Transactions on Machine Intelligence 3(2) (2020) 121-135

ISSN Online: 2821-1693

Transactions on Machine Intelligence

Journal Homepage: https://tmachineintelligence.ir/index.php/TMI

Intrusion Detection in Cloud Computing Virtualization Using Radial Basis Function Neural Network Optimized by Grasshopper Optimization Algorithm

S. Mirabi[1,*], S. Alizadeh[2]

[1] Department of Information Technology Engineering, Faculty of Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran
[2] Department of Computer Engineering, South Tehran Branch, Islamic Azad University, Tehran, Iran

ARTICLE INFO

Article History:
Received 7 March 2020
Received in revised form 6 April 2020
Accepted 28 June 2020
Available online 29 June 2020

Keywords:
Intrusion Detection, Virtualization, Cloud Computing, Radial Basis Function Neural Network, Grasshopper Optimization Algorithm

ABSTRACT

Cloud computing plays a crucial role in handling massive computations, providing a very simple computational model for users that meets their requests and needs at minimal cost. One of the major challenges in using cloud computing infrastructure is data security and preventing various possible intrusions. Intrusion detection systems (IDS) are one of the main components of cloud computing environment monitoring systems. This paper presents a hybrid learning system for use in intrusion detection systems in virtualization within cloud computing. After data collection and preparation, a Radial Basis Function Neural Network (RBFNN) trained with the Grasshopper Optimization Algorithm (GOA) is used as the proposed method for intrusion detection in cloud computing virtualization. GOA is employed to determine the centers, spread parameters, and weights of neurons in the RBFNN. The results are compared with the k-Nearest Neighbor (k-NN) classifier based on various error types and standard performance criteria. Simulation results indicate a 96.3% accuracy for the proposed method and show superior performance.

1. INTRODUCTION

The rapid increase in users and their need for internet services led to a situation where companies providing such services faced challenges like an inability to quickly respond to users and increased costs. As a result, many companies invested heavily in research to find effective and cost-efficient ways to serve a large number of users, leading to the emergence of a novel and efficient technology known as cloud computing [1]. Based on studies conducted in 2008, security has been identified as the most significant challenge in cloud computing. Cloud providers use virtualization technologies to enhance processing resources, making virtualization the most important infrastructure of cloud computing. Therefore, the security of cloud computing is of high importance [2]. Cloud computing, due to its advanced technology and accessibility, has garnered much attention. With the widespread use of cloud computing and the importance of software-as-a-service (SaaS) for daily information transfer, cloud security has become critical. Enhancing security in each cloud computing model is of paramount importance. Thus, intrusion

* Corresponding Author: mirabi.sara@gmail.com
Department of Information Technology Engineering, Faculty of Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

http://dx.doi.org/10.47176/TMI.2020.121

detection in cloud computing has become one of the most challenging security needs in recent years. Intrusion detection systems (IDS) help enhance security by detecting or preventing attacks in cloud computing and play an essential role in ensuring security. Therefore, IDS must be developed to analyze large volumes of data within an acceptable time frame to take appropriate action against attacks. Identifying attacks and threats is the first step in establishing security, followed by devising methods to address and prevent attacks [3]. The goal of this research is to detect intrusions in virtualization within cloud computing. The structure of this paper is as follows: Section 2 reviews related research; Section 3 discusses the algorithms used in this study; Section 4 describes the proposed model and its steps; Section 5 evaluates the results of using different algorithms to solve the problem, and the final section presents the conclusion.

2. LITERATURE REVIEW

In 2014, Nikolai et al. proposed a cloud intrusion detection system based on the hypervisor. Shared resources are a critical component of cloud computing. Virtualization provides several advantages for enhancing resource efficiency on a demand basis. However, these cloud features also raise security concerns. The paper presents an architecture for using virtualization technology at the core of cloud computing to detect intrusions using hypervisor metrics. By utilizing performance metrics collected by virtual machines from hypervisors, such as sent/received packets, read/write requests, and CPU usage, suspicious activities can be identified without detailed knowledge of the operating system profile running within the virtual machine. The proposed method for the cloud IDS does not require additional software installation on virtual machines and offers several advantages over network-based and host-based IDS, potentially complementing traditional IDS methods [4].

In 2015, Iqbal et al. addressed classification, intrusion detection, and prevention as a service in cloud attacks. The major part of cloud computing is primarily delivered through SaaS, PaaS, and IaaS. However, these service delivery models are vulnerable to a wide range of security attacks. Classifications provide a useful tool for system designers by offering a systematic method for understanding, identifying, and mitigating security services. This study collects and classifies cloud attacks and vulnerabilities based on cloud models. The paper presents a classification of cloud security operations and mitigation strategies aimed at providing a deep understanding of security needs in a cloud environment. This research highlights the importance of intrusion detection and prevention as a service [5].

In 2016, Kumar et al. developed an anomaly detection system in the cloud using a fuzzy-neural system. Despite ongoing advancements, cloud computing systems are still vulnerable to malicious activities. This necessitates the creation of an anomaly detection component to discover anomalies in the cloud environment. The paper proposes an anomaly detection system at the hypervisor layer within the cloud environment. Deploying fuzzy systems in IDS enhances the ability to detect unknown and uncertain anomalies in the cloud environment. One successful data-driven approach is integrating fuzzy systems with neural network adaptability and learning skills, known as the Adaptive Neuro-Fuzzy Inference System (ANFIS). The designed and developed detection system with ANFIS uses a hybrid algorithm combining backpropagation and gradient descent with the least-squares method. The study utilizes the KDD dataset, demonstrating that the ANFIS-based detection method performs anomaly detection in the cloud environment with minimal error and high accuracy [6].

In 2017, Moody et al. conducted a comprehensive review of virtualization layer security challenges and intrusion detection/prevention systems in cloud computing. Virtualization plays a significant role in constructing cloud computing. However, various vulnerabilities exist in current virtualization implementations, leading to diverse security challenges at the virtualization layer. This paper examines various vulnerabilities and attacks in the virtualization layer of cloud computing. The study reviews cloud IDS, intrusion detection systems, and preventive system frameworks. It examines the requirements for cloud IDS and the scope of research for enhancing security at the virtualization layer of cloud computing [7].

In 2018, Hatef et al. proposed a hybrid intrusion detection method in cloud computing. The rapid growth of distributed computing systems, which are highly interconnected and interact with each other, has increased the importance of countering cyber intruders, attackers, and saboteurs. Given the global deployment of cloud computing and its distributed and decentralized nature, special security measures are necessary to protect this paradigm. IDS can enhance security in cloud computing systems by monitoring, verifying, and controlling configurations, system

logs, network traffic, user activities, and even the actions of various processes. The position of intrusion detection mechanisms in cloud computing systems and the algorithms applied in those mechanisms are two main focuses of many studies. The aim is to detect as many attacks as possible while increasing detection speed and accuracy and reducing false alarms. However, these solutions typically have high computational loads, low accuracy, and high implementation costs. This paper presents a comprehensive and accurate solution for identifying and preventing intrusions in cloud computing systems using a hybrid method named HIDCC. Results show that intrusion coverage, detection accuracy, reliability, and availability in cloud computing systems significantly increase, while false alarms considerably decrease [8].

In 2019, Haji Mirzaei et al. investigated intrusion detection in cloud computing using the Bee Colony Optimization (BCO) algorithm-based neural network. This study presents a novel intrusion detection system based on a combination of multilayer perceptron (MLP) and artificial bee colony (ABC) algorithm and fuzzy clustering. Normal and abnormal network traffic packets are identified by the MLP, while the MLP is trained by the ABC algorithm through optimizing connection weight values. The CloudSim simulator and NSL-KDD dataset are used to validate the proposed method. Mean absolute error and root mean square error are considered as evaluation criteria. The obtained results indicate the superiority of the proposed method compared to modern methods [9].

In 2020, Sethi et al. proposed a deep reinforcement learning-based intrusion detection system in cloud infrastructure. Intrusion detection in cloud infrastructure is a challenging issue due to the extensive use and distributed nature, which are fixed targets for new and unknown attacks. IDS monitors and detects malicious activities in any computing system or network. However, most traditional computer IDS are vulnerable to new attacks. Additionally, they fail to maintain a balance between high accuracy and a low false positive rate (FPR). This paper presents a deep reinforcement learning-based architecture that addresses the aforementioned limitations and accurately detects and classifies new and complex attacks. Extensive experiments using the UNSW-NB15 benchmark dataset show that the proposed system demonstrates better accuracy and a lower FPR compared to advanced IDS [10].

3. RESEARCH ALGORITHMS

This section discusses the Radial Basis Function Neural Network (RBFNN), k-Nearest Neighbor (k-NN) classifier, Grasshopper Optimization Algorithm (GOA), and Principal Component Analysis (PCA).

3.1. Radial Basis Function Neural Network (RBFNN)

Neural networks have emerged as a practical technology that has been successfully applied in various fields. The main advantages of neural networks are their self-adaptive, self-organizing, and real-time operation capabilities. Radial Basis Function Neural Networks (RBFNNs) are widely used for non-parametric estimation of multi-dimensional functions from a limited set of training data. RBFNNs are particularly appealing due to their rapid and comprehensive training, drawing significant attention. RBFNNs can approximate any continuous function with any degree of accuracy. Notably, these networks possess this capability with only one hidden layer. RBFNNs are inspired by statistical pattern classification techniques and have revitalized themselves as a type of neural network, primarily benefiting the classification of patterns in non-linear spaces [11].

In an RBFNN, the input layer serves solely as an input layer without any processing. The second layer, or hidden layer, establishes a non-linear mapping between the input space and a typically higher-dimensional space, playing a crucial role in transforming non-linear patterns into linearly separable patterns. Finally, the third layer produces a weighted sum along with a linear output. If this neural network is used for function approximation, such an output will be useful, but if pattern classification is needed, a hard limiter or a sigmoid function can be applied to the output neurons to produce output values of 0 or 1.

As described, the unique characteristic of this network lies in the processing performed in the hidden layer. The function of the hidden layer is defined by the relationship:

$$f(x) = \sum_{i=1}^{p} w_i \varphi(Xc_i - x) \quad (1)$$

where p is the number of radial functions, w_i is the weight of the ith neuron, X_{ci} is the centroid of the i-th neuron, x is the input vector, and φ is the radial function. This relationship indicates that to approximate the function f, p radial functions with centroids X_{ci} are used. The notation $\|.\|$ represents the distance function in the space Rn, usually chosen as the Euclidean distance.

Since the curve of the radial basis functions is radially symmetric, the neurons in the hidden layer are known as radial function neurons. The well-known function in radial networks is the Gaussian or exponential function, defined as:

$$\phi(r) = e^{-\frac{r^2}{2\sigma^2}} \qquad c > 0 \qquad r \in R \quad (2)$$

where r=‖x-xc‖ is the width factor of the kernel. The reason for choosing the Gaussian exponential function as the response function of neurons in neural networks is that the exponential function is part of a group of functions with the best properties in approximation [12].

3.2. k-Nearest Neighbour Classifier

When attempting to solve new problems, people often refer to the solutions of similar problems that have been previously solved. The k-Nearest Neighbor (k-NN) is a classification technique that uses a version of this method. In this approach, the decision of which class a new sample belongs to is made by examining a number (k) of the most similar samples or neighbors. Among these k samples, the number of samples for each class is counted, and the new sample is assigned to the class to which most of the neighbors belong. Figure 1 shows the neighborhood range of sample N, where most neighbors are in class X.

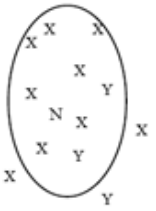

Fig.1. Neighborhood range of sample N

The first task in using k-NN is to find a measure of similarity or distance between the attributes in the data and calculate it. While this is straightforward for numerical data, categorical variables require special handling. Once the distance between different samples is calculated, the set of previously classified samples can be used as a basis for classifying new samples. The distance between two samples $X_i=(x_{i1},x_{i2},\ldots,x_{in})$ and $X_j=(x_{j1},x_{j2},\ldots,x_{jn})$ is calculated using the Euclidean distance as shown in the following equation [13]:

$$d(X_i, X_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{in} - x_{jn})^2} \quad (3)$$

3.3. Grasshopper Optimization Algorithm (GOA)

Grasshoppers are one of the largest groups among all organisms. The unique aspect of grasshopper groups is that their group-living behavior can be found in both adult and juvenile grasshoppers. Nature-inspired algorithms divide the search process into two phases: exploration and exploitation. During exploration, search agents are encouraged

to move abruptly, while during exploitation, they tend to move locally. These two functions, as well as target searching, are performed instinctively by grasshoppers. The mathematical model to simulate the group behavior of grasshoppers is shown by the following equation:

$$X_i=S_i+G_i+A_i \tag{4}$$

Where X_i defines the position of the *i*-th grasshopper, S_i denotes social interactions, G_i represents the gravitational force acting on the *i*-th grasshopper, and Ai signifies the horizontal wind force. Note that to illustrate stochastic behavior, this relationship can be rewritten as $X_i=r_1S_i+r_2G_i+r_3A_i$, where r_1, r_2 and r_3 are random numbers in the range (0,1). The distance between grasshoppers *i* and *j* is computed as $d_{ij}=|X_j-X_i|$, where *S* is a function defining the strength of social forces, shown in Equation (5), and $d_{ij}=\frac{Xj-Xi}{d_{ij}}$ is a unit vector from grasshopper *i* to grasshopper *j*. The function *S*, which defines the social forces, is defined in Equation (5).

$$s(r)=fe^{\frac{-r}{l}}-e^{-r} \tag{5}$$

where f denotes the intensity of attraction and l represents the gravitational length scale. The component G in Equation (1) is computed as in Equation (6).

$$G_i=-ge_g \tag{6}$$

where g is the gravitational constant, and e_g is a unit vector directed towards the center of the Earth. The component A in Equation (4) is computed as in Equation (7).

$$A_i=-ue_w \tag{7}$$

where *u* is a drift constant and e_w is a unit vector in the direction of the wind. Juvenile grasshoppers lack wings; hence, their movement is strongly influenced by wind direction. By substituting *S, G* , and *A* into Equation (4), the relationship can be expanded as in Equation (8).

$$X_i=c\sum_{j\neq i}\frac{ub_d-lb_d}{2}S(|X_j-X_i|)\frac{X_j-X_i}{d_{ij}}+T_d \tag{8}$$

where ub_d is the upper bound in the d-th dimension, lb_d is the lower bound in the d-th dimension, T^d is the value of the d-th dimension in the target (the best solution found so far), and c is a reduction coefficient to shrink the comfort zone, repulsion zone, and attraction zone. To balance exploration and exploitation, parameter c should decrease proportionally with the number of iterations. This mechanism increases the number of interactions in exploitation. The coefficient c reduces the comfort zone proportionally to the number of interactions and is calculated as in Equation (9).

$$c=c_{max}\frac{c_{max}-c_{min}}{L} \tag{9}$$

where c_{max} is the maximum value, c_{min} is the minimum value, l represents the current iteration, and *L* is the maximum number of iterations [14].

3.4. Principal Component Analysis (PCA) Algorithm

Principal Component Analysis (PCA) is a type of statistical analysis that selects a smaller number of factors, known as principal components, from the initial set of variables. Consider an n-tuple of vectors, each having p elements. Each of these vectors can be represented as a point in a p-dimensional space. The PCA algorithm identifies orthogonal axes that best represent these n points. The number of these axes is less than or equal to the number of original variables. Initially, the PCA algorithm determines the closest axis to the data cloud. This criterion can also be expressed in a mathematically equivalent form. In other words, the variance of these projections is maximized. The second axis is determined similarly, with the condition that it is orthogonal to the first axis. Together, these two axes form a plane that best fits the data. This process continues to find all orthogonal axes (principal components). In the new space, there is no correlation between the variables. The first new variable contains the highest variance of the data, and the second variable contains the second highest variance not explained by the first variable and is orthogonal to it [15].

4. PROPOSED METHOD

The aim of this section is to present the research method and the process of training a Radial Basis Function (RBF) neural network using the Grasshopper Optimization Algorithm (GOA) as the proposed model. The research steps are depicted in Figure 2.

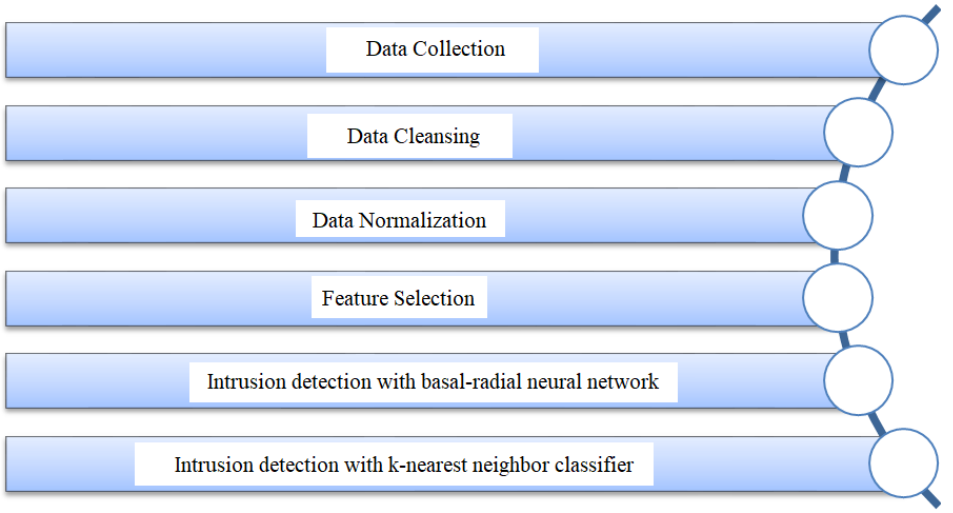

Fig.2. Intrusion Detection System in Cloud Computing Virtualization

4.1. Data Collection

In this research, data has been collected from the NSL-KDD database, comprising 1000 data samples with 17 features. 80% of the data is used for model training and 20% for model testing.

4.2. Data Cleaning

Data cleaning is performed when there are attributes in the database with missing values. In this research, the central statistic of the mean is used for data cleaning for samples with missing data [16].

4.3. Data Normalization

Due to the non-uniform range of feature variations and the different units of variables, larger values have a greater impact on the functions used, which does not necessarily mean they are more important. To address this issue, data normalization is performed. In this study, data is normalized to the range [-1, 1] using linear normalization:

$$X = 2 \times \frac{x - \min(x)}{\max(x) - \min(x)} - 1 \quad (10)$$

where min(x) is the minimum of the input vector x and max(x) is the maximum of the input vector x, and X is the normalized value [17].

4.4. Feature Selection

PCA is one of the methods for dimensionality reduction and feature selection. One of the significant applications of PCA is in classification. The PCA algorithm maps data from the input space to a new space where there is no dependency between features, and features are ordered from the highest to the lowest variance. Features with low variance are eliminated, and features with high variance are selected. In this research, using the PCA algorithm, 8 features with high variance out of the 17 existing features in the database are selected for more accurate classification.

4.5. Proposed Method Steps

Hybrid models, as their name suggests, combine computational intelligence techniques to leverage each other's capabilities to solve problems and improve solutions. In such systems, different approaches are either combined or used alongside each other. Typically, using one system within another helps cover each other's weaknesses, enhance their strengths, or both. Examples of such systems include neuro-evolutionary systems. The significant strength of neural systems in estimation is combined with the high-speed calculation of optimal parameters by evolutionary algorithms. The following describes the process of training an RBF neural network with the Grasshopper Optimization Algorithm (GOA). In an RBF neural network, the centers of neurons in the hidden layer, the values of the spread parameters, and the weights are unknowns determined during the training process. Assuming the RBF neural network includes two inputs and three neurons in the hidden layer, the general form of a grasshopper's position in the GOA is shown in Figure 3.

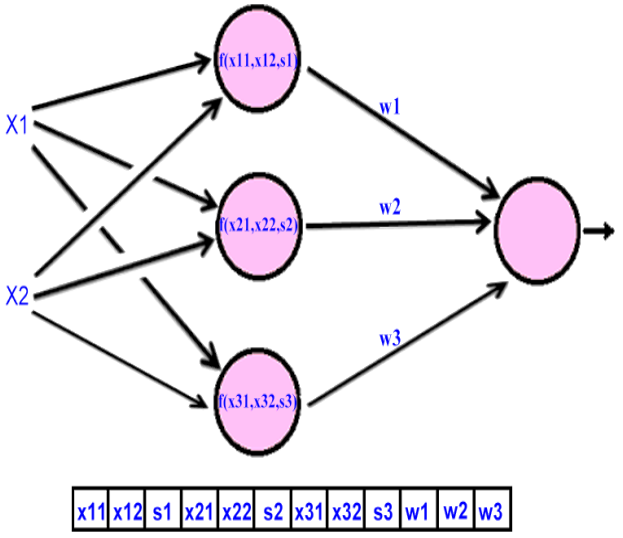

Fig.3. Radial Basis Function Neural Network

The Grasshopper Optimization Algorithm consists of two main phases: the initial preparation phase and the iteration phase, detailed below for training the RBF neural network.

4.6. Initial Preparation Phase

In this phase, a population of grasshoppers is created, each consisting of decision variables (position) and an objective function. The decision variables for each grasshopper include the centers, spread values, and weights of the RBF neural network, with the mean square error for the training data considered as the objective function. Figure 4 shows the position of a grasshopper representing the centers, spread values, and weights of the RBF neural network. In this phase, all grasshoppers' positions are initialized randomly, and the mean square error of the neural network is calculated for each population member.

4.7. Iteration Phase

In this phase, the following operations are repeated until the termination conditions are met:

1. Social Interaction Operator: In this step, the position of grasshopper i is determined based on the positions of other grasshoppers and the best grasshopper's position using the social interaction operator. Figure 4 shows the position of grasshopper i before and after social interaction. Based on the new position of the grasshopper, new values for the centers, spread parameters, and weights are placed in the RBF neural network. The training inputs are applied to the neural network, and the mean square error is calculated as the grasshopper's fitness.

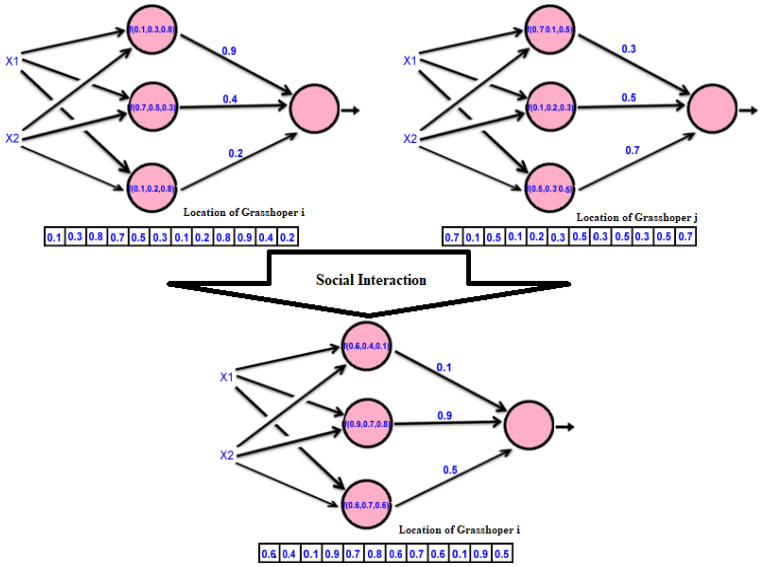

Fig.4. Social Interaction in the Grasshopper Optimization Algorithm

2. Updating the Damping Coefficient: In this step, the damping coefficient parameter is updated linearly.

These operations are repeated until the termination conditions are met. The output of the Grasshopper Optimization Algorithm is the best values for the centers, spread parameters, and weights in the RBF neural network, with the minimum possible mean square error.

5. SIMULATION RESULTS

The aim of this section is to examine the results of using an RBF neural network trained with the Grasshopper Optimization Algorithm (GOA) and the k-nearest neighbors (k-NN) classifier in detecting intrusions in cloud computing virtualization.

5.1. RBF Neural Network Results

This section discusses the results of the intrusion detection system in cloud computing virtualization using an RBF neural network trained by the GOA for training, testing, and all data. In the GOA, the population size is set to 50, and the maximum number of algorithm iterations is 500. Table 1 presents the results of using an RBF neural network trained by the GOA in the intrusion detection system in cloud computing virtualization based on various error types and for training, testing, and all data. The mean square error for the training data is 0.15, for the testing data is 0.14, and for all data is 0.148.

Table 1. Results of RBF Trained with GOA in Intrusion Detection in Cloud Computing Virtualization

Data	MSE	RMSE	MAE	SSE
Train Data	0.15	0.387	0.075	**120**
Test Data	0.14	0.374	0.07	**28**
All Data	0.148	0.385	0.074	**148**

Table 2 shows the standard performance metrics in the intrusion detection system in cloud computing virtualization using an RBF neural network trained with the GOA for training, testing, and all data. The accuracy for training data is 96.3%, for testing data is 96.5%, and for all data is 96.3%.

Table 2. Standard Performance Metrics in RBF Trained with GOA in Intrusion Detection in Cloud Computing Virtualization

Data	Sn	Sp	PPV	NPV	P
Train Data	93.3	99.5	99.5	93.1	**96.3**
Test Data	94.6	98.9	99.1	93.6	**96.5**
All Data	93.6	99.4	99.4	93.2	**96.3**

Figure 5 shows the error histogram for training and testing data in the RBF neural network trained with the GOA. The x-axis represents the error, and the y-axis represents the frequency of errors. The error histogram for training data is displayed in blue, and for testing data in green. Most training and testing data have zero error.

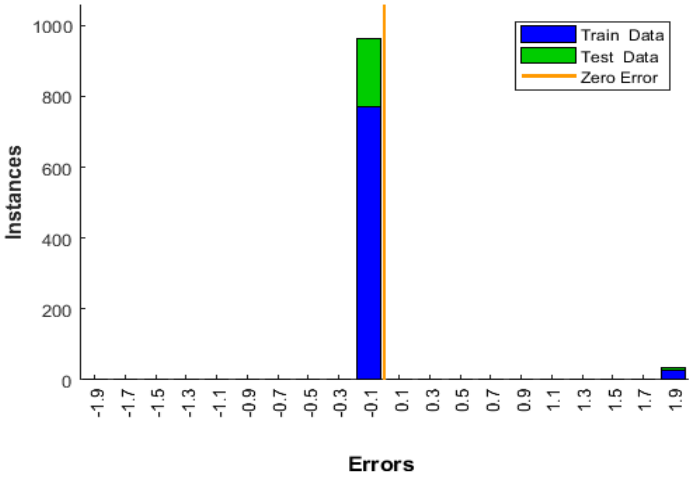

Fig.5. Error Types in RBF Trained with GOA

Figure 6 shows the ideal outputs with red circles and the outputs of the RBF neural network trained using the GOA with blue squares for the training data. The closer the network outputs are to the ideal outputs, the lower the detection error.

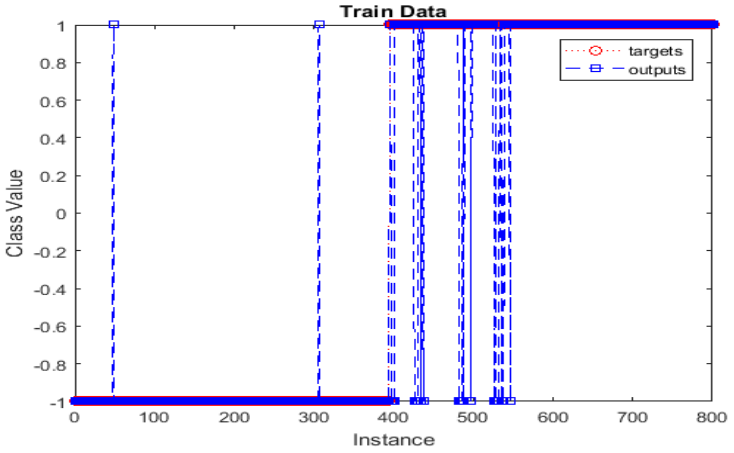

Fig.6. Ideal Outputs and Outputs of RBF Neural Network Trained with GOA for Training Data

Figure 7 shows the ideal outputs with red circles and the outputs of the RBF neural network trained using the GOA with blue squares for the testing data. The closer the network outputs are to the ideal outputs, the lower the detection error.

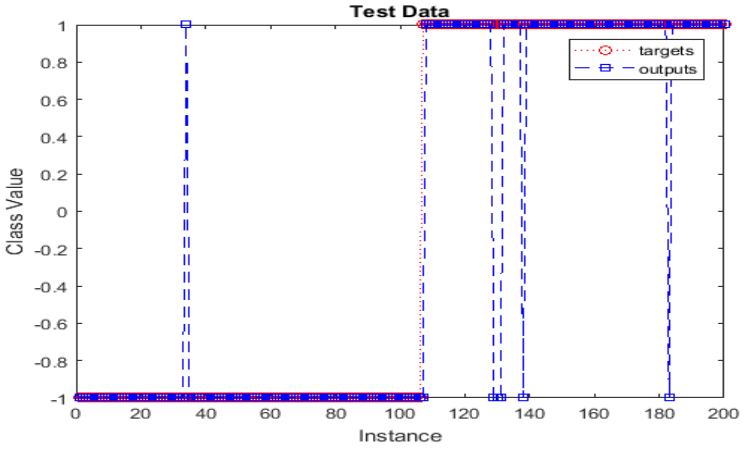

Fig.7. Ideal Outputs and Outputs of RBF Neural Network Trained with GOA for Testing Data

Figure 8 illustrates the ideal outputs marked with red circles and the outputs of the Radial Basis Function (RBF) neural network trained using the Grasshopper Optimization Algorithm (GOA) marked with blue squares for the entire dataset. The closer the outputs of the neural network are to the ideal outputs, the lower the error in detection.

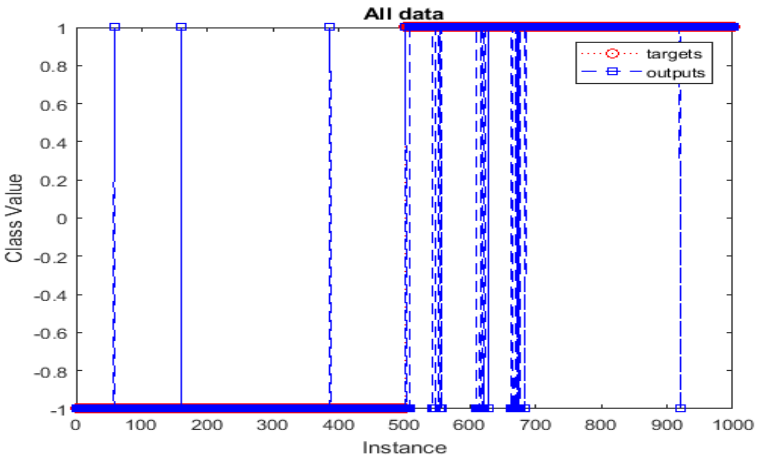

Fig.8. Graph of Target and Model Outputs for the Entire Data Set in RBF Trained with the Grasshopper Optimization Algorithm

5.2. Results of Using k-Nearest Neighbors (k-NN)

In the application of k-Nearest Neighbors (k-NN), the value of k is considered to be 3 to perform intrusion detection in cloud computing virtualization. Table 3presents the results of using k-NN for intrusion detection in cloud computing virtualization in terms of various error types and for training data, test data, and the entire dataset. The mean squared error (MSE) for the training data is 0.315, for the test data is 0.38, and for the entire dataset is 0.328.

Table 3. Results of Using k-Nearest Neighbors for Intrusion Detection in Cloud Computing Virtualization

Data	MSE	RMSE	MAE	SSE
Train Data	0.315	0.561	0.158	**252**
Test Data	0.38	0.616	0.19	**76**
All Data	0.328	0.573	0.164	**328**

Table 4 shows the standard performance metrics in the intrusion detection system in cloud computing virtualization using k-Nearest Neighbors for the training data, test data, and the entire dataset.

Table 4. Standard Performance Metrics in k-Nearest Neighbors for Intrusion Detection in Cloud Computing Virtualization

Data	Sn	Sp	PPV	NPV	P
Train Data	86.4	100	100	84.3	**92.1**
Test Data	84.2	100	100	80.8	**90.5**
All Data	85.9	100	100	83.6	**91.8**

Figure 9 depicts the error graph for the training data, test data, and the entire dataset in k-Nearest Neighbors. The horizontal axis represents the error value, and the vertical axis represents the error frequency. The histogram of the training data errors is shown in green, and the histogram of the test data errors is also shown in green. Most of the training and test data errors are zero.

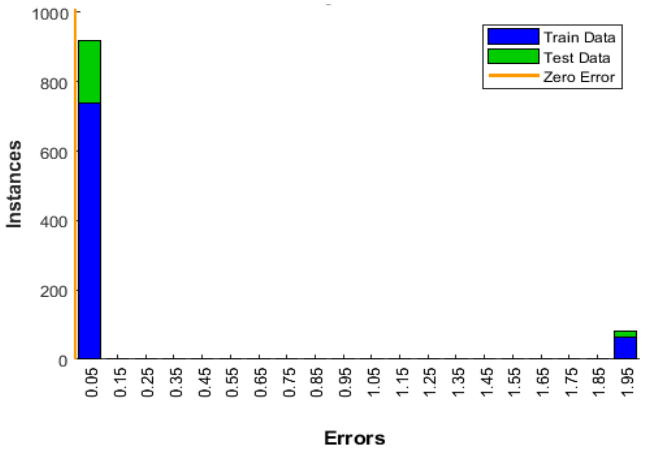

Fig.9. Types of Errors in k-Nearest Neighbors for Intrusion Detection in Cloud Computing Virtualization

Figure 10 shows the ideal outputs marked with red circles and the outputs of the k-Nearest Neighbors classifier marked with blue squares for the training data. The closer the outputs of the k-Nearest Neighbors classifier are to the ideal outputs, the lower the detection error. Class 1 is considered as an attack, and Class -1 is considered as a non-attack class.

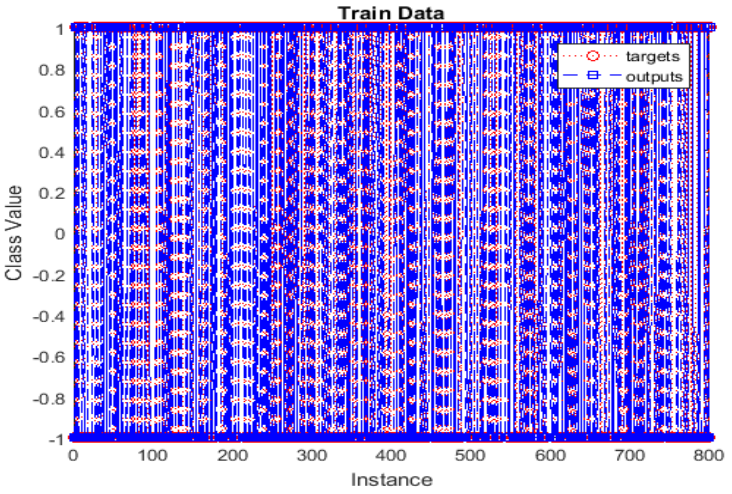

Fig.10. Graph of Target and k-Nearest Neighbors Outputs for Training Data in Intrusion Detection in Cloud Computing Virtualization

Figure 11 illustrates the ideal outputs marked with red circles and the outputs of the k-Nearest Neighbors classifier marked with blue squares for the test data. The closer the outputs of the k-Nearest Neighbors classifier are to the ideal outputs, the lower the detection error.

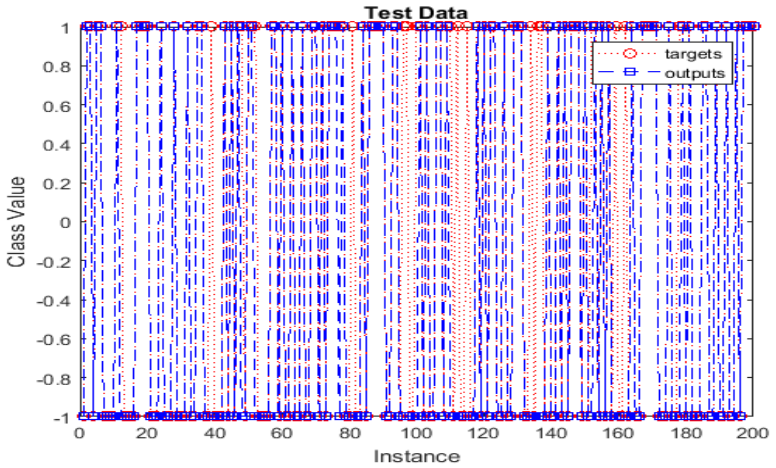

Fig.11. Graph of Target and k-Nearest Neighbors Outputs for Test Data in Intrusion Detection in Cloud Computing Virtualization

Figure 12 shows the ideal outputs marked with red circles and the outputs of the Bayesian classifier marked with blue squares for the entire dataset. The closer the outputs of the k-Nearest Neighbors classifier are to the ideal outputs, the lower the detection error.

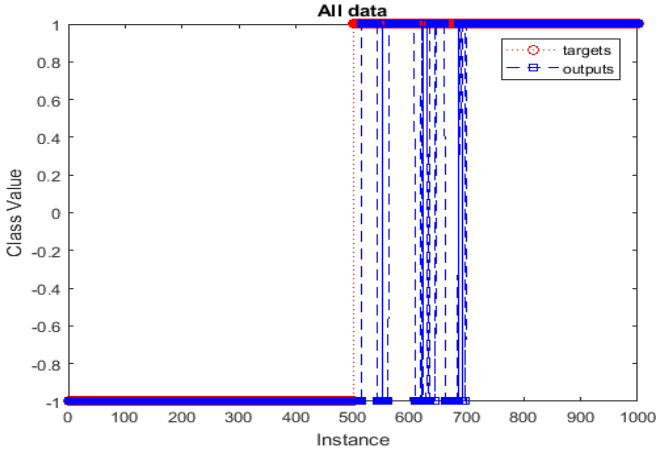

Fig.12. Ideal and k-Nearest Neighbors Outputs for Test Data in Intrusion Detection in Cloud Computing Virtualization

6. CONCLUSION

The focus of this paper is on intrusion detection in cloud computing virtualization. Therefore, in this study, we first collected data from the intrusion detection system in cloud computing virtualization, cleaned the data using central statistics like the mean, normalized the data linearly, and selected features using Principal Component Analysis (PCA), reducing the features from 17 to 8. Then, the RBF neural network was trained using the Grasshopper Optimization Algorithm, and the results were compared with the k-Nearest Neighbors classifier based on various error types and standard performance metrics. The results indicate that the RBF neural network trained with the Grasshopper Optimization Algorithm performed better with a mean squared error of 0.148 and an accuracy of 96.3%.

Declaration

We acknowledge that we used ChatGPT to enhance the academic writing of our manuscript while ensuring the originality and integrity of our work.

Transparency Statement

The data supporting this study are available upon reasonable request to the corresponding author, subject to ethical and confidentiality considerations.

Acknowledgments

We would like to express our gratitude to all individuals who contributed to this project.

Declaration of Interest

The authors declare that they have no competing interests.

Funding

This research received no specific grant from any funding agency, commercial, or not-for-profit sectors.

REFERENCES

[1] Almomani, A., Alauthman, M., Albalas, F., Dorgham, O., & Obeidat, A. (2020). An online intrusion detection system to cloud computing based on NeuCube algorithms. In Cognitive Analytics: Concepts, Methodologies, Tools, and Applications (pp. 1042-1059). IGI Global. https://doi.org/10.4018/978-1-7998-2460-2.ch053

[2] Aldribi, A., Traoré, I., Moa, B., & Nwamuo, O. (2020). Hypervisor-based cloud intrusion detection through online multivariate statistical change tracking. Computers & Security, 88, 101646. https://doi.org/10.1016/j.cose.2019.101646

[3] Abusitta, A., Bellaiche, M., Dagenais, M., & Halabi, T. (2019). A deep learning approach for proactive multi-cloud cooperative intrusion detection system. Future Generation Computer Systems, 98, 308-318. https://doi.org/10.1016/j.future.2019.03.043

[4] Nikolai, J., & Wang, Y. (2014). Hypervisor-based cloud intrusion detection system. In Proceedings of the 2014 International Conference on Computing, Networking and Communications (ICNC). https://doi.org/10.1109/ICCNC.2014.6785472

[5] Iqbal, S., Kiah, M. L. M., Dhaghighi, B., Hussain, M., Khan, S., Khan, M. K., & Choo, K.-K. R. (2016). On cloud security attacks: A taxonomy and intrusion detection and prevention as a service. Journal of Network and Computer Applications, 74, 98-120. https://doi.org/10.1016/j.jnca.2016.08.016

[6] Ganeshkumar, P., & Pandeeswari, N. (2016). Adaptive neuro-fuzzy-based anomaly detection system in cloud. International Journal of Fuzzy Systems, 18(3), 367-378. https://doi.org/10.1007/s40815-015-0080-x

[7] Modi, C. N., & Acha, K. (2017). Virtualization layer security challenges and intrusion detection/prevention systems in cloud computing: A comprehensive review. The Journal of Supercomputing, 73(3), 1192-1234. https://doi.org/10.1007/s11227-016-1805-9

[8] Hatef, M. A., Shaker, V., Jabbarpour, M. R., Jung, J., & Zarrabi, H. (2018). HIDCC: A hybrid intrusion detection approach in cloud computing. Concurrency and Computation: Practice and Experience, 30(3), e4171. https://doi.org/10.1002/cpe.4171

[9] Hajimirzaei, B., & Navimipour, N. J. (2019). Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm. ICT Express, 5(1), 56-59. https://doi.org/10.1016/j.icte.2018.01.014

[10] Sethi, K., Kumar, R., Prajapati, N., & Bera, P. (2020). Deep reinforcement learning based intrusion detection system for cloud infrastructure. In Proceedings of the 2020 International Conference on COMmunication Systems & NETworkS (COMSNETS). https://doi.org/10.1109/COMSNETS48256.2020.9027452

[11] Howlett, R. J., & Jain, L. C. (2013). Radial Basis Function Networks 2: New Advances in Design. Physica-Verlag HD.

[12] Biancolini, M. E. (2018). Fast Radial Basis Functions for Engineering Applications. Springer International Publishing. https://doi.org/10.1007/978-3-319-75011-8

[13] Akbulut, Y., Sengur, A., Guo, Y., & Smarandache, F. (2017). Ns-k-nn: Neutrosophic set-based k-nearest neighbors classifier. Symmetry, 9(9), 179. https://doi.org/10.3390/sym9090179

[14] Saremi, S., Mirjalili, S., & Lewis, A. (2017). Grasshopper optimisation algorithm: Theory and application. Advances in Engineering Software, 105, 30-47. https://doi.org/10.1016/j.advengsoft.2017.01.004

[15] Aït-Sahalia, Y., & Xiu, D. (2019). Principal component analysis of high-frequency data. Journal of the American Statistical Association, 114(525), 287-303. https://doi.org/10.1080/01621459.2017.1401542

[16] Bard, J. (2018). What is data mining? PowerKids Press.

[17] Yang, X. S. (2019). *Introduction to algorithms for data mining and machine learning*. Academic Press.

[18] Lee, S. K., Cho, Y. H., & Kim, S. H. (2010). Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations. Information Sciences, 180(11), 2142–2155. doi:10.1016/j.ins.2010.02.004

[19] Choi, K., Yoo, D., Kim, G., & Suh, Y. (2012). A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis. Electronic Commerce Research and Applications, 11(4), 309–317. doi:10.1016/j.elerap.2012.02.004

[20] Núñez-Valdéz, E. R., Cueva Lovelle, J. M., Sanjuán Martínez, O., García-Díaz, V., Ordoñez de Pablos, P., & Montenegro Marín, C. E. (2012). Implicit feedback techniques on recommender systems applied to electronic books. Computers in Human Behavior, 28(4), 1186–1193. doi:10.1016/j.chb.2012.02.001

[21] Avini, H. (2020). Improved recommender systems. Aftab Giti Publishing House.

[22] Maitham, J., & Qolizadeh, H. (2014). Improvement of recommender systems based on group refinement using social networks. In Proceedings of the 7th International Conference on Information Technology and Knowledge (Vol. 23). Urmia, Iran.