SaaS, PaaS, and IaaS service types. Again, these have three models: community cloud, hybrid cloud, private cloud, and public cloud.

Considering its open nature & volume of traffic makes hackers increasingly appealing. For example, DDoS attacks are common cybercrime following data theft. Flood attacks in UDP or TCP could quickly deplete the cloud's resources, take the majority of its BW, as well as destroy a whole cloud project. Due to these security threats, an efficient intrusion system must be developed & implemented to protect the cloud against newly emerging assaults, also known as zero-day attacks. ANN, SVM, FL, GA, and association rule mining are among the current soft computing approaches. These methods improve the accuracy & efficacy

of anomaly detection / signature-based intrusion detection systems [10] [11]. ANN intrusion systems are used frequently due to their capacity to take insufficient data and identify it as invasive or normal. Various forms of ANN are adapted by IDS [12]. Multi-layer feed forward NN, Multi-layer preceptor, & backpropagation are the most often utilized ANN varieties. For unorganized network data, an ANN-based IDS is a suitable option. To be extremely successful, an IDS often takes a significant amount of time and multiple training samples; it is a key disadvantage of the devices. These global heuristic search strategies choose network characteristics or discover the best parameters to employ other approaches, thereby boosting the IDS's effectiveness. But these IDS [7] produces false alerts and does not employ adequate criteria or parameters to assess threat with great effect. These two most typical problems are of old techniques and thus exacerbate the problem of overuse. However, numerous solutions based on optimization methods inspired by biological evolution, including GA, PSO, HSA, & ABC, have been explored to tackle the ANN difficulties over N-IDS cloud computing [14].

Authors of the current research have elaborately explored state-of-the-art literature on the security threat analysis over the cloud services and Anomaly detection methods that could be used by cloud computing [8] [9] to identify and mitigate unknown threats. The vast events that occurred throughout the regular cloud operation make the task of recognizing & monitoring the attacks cumbersome, which is the critical challenge in applying the strategies. Cloud framework needs a quick technique to detect the intrusions but focusing solely on ANN-based IDS [13] is an insufficient approach. These challenges need to be effectively and efficiently addressed to detect the intrusion pattern detection

This research work implied to propose a new Intrusion pattern detection system that intends to use soft Computing capability to deal with both true and uncertain data which makes it ideal for identifying intrusions in multi-dimensional data containing both spatial, time series, and multiple files of data. As part of preprocessing z-score normalization is performed to formalize scale adjustment. In the second stage after pre-processing higher-order statistical features are extracted to be fed into QNN based classification, which is of the third stage and subsequently, the system is optimized using Cubic Chaotic Map integrated Cat and Mouse Based Optimization technique.

Contributions: The significant contributions made by this work are as follows.

- To explore the various Intrusion Detection System (IDS) presently used across a cloud computing environment.
- To suggest a novel approach for the intrusion pattern detection system (IPDS) in a cloud computing environment.
- To Implement the Cubic Chaotic Map integrated Cat and Mouse Based Optimization (CC-CMBO) Algorithm for tuning the optimal hidden neurons in QNN.
- To verify and validate the proposed method using simulated tests using different performance parameters.

A review of Institution detection systems (IDS) in the cloud is discussed in section 2. Section 3 portrays the system model of an intrusion pattern detection system (IPDS) in cloud architecture. Section 4 illustrates the pre-processing & feature extraction method. Section 5 depicts the Classification via QNN Classifier. Optimal Hidden Neuron Tuning Using a Cubic Chaotic Map Integrated Cat and Mouse Based Optimization Algorithm is specified in section 6. The outcomes & discussion were depicted in section 7. The conclusion is mentioned in section 8.

2. Motivation

This section gives an overview of some of the previous works as well as the obstacles encountered in existing approaches, which are used to inspire the creation of more advanced and more efficient procedures.

2.1 Literature Review

In 2021, Thilagam et. al. [16] established IDS using the ALO method, and a novel optimized custom RC-NN was suggested for attack detection. CNN was hybridized with LSTM in this technique. Consequently, every attack discovered at the network layer of the cloud was correctly categorized. The experiment results demonstrated that the IDS classification algorithm has presented better accuracy, improving the error rate/detection rate. Finally, the suggested method was compared to several conventional systems with respect to DARPA IDS assessment Data Sets and the CSE-CIC-IDS2018 dataset [38].

In 2021, Srilatha et. al. [17] presented a unique cloud- IDS using a hybrid of KFCM and an OT2FNN. Kernel FCMs have already been suggested for clustering. Following the cluster's creation, attacks such as DoS, probing, R2L, & U2R were discovered in the surroundings. To

accomplish this, they use the LOA for weight optimization to choose the ideal T2FNN parameters. The suggested IDS identifies the intrusion & allows normal data to maintain in the cloud. The suggested IDS method outperformed conventional IDS systems based on accuracy, recall, & F-measure, according to simulation findings on the NSL-KDD dataset.

In 2021, Mayuranathan et. al. [19] presented an effective feature subset selection-based classification system for detecting DDoS attacks in this study. In order to identify an IDS DDoS attack, the optimal sets of features with the highest detection rate were chosen using the RHS optimization technique. A DL-based classification algorithm utilizing RBM to identify DDoS was deployed once to select the characteristics. Added a collection of seven more levels to the RBM's visible & hidden layers to increase the rate of detection for DDoS attacks. Further, the RHS-RBM system has a high kappa value, sensitivity, F-score, accuracy, & specificity according to the experimental observations.

In 2020, Jaber et. al. [20] offered a novel detection mechanism in the cloud computing environment that integrates an FCM clustering technique with SVM to boost detection accuracy in this research. Various SVM modules were also trained in the 2nd phase based on the allotted cluster values. Tests were conducted using the NSLKDD dataset. The new hybrid FCM-SVM mechanism's research findings have revealed that the suggested method can identify abnormalities with improved detection accuracy and low false alarm rates compared to conventional methodologies.

In 2021, Xue, L et al.[21] has proposed a Quantum classification algorithm to identify theft activity detection by classifying the electricity dataset. The conclusion drawn from the study signifies an improvement in prediction accuracy with an improvement in the number of training epochs.

3. Intrusion Pattern Detection Model in Cloud

Cloud computing is the platform to facilitate ubiquitous and suitable access to an on-demand network to a shared pool of computational services, which contains networks, applications, storage, & servers. There are two kinds of security concerns in Cloud networks: internal attackers & external attackers. Furthermore, the client-side insider attacker is a malicious user, while the provider-side insider attacker is the Cloud provider. Additionally, external attackers in the system conduct DDoS/DoS attacks to disrupt the system's availability and cloud resources. They

may determine access to one another's VMs by learning authentication information. VM-to-host machine & VM-to-VM attacks are two popular types of attacks. More VMs in the cloud results in increased network traffic. It's challenging to identify dispersed attacks on the cloud for its virtualized & distributed nature.

3.1 *Proposed framework for Intrusion Pattern detection*

This work presents a new intrusion detection-based Cloud Computing system with three stages: (1) pre-processing, (2) feature extraction, & (3) classification. CSE-CIC-IDS2018 dataset [38] is considered. Initially, Z-score normalization will be performed to pre-process the data. Subsequently, extracting the statistical and higher-order statistical features from the pre-processed data. An Optimized QNN is used to classify the retrieved features. Here, the CC-CMBO model is introduced to optimize the hidden neuron to ensure the classification's efficiency. The proposed framework is depicted in Figure 1

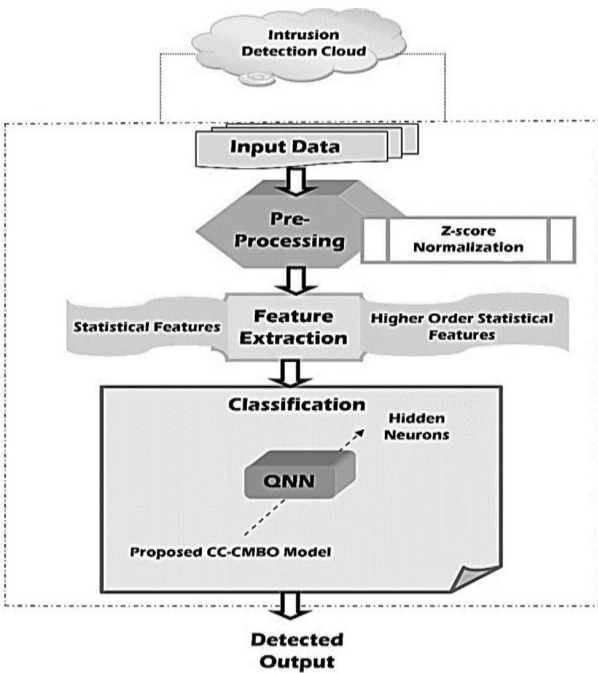


Figure 1
The architecture of proposed framework.

4. Proposed Framework for Pre-Processing and Feature extraction

4.1 Pre-processing

Data cleaning is performed in the pre-processing stage. During the preparation phase, duplicated and unnecessary data is eliminated. We are considering CSE-CIC-IDS2018 dataset to predict the presence of attackers. CIC-IDS2018 is a comparable big dataset publicly available and covers a wide range of attack types. This data set is modern and scalable and is not an entirely new project but has been updated recently. CIC-IDS2018 has been prepared from comparably more extensive networks of simulated client targets and attack machines. System training on this vast and scaled public dataset will benefit in achieving the detection rate.

Initially, the acquired dataset undergoes Z-score normalization to pre-process the data. The scales of the characteristics are adjusted in Data Normalization to provide a standard scale of measurement. The amount of SDs away from the mean is represented by the Z-score, a scaling variation. The Z-score [22] guarantees that the feature distributions have a mean of 0 and an SD of 1. The formula for computing a point's Z-score is shown in Eq. (1).

$$y' = (y - \mu) / \sigma \quad (1)$$

Where y specifies the original value, μ represents the mean, & σ denotes the SD.

4.2 Feature Extraction

In addition, the pretreated data are submitted to the feature extraction step to extract the features.

- Statistical features

- (a) *Mean*: It is the process of dividing the entire values by the total number of entire values [22].

$$\mu = \frac{1}{m} \sum_{z=1}^m R_z \quad (2)$$

In Eq. (2), R denotes observed value m refers to observed values count, as well as μ is the mean

- (b) *Median*: It arranges a dataset's centre values in rising order [22]. If there are two values in the centre of a dataset, the mean of the two centre values is regarded as median.

$$\text{Median} = \begin{cases} R\left(\frac{m}{2}\right) & \text{if } m \text{ is odd} \\ \frac{R\left(\frac{m-1}{2}\right) + R\left(\frac{m+1}{2}\right)}{2} & \text{if } m \text{ is even} \end{cases} \quad (3)$$

- (c) *SD*: SD measures the amount of variance in the dispersion values set. The lesser SD [22] suggests results close to the mean value, whereas the more significant SD indicates the values in a broader range. The SD is provided in Eq. (4). SD is referred to as a σ .

$$\sigma = \sqrt{\frac{1}{m-1} \sum_{z=1}^m (R_z - \mu)^2} \quad (4)$$

- (d) *Variance*: The possibility of squared variation of random variables from the population sample/mean [22]. Moreover, it is specified as per Eq. (5).

$$\text{Variance} = \frac{\sum (R_z - \mu)^2}{m-1} \quad (5)$$

The statistical features *SF* is given in Eq. (6). Here, *Max* refers to the maximum value and *Min* denotes the minimum value.

$$SF = \mu + \text{Median} + \sigma + \text{varinace} + \text{Max} + \text{Min} \quad (6)$$

- Higher-order statistical features

- (a) *Skewness*: It's a measurement of symmetry vs. lack of homogeneity [22]. A distribution/data set is considered symmetrical if it is symmetric to the left/right of the centre point. Moreover, the skewness is determined by Eq. (7).

$$\text{skewness} = \frac{\sum_{z=1}^m (R_z - \mu)^3 / m}{\sigma^3} \quad (7)$$

In Eq. (7), $R_z = R_1, R_2, \dots, R_m$, m indicates the data points count, μ refers to mean value, and SD indicated as σ . Furthermore, σ is calculated as m in the denominator relatively to $m - 1$, with respect to the calculated skewness. Further, the skewness is 0 for normal distribution and for any symmetrical data similar to 0.

- (b) *Kurtosis*: It's a measure used to determine whether data is light or heavy-tailed and linked to the normal distribution [22]. Smaller tails, also known as outliers, are marginally uncommon with reduced kurtosis in datasets. Large tails or outliers are more likely to be found in datasets with a greater kurtosis. The SD is calculated by evaluating the denominator value for calculating the kurtosis. For univariate data $R_1, R_2, \dots, R_m$, the kurtosis is calculated in Eq. (8).

$$kurtosis = \frac{\sum_{z=1}^m (R_z - \mu)^4 / m}{\sigma^4} \quad (8)$$

- (c) *Percentile*: A percentile is a score that falls inside the frequency distribution of a given percentage [22]. It brings up the data values being spread out along the interval, from the lowest to the highest, concerning the p^{th} percentile of $100 - P$ data values, as well as P percentage of data values that exceed the K^{th} percentile. Further, it is referred as *PER*.
- (d) *RMS*: The square root of the mean square is the arithmetic mean of the squares of the integers, which is specified as RMS [22]. The quadratic mean is also known as RMS.

$$RMS = \sqrt{\frac{1}{s}(q_1^2 + q_2^2 + \dots + q_s^2)} \quad (9)$$

Where, $q_1, \dots, q_s$ indicates the items under observation and s denotes the total number of items. The higher-order statistical features are portrayed as *HF* in Eq. (10).

$$HF = Kurtosis + Skewness + PER + RMS \quad (10)$$

The entire extracted features are indicated as $FE = SF + HF$.

5. Proposed Classification using QNN Classifiers

The QNN classifier is used in the classification stages from these extracted features (FE). FE are passed throughout the classification step to the QNN classifier [23]. The QNN is a basic construction block; the quantum perceptron is a counterpart of perceptrons used throughout traditional ML. A quantum perceptron with I input & O output qubits is an arbitrary unitary operator. The perceptron is merely an arbitrary unitary with $I + O$ output and input qubits that have $(2^{I+O})^2 - 1$ parameters. The output qubits are placed in a fiducial state $|0\dots 0\rangle_{out}$, while the input qubits are allocated to an indeterminate mixed state ψ^{in} . They consider

perceptrons on 1 output qubit or I input qubit i.e.; $I + 1$ - qubit unitaries, for simplification.

A quantum neuron explains the structure of the QNN. In quantum perceptrons, the QNN is a quantum circuit with Q hidden layers that operate on original state-input qubits ψ & generate output diverse state qubits ψ and operational needs.

$$\psi^{out} \equiv tr_{in,hid}(G(\psi^{in} \otimes |0\dots 0\rangle_{hid,out} \langle 0\dots 0| G^+)) \quad (11)$$

Therefore, G^k represents the unitaries layer with quantum perceptrons qubits product in layers $k-1$ and k & $G \equiv G^{out} G^Q G^{Q-1} \dots G^1$ signifies the QNN quantum circuit. The network's output may be defined as a collection of completely positive layer-to-layer transition maps ϑ^k , that is an essential element of QNN explanation:

$$\psi^{out} = \vartheta^{out}(\psi^Q(\dots \vartheta^2(\vartheta^1(\psi^{in})))\dots) \quad (12)$$

$$\vartheta^k(D^{k-1}) \equiv tr_{k-1} \left(\prod_{l=1}^1 G_l^k(D^{k-1} \otimes |0\dots 0\rangle_l \langle 0\dots 0|) \prod_{l=1}^{I_k} G_l^{k+} \right) \quad (13)$$

Wherein G_l^k represents the l^{th} perceptron in layers $k-1$ and k . Ik Indicates the total number of perceptrons in $k-1$ & k layers.

Information propagates from the sender to the receiver, indicating a quantum feedforward NN is easily constructed as per this description of a QNN's output. This crucial finding is the foundation of the backpropagation process quantum version. A current concept of a quantum perceptron as a managed unitary, i.e. $G = \sum_{\omega} |\omega\rangle \langle \omega| \otimes G(\omega)$, as $|\omega\rangle$ signals particular fundamentals of the input space & $G(\omega)$ are parametrized unitary, might describe the quantum perceptron's choosing for QNNs. Where, $\psi^{out} = \sum_{\omega} \langle \omega | \psi^{in} | \omega \rangle G(\omega) | 0 \rangle \langle 0 | G(\omega)^+$.

The training algorithm is as follows: In the quantum surroundings, the fundamental probability distribution is a distribution of quantum states. It is logical to assume the J samples seem to be J comparable quantum states.

Researchers emphasize on the circumstance with reliable access to training data ($|\alpha_b^{in}\rangle, |\alpha_b^{out}\rangle$), $b = 1, 2, \dots, J$ of possibly unexpected quantum states in pairs.

Furthermore, the copy of data necessary per training round will develop rapidly with a number of hidden neurons, i.e. $T_{proj} + T_{params}$, T_{proj} suggests the factor caused by repeated metrics to reduce projection noise, as well as T_{params} signifies the whole number of parameters in the system operated as $\sum_{k=1}^{Q+1} (4^{(I_{k-1}+1)} - 1) \times I_k$, wherein I_k relates to the

number of perceptrons on $k - 1$ & k layer, as well as the -1 term appears to be insignificant. They will now emphasize the limited case wherein $|\alpha_b^{out}\rangle = M |\alpha_b^{in}\rangle$ and M signify some undefined unitary function for the sake of clarity.

The model needs a cost function to quantify the QNN's results in training data, i.e., the network's output S_x^{out} is the correct output $|\alpha_b^{out}\rangle$ given the input. $|\alpha_b^{in}\rangle$ For (purity) quantum states, the fidelity between QNN output & planned output aggregated throughout the training data:

$$C = \frac{1}{J} \sum_{b=1}^J \langle \alpha_b^{out} | \psi_b^{out} | \alpha_b^{out} \rangle \quad (14)$$

Whereas the training data output states are not hygienic (the fidelity for mixed states: $Z(F, \Omega) := [\text{tr} \sqrt{F^{1/2} \Omega F^{1/2}}]^2$), this may occur if the network is trained to acquire a quantum channel, the network takes another more complex shape.

The cost function has a range of 0 to 1 (minimum to maximum). To train the QNN, the cost function C is optimized. Additionally, this is achieved by upgrading the QNN attributes: updating the perceptron unitaries as per $G \rightarrow e^{i\tau L} G$ at every training step, L representing the matrix comprising unitaries parameters of the relevant perceptron, and τ signifies the given step size. Moreover, the matrices L are chosen depending on the cost function rise as much as possible, so an alteration in C equals to

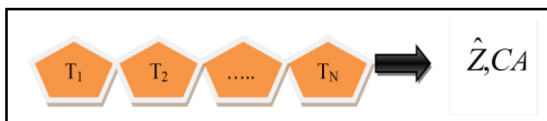
$$DC = \frac{\tau}{J} \sum_{b=1}^J \sum_{k=1}^{Q+1} \text{tr}(\Omega_b^k \Delta \vartheta^k (F_b^{k-1})) \quad (15)$$

In Eq. (15), $Q+1 = out$, $\psi_b^k = \vartheta^k(\dots \vartheta^2(\vartheta^1(\psi_b^{k-1}))\dots)$, $\Omega_b^k = \mathfrak{Z}^{k+1}(\dots \mathfrak{Z}^Q(\mathfrak{Z}^{out}(|\alpha_b^{out}\rangle\langle\alpha_b^{out}|))\dots)$ & $\mathfrak{Z}(D) = \sum_{\omega} A_{\omega}^+ D A_{\omega}$ indicates the adjoint channel in the CP map $\vartheta(D) = \sum_{\omega} A_{\omega} D A_{\omega}^+$. The output of the QNN classifier is denoted as CL_{QNN} .

6. Optimal tuning of hidden neurons Via Cubic Chaotic Map Integrated Cat and Mouse based Optimization model

6.1 Objective Function and Solution Encoding

The number of hidden neurons is optimally tuned using the CC-CMBO approach. The encoding solution of the suggested CC-CMBO approach is given in Figure 2. Moreover,

**Figure 2****Solution Encoding**

N represents the total number of hidden neurons & T signifies the hidden neurons. Eq. (16) specifies the objective function. Here, the error function in QNN is given as (err).

$$Obj = \min(err) \quad (16)$$

6.2 Proposed CC-CMBO model

Even though the standard CMBO [25] model is used in optimal designs and engineering sciences, uniquely picked variables should be properly identified, thereby increasing device efficiency. Yet this may not solve the optimization problem. The CC-CMBO model is proposed in this study to address these problems. Self-enhancement has already been demonstrated in traditional optimization approaches [26][27][28].

The CMBO is a population-based approach based on a cat pursuing a mouse and the mouse running to a place of refuge. The proposed technique categorizes the search agents into two groups: cats and mice, randomly exploring the issue search area. Members of the population are updated in two stages by the suggested technique. The proposed technique upgrades population members in two sections. The first phase depicts the cats' hunting of mice, whereas the second depicts mice escaping to safe havens to save their lives.

Every member of the population represents a possible solution to the problems from a mathematical viewpoint. Consequently, each individual in the population is a vector, and the vectors' values describe the issue's variables. The CC-CMBO model is accessible below. The search agent population is set up for the first time. A matrix termed as the population matrix E is used to assess the population of this model, as shown in Eq (17).

$$E = \begin{bmatrix} E_1 \\ . \\ E_g \\ \vdots \\ E_u \end{bmatrix}_{U \times V} = \begin{bmatrix} e_{1,1} & \cdots & e_{1,d} & \cdots & e_{1,V} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ e_{g,1} & \cdots & e_{g,d} & \cdots & e_{g,V} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ e_{u,1} & \cdots & e_{u,d} & \cdots & e_{u,V} \end{bmatrix}_{U \times V} \quad (17)$$

Where, $e_{g,d}$ signifies the d^{th} problem variable discovered of g^{th} search agent, E indicates the CMBO population matrix, E_g is the g^{th} search agent, U defines the population members count, & V relates to the number of problem variables. Eq. (18) gives the results obtained for the objective function.

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_u \end{bmatrix}_{U \times 1} \quad (18)$$

In Eq. (18), X_g signifies the g^{th} search agent objective function value as well as X the objective function values vector. Modify the sorting population matrix E^W by Eq. (19). The population h^{th} of the sorted population matrix is indicated as $E_{g,d}^W$ & X^W a sorted objective function-based vector in Eq. (20).

$$E^W = \begin{bmatrix} E_1^W \\ E_2^W \\ \vdots \\ E_u^W \end{bmatrix}_{U \times V} = \begin{bmatrix} e_{1,1}^W & \cdots & e_{1,d}^W & \cdots & e_{1,V}^W \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ e_{g,1}^W & \cdots & e_{g,d}^W & \cdots & e_{g,V}^W \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ e_{u,1}^W & \cdots & e_{u,d}^W & \cdots & e_{u,V}^W \end{bmatrix}_{U \times V} \quad (19)$$

$$X^W = \begin{bmatrix} X_1^W & \min(X) \\ X_2^W & \min(X) \\ \vdots & \vdots \\ X_u^W & \min(X) \end{bmatrix}_{U \times 1} \quad (20)$$

Whereas $e_{g,d}^W$ indicates the value obtained by the g^{th} search agent for the d^{th} issue variable, E_g^W corresponds to a g^{th} member of the sorted population matrix, E^W shows the sorted population matrix on the objective

function value, & X^W signifies the sorted vector of an objective function. The mouse population is selected by Eq. (21).

$$\hat{Z} = \begin{bmatrix} \hat{Z}_1 = E_1^W \\ \vdots \\ \hat{Z}_g = E_g^W \\ \vdots \\ \hat{Z}_{U_V} = E_{U_V}^W \end{bmatrix}_{U_V * V} = \begin{bmatrix} e_{1,1}^W & \cdots & e_{1,d}^W & \cdots & e_{1,V}^W \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ e_{g,1}^W & \cdots & e_{g,d}^W & \cdots & e_{g,V}^W \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ e_{U_V,1}^W & \cdots & e_{U_V,d}^W & \cdots & e_{U_V,V}^W \end{bmatrix}_{U_V * V} \quad (21)$$

The cat population is selected using Eq. (22).

$$CA = \begin{bmatrix} CA_1 = E_{U_V+1}^W \\ \vdots \\ CA_v = E_{U_V+t}^W \\ \vdots \\ CA_{U_n} = X_{U_V+U_n}^R \end{bmatrix}_{U_n * V} = \begin{bmatrix} e_{U_V+1,1}^W & \cdots & e_{U_V+1,d}^W & \cdots & e_{U_V+1,V}^W \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ e_{U_V+t,1}^W & \cdots & e_{U_V+t,d}^W & \cdots & e_{U_V+t,V}^W \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ e_{U_V+U_n,1}^W & \cdots & e_{U_V+U_n,d}^W & \cdots & e_{U_V+U_n,V}^W \end{bmatrix}_{U_n * V} \quad (22)$$

Thus, $\hat{Z}, U_V, \hat{Z}_g, CA, U_n, CA_t$ corresponds to mouse population, mice count, t^{th} mice, cat population, cat count, & g^{th} cat, correspondingly. In 1st phase, the variation in the position of cats is simulated depending on actual cat behavior and attitude towards mice for updating the search parameters. In addition, Eq. (23) to Eq. (25) quantitatively describes this phase of the suggested CC-CMBO's update.

$$CA_t^{new} : CA_{t,d}^{new} = n_{t,d} + r * (V_{\hat{l},d} - p * n_{t,d}) \& t = 1 : U_n, \\ d = 1 : V, \hat{l} \in 1 : U_V \quad (23)$$

$$\text{Here,} \quad p = \text{round}(1 + \text{rand}) \quad (24)$$

$$CA_t = \begin{cases} CA_t^{new}, & |X_t^{n,new} < X_t^n, \\ CA_t, & \text{else} \end{cases} \quad (25)$$

Conventionally, in Eq. (24), the V value can be calculated in cat updation. As per the adopted CC-CMBO model, rand is the random number generated using cubic chaotic map in Eq. (26).

$$v_{w+1} = \rho v_w (1 - v_w^2) \quad v_w \in (0, 1) \quad (26)$$

Where, $CA_{t,d}^{new}$ signifies the new value for the d^{th} problems variable obtained by the t^{th} cat, $X_t^{n,new}$ represents the objective function value based on the t^{th} cat's new position, CA_t^{new} corresponds to the t^{th} cat's new status, corresponds to a random integer between $[0, 1]$, and $V_{i,d}$ describes the $\hat{i}^{th}$ mouse's d^{th} dimension.

The projected CC-CMBO phase 2 simulates mice escaping to a place of refuge. Every mouse is said to have a random refuge in CC-CMBO, and mice seek shelter in these havens. By arranging the locations of different technique elements, the havens placements in the search area are created randomly. Eq. (27) to (29) describe this phase of the mice's position update.

$$\tilde{B}_g = \tilde{q}_{g,d} = \hat{z}_{pd} \ \& \ g = 1 : U_v, d = 1 : V, p \in 1 : U \quad (27)$$

$$\hat{Z}_g^{new} : V_{g,d}^{new} = V_{g,d} + r * (\tilde{q}_{g,d} - p * V_{g,d}) * Sign(X_g^V - X_g^{\tilde{B}}) + Levy(\beta) \quad (28)$$

where, $g = 1 : U_v, d = 1 : V$

$$\hat{Z}_g = \begin{cases} \hat{Z}_g^{new} & | \ X_g^{V,new} < X_g^V \\ \hat{Z}_g & | \ \text{else} \end{cases} \quad (29)$$

As per the proposed CC-CMBO model, the $\tilde{q}_{g,d}$ value is determined in mouse updation as per Eq. (30).

$$\tilde{q}_{g,d} = \frac{\hat{Z}_g^{best} + \hat{Z}_g^{worst}}{2} \quad (30)$$

Here, $\tilde{q}_{g,d}$ represents the average of the best and worst mice populations, $Levy(\beta)$ the levy updation function, $\tilde{B}_g$ indicates to the g^{th} mouse's haven, $X_g^{\tilde{B}}$ signifies its objective function value, $\hat{Z}_g^{new}$ represents the g^{th} mouse's novel status and $X_g^{V,new}$ is the objective function value, correspondingly.

Arithmetic Crossover: It is used in the case of real-value encoding. The operator of an arithmetic crossover predictably connected the two parent chromosomes. Two chromosomes are randomly chosen for crossover in an arithmetic crossover, but a linear combination of chromosomes creates two children. This linear combination may be calculated using the following formula:

$$\tilde{G}1 = \hat{W}.\tilde{Q}1gene + (1 - \hat{W}).\tilde{Q}2gene \quad (31)$$

$$\tilde{G}2 = \hat{W}.\tilde{Q}2gene + (1 - \hat{W}).\tilde{Q}1gene \quad (32)$$

Whereas $\tilde{Q}_{gene}$ an old generation individual, $\tilde{G}$ a new generation individual, as well as $\hat{W}$ denotes the weight of a direct dominating individual between 0 as well as 1. The pseudocode is determined in Algorithm 2.

Algorithm 2: Proposed CC-CMBO method

Begin CC-CMBO

Objective function, constraints, and variables.

Group of search agents (U) & iterations ($\tilde{T}$).

Randomly create an initial population matrix.

Determine the objective function.

For $\hat{t} = 1 : \tilde{T}$

The population matrix depends on objective function value in Eq. (19) and (20).

Choose mice population $\hat{Z}$ in Eq. (21)

Choose cats population CA in Eq. (22).

Phase 1: Cat update.

For $t = 1 : U_n$

Inform status of the t^{th} cat in Eq. (23) to (25).

As per the adopted CC-CMBO model, $rand$ is the random number generated using cubic chaotic map in Eq. (26)

End

Phase 2: Mice update.

For $g = 1 : U_v$

Generate haven for the g^{th} mouse in Eq (27).

Update the g^{th} mouse in Eq (28) and (29).

As per the proposed CC-CMBO model, the $\tilde{q}_{g,d}$ value is determined in mouse updation as per Eq. (30).

End

Perform Arithmetic crossover in Eq. (31) and Eq. (32).

End

Output best solution

End CC-CMBO

7. Results & Discussions

7.1 Simulation Procedure

The suggested QNN+CC-CMBO approach for IPDS in cloud was implemented in PYTHON and its outcomes were verified. Also, the findings of the implemented QNN+CC-CMBO technique were compared to those of traditional approaches such as SVM [29], LSTM [30], Bi-GRU [31], DBN [32], RNN [33], QNN + BOA [34], QNN + CS [35], QNN + SSA [36], QNN+ SMO [37], and QNN + CMBO [25]. Here, the dataset was obtained from CSE-CIC-IDS2018 [38]. The University of New Brunswick initially developed this dataset to analyze DDoS data. This dataset was obtained entirely in 2018; nevertheless, future updated versions will be available to access. The dataset was created out of university's servers logs, revealing several DoS assaults during the course of accessible time. While creating machine learning notebooks for this data, it was made sure that the Label column was the most crucial part as it decides whether the delivered packets are malicious or not. For several measures, including FPR, accuracy, F1-score, precision, MCC, recall, FNR, sensitivity, specificity, & NPV, the outcomes were examined by adjusting the training percentage from 60 %, 70 %, 80 %, & 90 %, respectively.

7.2 Performance Analysis

The developed QNN+CC-CMBO system is tested against established approaches such as SVM, LSTM, Bi-GRU, DBN, RNN, QNN + BOA, QNN +CS, QNN + SSA, QNN+ SMO, & QNN +CMBO, depending on different measures, as shown in Figure 3 to 5. Positive indicators, including specificity, accuracy, sensitivity, and precision, are also depicted in Figure 3. Furthermore, the developed QNN+CC-CMBO system has the best accuracy value (~92%), outperforming conventional approaches at a training percentage of 80% in Figure 3. (b). Furthermore, the suggested QNN+CC-CMBO system provides higher sensitivity of 5.61 %, 2.24 %, 8.98 %, 12.35 %, 4.49 %, 3.37 %, 2.80 %, 3.03 %, 3.14 %, and 2.13% to traditional models like SVM, LSTM, Bi-GRU, DBN, RNN, QNN + BOA,

QNN + CS, QNN + SSA, QNN+ SMO, and QNN +CMBO. This showed that the suggested QNN+CC-CMBO scheme outperforms the alternative models to a larger extent. In addition, when compared to existing models, the developed QNN+CC-CMBO method reaches greater specificity (0.90) for a training percentage of 70% Figure 3(d). As a result, the established QNN+CC-CMBO method cleared the path for improved IDS to IPDS in the cloud outcomes. Furthermore, as shown in Figure 3(c), the precision of the proposed QNN+CC-CMBO scheme for training 90% is greater than existing techniques such as SVM, LSTM, Bi-GRU, DBN, RNN, QNN + BOA, QNN +CS, QNN + SSA, QNN+ SMO, and QNN +CMBO, correspondingly as it shows enhancement in the adopted scheme. This performance shows higher results based on the intrusion detection system in the cloud due to the appropriate training of hidden neurons in QNN to minimize the errors.

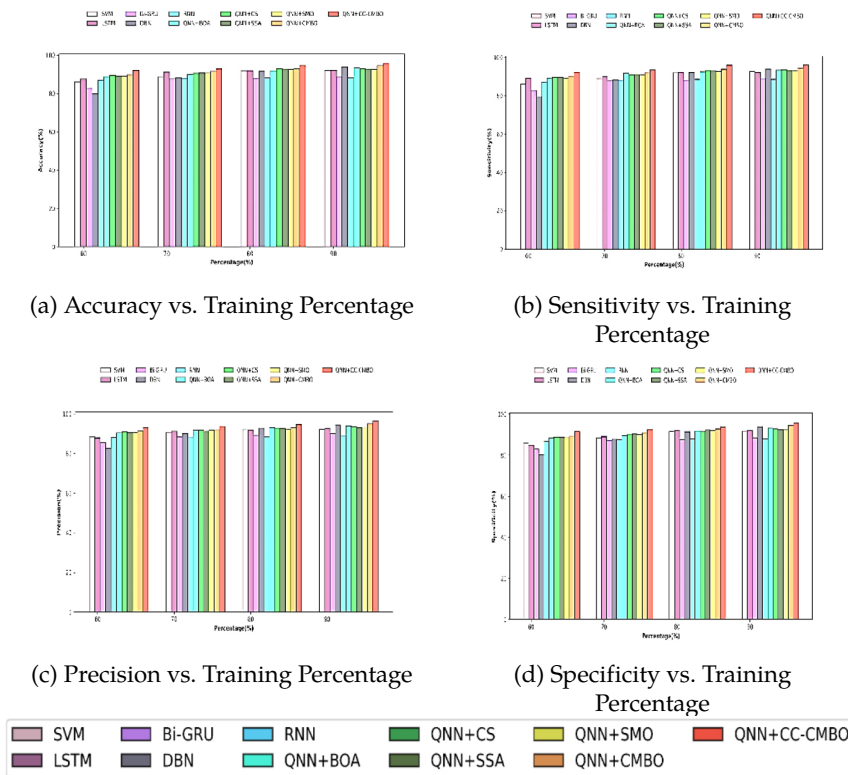


Figure 3

Performance Analysis of the suggested scheme to previous approaches for
(a)accuracy (b)sensitivity (c) precession (d) specificity.

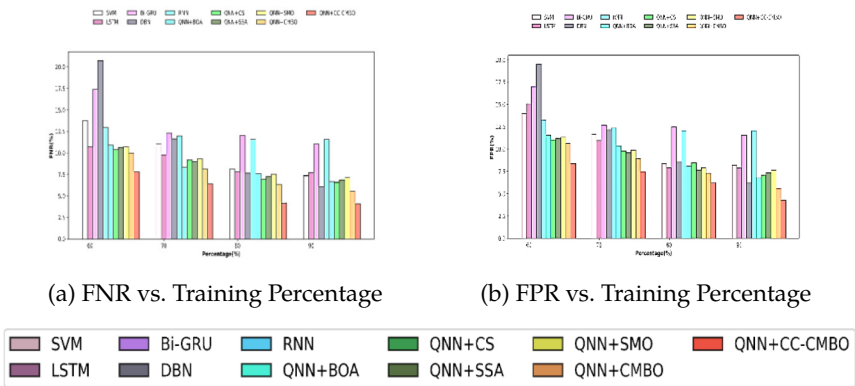


Figure 4

Performance Analysis of the suggested scheme to previous approaches for (a)FNR (b)FPR

The FPR and FNR of the developed QNN+CC-CMBO approach against the traditional approaches such as SVM, LSTM, Bi-GRU, DBN, RNN, QNN + BOA, QNN +CS, QNN + SSA, QNN+ SMO, and QNN +CMBO, respectively is given in Figure 4. Further, the developed QNN+CC-CMBO approach at training percentage of 70% achieved less FPR value (~7.5) with higher outcomes than extant approaches in Figure 4(b). The resultant of the developed approach attains minimum values as the difference in training percentage is maximized. Further, the developed QNN+CC-CMBO scheme for training percentage of 90% attains minimum FNR with superior outcomes to extant approaches SVM, LSTM, Bi-GRU, DBN, RNN, QNN + BOA, QNN +CS, QNN + SSA, QNN+ SMO, and QNN +CMBO, correspondingly in Figure 4(a).

7.3 Statistical Analysis

Table 1 provides the statistical analysis of the proposed QNN+CC-CMBO system based on error measures compared to conventional techniques. Nature’s meta-heuristic methods are stochastic, which indicates they’re performed numerous times to determine if the given target is reached. In the best-case scenario, the presented QNN+CC-CMBO model outperforms with more accurate results (~1.07) than previous existing models. The proposed QNN+CC-CMBO strategy has superior results to existing schemes as QNN + BOA, QNN +CS, QNN + SSA, QNN+ SMO, and QNN +CMBO based on mean performance. The

presented QNN+CC-CMBO approach has outperformed other existing models in terms of IDS in the cloud.

Table 1
Statistical Analysis based on Error: Suggested Vs Approaches

Method	Standard Deviation	Mean	Median	Max (Best)	Min (Worst)
QNN + BOA [29]	0.006622	1.052101	1.049326	1.06978	1.049326
QNN +CS [30]	0.010469	1.053168	1.050376	1.095593	1.04728
QNN + SSA [31]	0.005193	1.049661	1.049825	1.081083	1.047253
QNN+ SMO [32]	0.002685	1.050733	1.05145	1.061927	1.047639
QNN +CMBO [25]	0.005389	1.052885	1.05337	1.066001	1.047236
Proposed QNN + CC-CMBO model	0.00838	1.051294	1.049306	1.071789	1.044745

7.4 Analysis of different combinations of suggested scheme

The analysis of suggested QNN+CC-CMBO scheme with different combinations for various metrics is illustrated in Table 2. Moreover, the suggested QNN+CC-CMBO model holds maximum sensitivity (~0.960) than suggested scheme with original features and suggested scheme without optimization, correspondingly. In addition, the proposed QNN+CC-CMBO model attains lower FPR with better outcomes than the suggested scheme with original features and suggested scheme without optimization, correspondingly. This has proved that the suggested QNN+CC-CMBO scheme detects the attacks in cloud accurately, while the proposed scheme with original features and suggested scheme without optimization shows the least performance with the suggested concept. This proves that the developed combination is suitable for IDS in cloud, leading it to IPDS.

Table 2
Analysis based on different combinations of suggested schemes

Metrics	Suggested scheme with original features	Suggested scheme without optimization	Proposed QNN+CC-CMBO scheme
Precision	0.939532	0.952633	0.965756
FNR	0.07446	0.053738	0.039914

Contd...

Specificity	0.919083	0.941472	0.955655
NPV	0.90086	0.932535	0.947639
F1-score	0.932483	0.949437	0.962913
MCC	0.842505	0.90745	0.925447
Accuracy	0.922802	0.944293	0.958271
FPR	0.080917	0.058528	0.044345
Sensitivity	0.92554	0.946262	0.960086

7.5 Convergence Analysis

Proposed QNN+CC-CMBO system is compared with the existing system in terms of convergence in Figure 5. As per Eq. (16), the suggested QNN+CC-CMBO strategy attains the least cost function. The convergence of the proposed QNN+CC-CMBO system to conventional methods including QNN + BOA, QNN +CS, QNN + SSA, QNN+ SMO, and QNN +CMBO is determined through adjusting the count of iteration from 0, 10, 20, 30, 40, & 50. Since the number of iterations rises, the cost function of the suggested QNN+CC-CMBO technique is reduced. In particular, the cost function of the recommended QNN+CC-CMBO model decreased at the 30th iteration. From 30th to the 50th iteration, the proposed QNN+CC-CMBO method produces a smaller constant value (0.15) to earlier techniques including QNN + BOA, QNN +CS, QNN + SSA, QNN+ SMO, and QNN +CMBO. Consequently, the proposed QNN+CC-CMBO technique yielded the lowest value with the best performance.

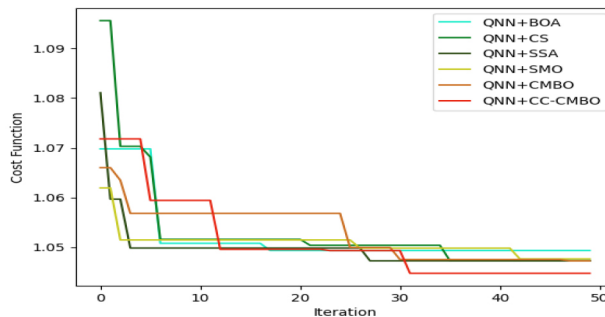


Figure 5

Comparison of the proposed QNN+CC-CMBO system to existing system in terms of convergence

8. Conclusion

This article has proposed a new intrusion detection-based cloud computing system comprising three stages: (1) pre-processing, (2) feature extraction, and (3) classification. pre-processing was performed on the input data in which the Z-score normalization was performed. The pre-processed data was subjected to feature extraction, which includes statistical and higher-order statistical features. In addition, the extracted features were passed on to the classification phase, which consists of the Optimized QNN classifier. For increased optimization, the hidden neuron optimization was performed by CC-CMBO Algorithm. Thus, the classified result was attained in a precise way. In addition, comparing existing models to that of the developed QNN+CC-CMBO method at 70% training reaches greater specificity of (0.90). Additionally, the suggested QNN+CC-CMBO model has the highest sensitivity (0.960) compared to the proposed schemes with original characteristics and without optimization. Compared to earlier methods such as SVM, LSTM, Bi-GRU, DBN, RNN, QNN + BOA, QNN + CS, QNN + SSA, QNN + SMO, and QNN + CMBO, the recommended QNN+CC-CMBO scheme with training percentage of 80% achieves a higher recall (~96). Further, the developed QNN+CC-CMBO scheme for 90% training attains minimum FNR with aggravated outcomes to the existing approaches like SVM, LSTM, Bi-GRU, DBN, RNN, QNN + BOA, QNN +CS, QNN + SSA, QNN+ SMO, and QNN+CMBO, correspondingly. Further, the F1-score of the suggested QNN+CC-CMBO scheme for 80% training achieves maximum value with respect to another training percentage of 70%, which is higher than previous approaches such as SVM, LSTM, Bi-GRU, DBN, RNN, QNN + BOA, QNN +CS, QNN + SSA, QNN+ SMO, and QNN +CMBO, respectively. The proposed QNN+CC-CMBO strategy has superior results than existing schemes as QNN + BOA, QNN +CS, QNN + SSA, QNN+ SMO, and QNN +CMBO based on mean performance. The proposed QNN+CC-CMBO technique yielded the lowest constant value (0.15) with the best performance. Quantum algorithms require bigdata datasets, but at the same time, they require a large number of physical qubits to be accommodated, further leading to a huge control overhead. In the future, the proposed algorithm can be applied in various application and domain-specific areas such as biomedicine and drug designing, financial modeling, and AI-based robotics adopting quantum random walks.

Appendix A

Abbreviation	Description
FL	Fuzzy Logic
CIA	Confidentiality, Integrity and Availability
ABC	Artificial Bee Colony
CC	Cloud Computing
GA	Genetic Algorithm
IT	Information Technology
PSO	Particle Swarm Optimization
DDoS	Distributed Denial of Service
HAS	Harmony Search Algorithm
OT2FNN	Optimal Type-2 Fuzzy Neural Network
RC-NN	Recurrent Convolutional Neural Network
N-IDS	Network-based IDS
SVM	Support Vector Machine
FNR	False Negative Rate
ANN	Artificial Neural Network
FPR	False Positive Rate
CNN	Convolutional Neural Network
IPDS	Intrusion Pattern Detection System
IDS	Intrusion Detection System
LSTM	Long Short Term Memory
KFCM	Kernel Fuzzy C-Means Clustering
CC-CMBO	Cubic Chaotic Map integrated Cat and Mouse Based Optimization
MCC	Mathews Correlation Coefficient
U2R	User-to-Root
RNN	Recurrent Neural Network
RHS	Random Harmony Search
LOA	Lion Optimization Algorithm
NK	Normalized K-Means clustering
R2L	Remote-to-Local
RBM	Restricted Boltzmann Machines
FCM	Fuzzy C Means
ALO	Ant Lion optimization
CMBO	Cat Mouse Based Optimization
DoS	Denial-of-Service

References

- [1] Manickam, M.; Rajagopalan, S.P. A hybrid multi-layer intrusion detection system in cloud. *Cluster Comput.* 22, 3961–3969 (2019), doi:10.1007/s10586-018-2557-5.
- [2] Tummalapalli, S.R.K.; Chakravarthy, A.S.N. Intrusion detection system for cloud forensics using bayesian fuzzy clustering and optimization based SVNN. *Evol. Intell.* 14, 699–709 (2021), doi:10.1007/s12065-020-00410-y.
- [3] Besharati, E.; Naderan, M.; Namjoo, E. LR-HIDS: logistic regression host-based intrusion detection system for cloud environments. *J. Ambient Intell. Humaniz. Comput.* 0, 0 (2018), doi:10.1007/s12652-018-1093-8.
- [4] Singh, P.; Ranga, V. Attack and intrusion detection in cloud computing using an ensemble learning approach. *Int. J. Inf. Technol.*, 13, 565–571 (2021), doi:10.1007/s41870-020-00583-w.
- [5] Abusitta, A.; Bellaiche, M.; Dagenais, M. Multi-cloud cooperative intrusion detection system: trust and fairness assurance. *Ann. des Telecommun. Telecommun.* 74, 637–653 (2019), doi:10.1007/s12243-019-00724-8.
- [6] Prabhakaran, V.; Kulandasamy, A. Hybrid semantic deep learning architecture and optimal advanced encryption standard key management scheme for secure cloud storage and intrusion detection. *Neural Comput. Appl.* 5 (2021), doi:10.1007/s00521-021-06085-5.
- [7] Wen, L. Cloud Computing Intrusion Detection Technology Based on BP-NN. *Wirel. Pers. Commun.* (2021), doi:10.1007/s11277-021-08569-y.
- [8] Veeraiah, N. Intrusion Detection Based on Piecewise Fuzzy C-Means Clustering and Fuzzy Naïve Bayes Rule. *Multimed. Res.* 1, 27–32 (2018), doi:10.46253/j.mr.v1i1.a4.
- [9] A.V.Krishna Prasad Deep Learning based optimization for Detection of Attacks in IoT. *J. Netw. Commun. Syst.* 4, 31–36 (2021), doi:10.46253/jnacs.v4i2.a5.

- [10] Mehanović, D.; Kečo, D.; Kevrić, J.; Jukić, S.; Miljković, A.; Mašetić, Z. Feature selection using cloud-based parallel genetic algorithm for intrusion detection data classification. *Neural Comput. Appl.*, 33, 11861–11873 (2021), doi:10.1007/s00521-021-05871-5.
- [11] Abdulqadder, I.H.; Zhou, S.; Zou, D.; Aziz, I.T.; Akber, S.M.A. Multi-layered intrusion detection and prevention in the SDN/NFV enabled cloud of 5G networks using AI-based defense mechanisms. *Comput. Networks*, 179, 107364 (2020), doi:10.1016/j.comnet.2020.107364.
- [12] Vaibhav Ankush Thorat Cloud Intrusion Detection using Modified Crow Search Optimized based Neural Network. *J. Netw. Commun. Syst.*, 4, 9–15 (2021), doi:10.46253/jnacs.v4i2.a2.
- [13] Thilagam, T.; Aruna, R. Intrusion detection for network based cloud computing by custom RC-NN and optimization. *ICT Express*, 7, 512–520 (2021), doi:10.1016/j.icte.2021.04.006.
- [14] Srilatha, D.; Shyam, G.K. Cloud-based intrusion detection using kernel fuzzy clustering and optimal type-2 fuzzy neural network. *Cluster Comput.*, 24, 2657–2672 (2021), doi:10.1007/s10586-021-03281-9.
- [15] Balamurugan, V.; Saravanan, R. Enhanced intrusion detection and prevention system on cloud environment using hybrid classification and OTS generation. *Cluster Comput.*, 22, 13027–13039 (2019), doi:10.1007/s10586-017-1187-7.
- [16] Mayuranathan, M.; Murugan, M.; Dhanakoti, V. Best features based intrusion detection system by RBM model for detecting DDoS in cloud environment. *J. Ambient Intell. Humaniz. Comput.*, 12, 3609–3619 (2021), doi:10.1007/s12652-019-01611-9.
- [17] 2Jaber, A.N.; Rehman, S.U. FCM–SVM based intrusion detection system for cloud computing environment. *Cluster Comput.*, 23, 3221–3231 (2020), doi:10.1007/s10586-020-03082-6.
- [18] Xue, L.; Cheng, L.; Li, Y.; Mao, Y. Quantum Machine Learning for Electricity Theft Detection: an Initial Investigation. In *Proceedings of the 2021 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCoM) and IEEE Smart*

- Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics); IEEE Computer Society: Los Alamitos, CA, USA, pp. 204–208 (2021).
- [19] NIST/SEMATECH e-Handbook of Statistical Methods. Available online: <https://www.itl.nist.gov/div898/handbook/> (accessed on 22 June 2022).
- [20] Beer, K.; Bondarenko, D.; Farrelly, T.; Osborne, T.J.; Salzmann, R.; Scheiermann, D.; Wolf, R. Training deep quantum neural networks. *Nat. Commun.*, 11, 1–6 (2020), doi:10.1038/s41467-020-14454-2.
- [21] Dehghani, M.; Hubálovský, Š.; Trojovský, P. Cat and mouse based optimizer: A new nature-inspired optimization algorithm. *Sensors*, 21, 1–30 (2021), doi:10.3390/s21155214.
- [22] Rajakumar, B.R. Impact of static and adaptive mutation techniques on the performance of Genetic Algorithm. *Int. J. Hybrid Intell. Syst.*, 10, 11–22 (2016), doi:10.3233/his-120161.
- [23] Rajakumar, B.R.; George, A. A new adaptive mutation technique for genetic algorithm. 2012 IEEE Int. Conf. Comput. Intell. Comput. Res. ICCIC 2012 (2012), doi:10.1109/ICCIC.2012.6510293.
- [24] Avci, E. A new intelligent diagnosis system for the heart valve diseases by using genetic-SVM classifier. *Expert Syst. Appl.*, 36, 10618–10626 (2009), doi:10.1016/j.eswa.2009.02.053.
- [25] Zhou, X.; Lin, J.; Zhang, Z.; Shao, Z.; Chen, S.; Liu, H. Improved itracker combined with bidirectional long short-term memory for 3D gaze estimation using appearance cues. *Neurocomputing*, 390, 217–225 (2020), doi:10.1016/j.neucom.2019.04.099.
- [26] Zhao, D.; Wang, J.; Lin, H.; Yang, Z.; Zhang, Y. Extracting drug–drug interactions with hybrid bidirectional gated recurrent unit and graph convolutional network. *J. Biomed. Inform.*, 99, 103295 (2019), doi:10.1016/j.jbi.2019.103295.
- [27] Wang, H.Z.; Wang, G.B.; Li, G.Q.; Peng, J.C.; Liu, Y.T. Deep belief network based deterministic and probabilistic wind speed forecasting approach. *Appl. Energy*, 182, 80–93 (2016), doi:10.1016/j.apenergy.2016.08.108.
- [28] Kao, L.J.; Chiu, C.C. Application of integrated recurrent neural network with multivariate adaptive regression splines on SPC-

- EPC process. *J. Manuf. Syst.*, 57, 109–118 (2020), doi:10.1016/j.jmsy.2020.07.020.
- [29] Arora, S.; Singh, S. Butterfly optimization algorithm: a novel approach for global optimization. *Soft Comput.*, 23, 715–734 (2019), doi:10.1007/s00500-018-3102-4.
- [30] Mohamad, A.; Zain, A.M.; Bazin, N.E.N.; Udin, A. Cuckoo search algorithm for optimization problems - a literature review. *Appl. Mech. Mater.*, 421, 502–506 (2013), doi:10.4028/www.scientific.net/AMM.421.502.
- [31] Xue, J.; Shen, B. A novel swarm intelligence optimization approach: sparrow search algorithm. *Syst. Sci. Control Eng.*, 8, 22–34 (2020), doi:10.1080/21642583.2019.1708830.
- [32] Sharma, H.; Hazrati, G.; Bansal, J.C. Spider monkey optimization algorithm; Springer International Publishing, Vol. 779 (2019); ISBN 9783319913414.
- [33] IDS 2018 Intrusion CSVs (CSE-CIC-IDS2018) Kaggle. Available online: <https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv> (accessed on 22 June 2022).