Journal of Computational Science & Intelligent Technologies, eISSN: 2582-9041

Volume 1, Issue 2, pages: 15–25, September 2020

A Hybrid Genetic-Neuro Algorithm for Cloud Intrusion Detection System

Suresh Adithya Nallamuthu

Gdansk University of Technology, Gdansk, Poland.

Corresponding Author: sureshadithya1991@gmail.com

Received: *10.06.2020,*
Revised: *10.07.2020,*
Accepted: *19.08.2020,*
Published: *31.09.2020*

DOI: *10.53409/mnaa.jcsit20201203*

Abstract: The security for cloud network systems is essential and significant to secure the data source from intruders and attacks. Implementing an intrusion detection system (IDS) for securing from those intruders and attacks is the best option. Many IDS models are presently based on different techniques and algorithms like machine learning and deep learning. In this research, IDS for the cloud computing environment is proposed. Here in this model, the genetic algorithm (GA) and back propagation neural network (BPNN) is used for attack detection and classification. The Canadian Institute for Cyber-security CIC-IDS 2017 dataset is used for the evaluation of performance analysis. Initially, from the dataset, the data are preprocessed, and by using the genetic algorithm, the attack was detected. The detected attacks are classified using the BPNN classifier for identifying the types of attacks. The performance analysis was executed, and the results are obtained and compared with the existing machine learning-based classifiers like FC-ANN, NB-RF, KDBN, and FCM-SVM techniques. The proposed GA-BPNN model outperforms all these classifying techniques in every performance metric, like accuracy, precision, recall, and detection rate. Overall, from the performance analysis, the best classification accuracy is achieved for Web attack detection with 97.90%, and the best detection rate is achieved for Brute force attack detection with 97.89%.

Keywords: *Intrusion Detection, Cloud Computing, Genetic algorithm, Back Propagation Neural Network, CIC-IDS*.

I. INTRODUCTION

Cloud computing has been introduced as a general term for describing a set of advanced on-demand services for computing given by commercial suppliers like Google, Microsoft and Amazon. It means the computing infrastructure model was considered as the "cloud," from which organizations, industries, and people accesses application on demand worldwide. The fundamental concept beneath this technology was providing storage, computing, and softwares "as a service." [1]. National Institute of Standards and Technology (NIST) describes Cloud computing as the "model for enabling universal, convenient, and on demand access of network to a configurable computing resources in a shared pool which could be quickly provided and delivered with minimum managerial efforts or service provider interactions" [2].

Cloud services were provided as Infrastructure-as-a-service (IaaS), Platform-as-a-service (PaaS), and Software-as-service (SaaS). It is a combination of many techniques, comprising distributed and grid computing, and service delivery network by using Internet. The public cloud platform is very complicated when contrasted with a conventional data center environment. Based on the model of Cloud computing, an institution or organization gives up direct access to significant features of security, provides a high level of trust over the Cloud provider [3-5].

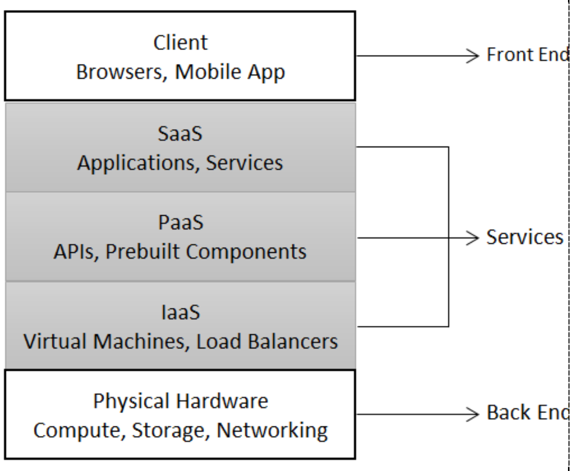

Figure.1. Cloud Computing Stack

In the Cloud system, distributed and shared resources makes it challenging to create a security system to ensure data privacy and security. Because of open issues, no Cloud service provider (CSP) accepts their users to use IDS or security techniques expanding into the management service layer behind virtualized Cloud instances. Users might not know about specific security-issues, vulnerabilities, or malware details. For example, the intruders might have the option to obtain the information of Cloud accounts and set up a kernel-level rootkit through back-channel [6]. Intrusions on "physical level" are like extracting the RAM of the virtualized host or undermining the virtualization layers were aware to the network. Indeed, the host system offering the information cannot be reliable entirely once the CSP holds the physical resource. CSPs regularly setup a SLA (Service Level Agreement) to feature the privacy and security of the relevant services [7].

Though numerous security complexities arise with the variation to this computing model, including IDS, however, the significant development of the data security innovations recently, attacks and intrusions keep on defeating existing IDSs in the Cloud environment. A disastrous DDoS attack recently has taken down over 70 significant Internet services, including Github, Amazon, Paypal, Twitter, and so on. Attackers have exploited IoT and Cloud Computing technologies to produce a large volume of traffic attack over 665 GB/s [8-9].

Intrusions and attacks have become a challenge to the existing Cloud IDSs through enormous amount of network traffic data, complex and dynamic actions, and new attack classes. IDS for Cloud must analyze network traffic data with vast volumes, identify the new attack actions effectively, and obtain higher accuracy with a low error rate [10].

In this research, a neural network-based technique Back Propagation Neural Network (BPNN) is proposed to classify the detected intrusions in the cloud environment and for detecting the intrusions the Genetic Algorithm (GA) is proposed. From the proposed dataset, the data are preprocessed and forwarded to detect the present anomalies by using the genetic algorithm. This detection enables the model to recognize and blocks suspicious data while granting accesses to general data. The anomaly data are then classified by using the BPNN classifier to detect each attack type present in the dataset.

II. RELATED WORK

IDS is a significant security tool used to secure the resources of the cloud. However, IDS frequently experience poor detection accuracy because of composed attacks like DDoS. Even though specific works have limitations and lack of methodology to decide a proper time to exchange attack data between nodes in the distributed IDS. In this way, N M Ibrahim and A Zainal proposed the distributed IDS that used an algorithm called binary segmentation change point detection, toward addressing the proper time frame to forward attack data to distributed IDS nodes and utilizing parallel Stochastic Gradient Descent with SVM (SGD-SVM) for accomplishing the distributed identification. This model was experimented in Apache Spark utilizing the NSL-KDD dataset [11].

The increasing demand for cloud computing causing it inclined to different attackers impacting the integrity and privacy of the information stored in cloud. The efficient IDS composed of a minimum time-compelling algorithm with minimum space complication and high accuracy. To make this, the total features were decreased while preserving less data loss. Partha Ghosh et al. proposed a feature selection model, in which the features were chosen based on mutual data gain between related attributes. To accomplish this, initially, the attributes as per the correlativity were grouped. Hence, from every group, the attributes with the most elevated mutual data gain in their relevant group were chosen. This resulted in a minimized feature set that provided fast learning and hence delivered great IDS that will secure the information in cloud [12].

Identification of attacks and intrusions through illegal users is perhaps the greatest challenge for both cloud service providers and users. Multi-Layer Perceptron Neural Networks and Particle Swarm Optimization were used by A.S Saljoughi et al. for identifying the attacks and intrusions in cloud computing. For optimizing the neural network, a combination of the neural network with the PSO was used to extricate optimal weights and attempted to decrease the time complications via training the network with random weights. The K-fold technique was utilized and selected a random group for result analysis. The model is assessed with KDDcup99 and NSL-KDD datasets [13]. Another combination of algorithms like MLP network, artificial bee colony, and fuzzy clustering were presented by B Hajimirzaei and N J Navimipour for detecting intrusions in the cloud environment [14].

Mohamed Idhammad et al. presented a distributed IDS based on machine learning for the Cloud environment. The model was developed to be embedded in the Cloud, along with the cloud provider's edge network elements. This enables intercepting incoming network traffic to the physical layer's edge network routers. The captured information was then preprocessed and forwarded to the initial detection of the anomaly process utilizing a Naive Bayes method. This enables detecting and blocking speculated traffic while granting normal traffic access. The speculated traffic at every network router part was synced to a main server. The Random Forest classifier was utilized for classifying the network traffic information accessible on the server and detects each attack type. This IDS model was performed on the Google Cloud and evaluated by the dataset CIDDS-001 [15].

A hybrid model composed of FCM and SVM was proposed by A.N Jaber and S.Ul Rehman. This hybrid technique was used as an IDS model to identify intrusions and attacks in a cloud computing environment. This model was separated into three stages. The primary stage presented the FCM clustering module that was utilized to divide huge dataset into small cluster for allowing the SVM to learn viably in an ideal way. The fuzzy model improved the SVM's performance. In the next stage, various SVM models were trained as per the allocated cluster values. Finally, the fuzzy aggregation model combined the results of the hypervisor inspector [16].

Edge computing expands conventional services of cloud to the network edge, and the exceptionally heterogeneous and dynamic condition at the network edges creates the security of networks circumstance confronting extreme difficulties. H.Yin et al. analyzed the improved k-dependency bayesian network technique that defined the trust relations between system elements and decreased the difficulty of the BN structure by lessening the coordinated weak dependence edges. By presenting a virtual augmentation technique and the maximum a posterior (MAP) criterion for small category samples, this classification model for intrusion detection based on improved KDNB was developed. The results were assessed using just 10% of the KDDCup99 dataset. This model solved the issues of poor stability and low detection accuracy [17].

III. PROPOSED METHODOLOGY

In this proposed model, the Genetic algorithm and BPNN classifier algorithms are used for detecting the intrusions and attacks in the cloud computing environment.

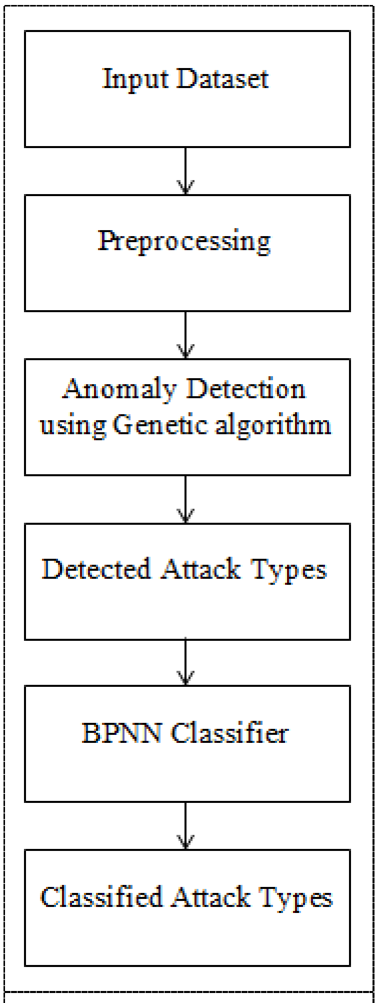

Figure.2. Proposed Method

Initially, the dataset is given as input to the system and preprocessed. For detecting the attacks and intrusions, CICIDS 2017 dataset is used. The preprocessed data forwarded to the initial anomaly detection using a GA, and the results from the GA are fed to the BPNN classifier. These results from the genetic algorithm are used to detect the intrusions and attacks present in the dataset as malignant and normal data. The BPNN classifier is used to classify the types of attacks.

3.1 Genetic Algorithm

The GA updates the population of individuals iteratively. By using fitness function, the individuals are assessed during every iteration process. A new population generation is acquired from the current generation by probabilistically choosing fitter individuals. Some of the individuals were allowed unchanged to the next generation. Others are referred to genetic operators like mutation and crossover to make new offspring. *g* is the current generation; *n* was the total individual in the populations; *Z* was the population fraction to be substituted by a crossover in every iteration, and *μ* was the mutation rate.

```
//Initialize generation
$g$=0;
$P_g$ = $n$ randomly produced individuals population;
//Calculate $P_g$
Calculate fitness ($f$) for every $f \in P_g$;
Do
{       //Create generation $g$+1
        //1. Copy
        Choose (1-$Z$) × $n$ individuals of $P_g$ and insert into $P_{g+1}$;
        //2. Crossover
        Choose $Z$ × $n$ individuals of $P_g$; pair them up; generate offspring; insert offspring into $P_{g+1}$;
        //3. Mutate
        Choose μ × $n$ individuals of $P_{g+1}$; invert a randomly chosen bit in each;
        //Calculate $P_{g+1}$
        Calculate fitness ($f$) for every $f \in P_g$;
        //Increment
        $g$= $g$+1;
}
while the fitness of the fittest individual in $P_g$ is not enough high;
return the fittest individual from $P_g$;
```

3.2 Back Propagation Neural Network

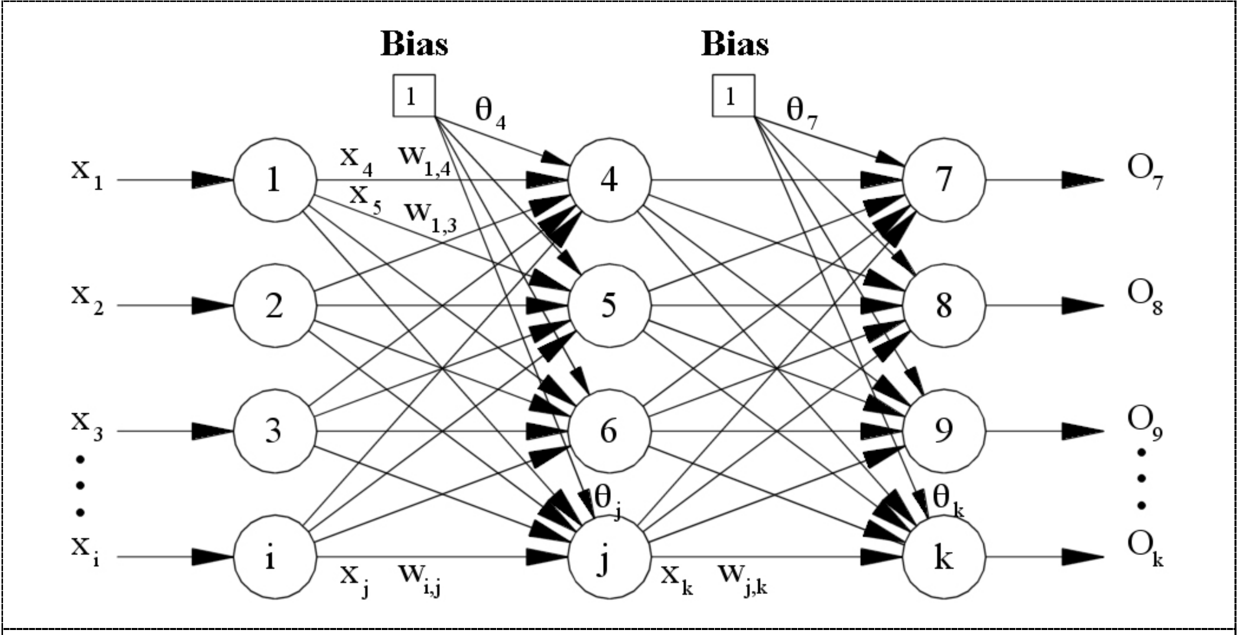

Figure.3. BPNN Architecture

The BPNN is simple, fast, and simple to program. It has no parameters to tune separated from the number of inputs. It is an adaptable technique as it does not need prior information about the network. It is a standard technique that usually performs well. It does not need any specific indications of the features of the function to be learned. Backpropagation is a short term of "backward propagation of errors." It is a standard technique of training ANN [18]. BPNN contains the input, hidden, and output layer, a normal multi-layer network regulated by the complete interconnection within layers but independent of integration in the same cell layer. The process of learning includes forward and backpropagation. The error resulting may be due to the connection weight abnormality and threshold within the connection layer nodes, thereby measuring the error value of the connecting node and changing it according to the connection weight [19].

Backpropagation refines the network structure by removing weighted links that minimally affect the trained network. It is the technique for adjusting the neural network's weight dependent on the error value acquired in the prior iteration. Proper weight tuning enables to minimize error rates and to make the system dependable by expanding its generalization. The inputs X arrive through the pre-connected path. The input was modeled using weight W. The weights are usually selected in random. The outputs for all neurons from the input layer to the hidden and output layers are calculated. To compute the error in the output, the following condition is used,

Error=Actual output - Desired output

Then switch to the hidden layer from the output layer to adjust the weights, thus reducing the error. This operation is repeated until it obtains the desired outcome. BPNN architecture is depicted in figure.3. It comprises of three layers. When the input was fed with a specific training pattern, the weighted input sum for node *j* in the hidden layer was determined by,

$$Net_j = \sum w_{i,j} x_j + \Theta_j \qquad (1)$$

Condition (1) was to compute the neuron's total input. The ϴj expression from a bias node, the weighted value, consistently presented output as one. This bias node was deemed as the "pseudo input" for every neuron present in the output and hidden layers and also utilized for solving the issues identified with circumstances where the input pattern is 0. If an input pattern gets zero value without a bias node, the NN cannot be trained.

To determine whether the neuron must fire, the "Net" term, otherwise called the action potential, was forwarded to a suitable actuation process. The activation function's output value decides the result of neurons and turns into the input to neurons in the following layers associated with it. Hence the differentiable activation function is a prerequisite for the BP algorithm. The sigmoid function was utilized as a standard activation function.

$$O_j = x_k = \frac{1}{1+e^{-Net_j}} \qquad (2)$$

Likewise, conditions (1) and (2) were utilized to decide the value of output for node k in the output layer.

Output Layer

If the output node's actual activation value, k, is O_k, and for node k's anticipated target output was t_k, the dissimilarity among the real and the anticipated output was presented by:

$$\Delta_k = t_k - O_k \qquad (3)$$

For node k, the error signal in the output layer computed as

$$\delta_k = \Delta_k O_k (1 - O_k) \qquad (4)$$

or

$$\delta_k = (t_k - O_k) O_k (1 - O_k)$$

$O_k(1-O_k)$ is the Sigmoid function derivative. Using delta rule, the weight correlation difference of the node *k* and *j* is determined by the *j* actuation in relation to the *k* error.

The equations for changing the weight, $w_{j,k}$, among the *j* and *k* are:

$$\Delta w_{j,k} = l_r \delta_k x_k \qquad (5)$$

$$w_{j,k} = w_{j,k} + \Delta w_{j,k} \qquad (6)$$

Where, $\Delta w_{j,k}$ was the modification in the weight among *k* and *j*, l_r was the learning rate. l_r is generally a smaller constant, which represents the relevant changes in weight. It must be noticed, in condition (5), the x_k was the value of input for *k*, and also a similar output value from *j*. For enhancing the way toward weights updating, a change to condition (5) was

$$\Delta w_{j,k}^{n} = l_r \delta_k x_k + \Delta w_{j,k}^{(n-1)} \mu \qquad (7)$$

The update of weight during the nth epoch was specified through adding the (μ) momentum term, multiplied to $\Delta w_{j,k}$ (n-1)th epoch.

Hidden Layer

For *j*, the error signal in this layer computed as

$$\delta_k = (t_k - O_k) O_k \sum (w_{j,k} \delta_k) \qquad (8)$$

Likewise, the equation to modify the $w_{i,j}$, within the *j* and *i* is

$$\Delta w_{j,k}^{n} = l_r \delta_j x_j + \Delta w_{j,k}^{(n-1)} \mu \qquad (9)$$

$$w_{i,j} = w_{i,j} + \Delta w_{i,j} \qquad (10)$$

Global Error

At last, the BP was acquired by presuming, which was appropriate to reduce the output nodes error across the patterns introduced to the NN. The accompanying condition was utilized for computing the error function E, for every pattern:

$$E = {}^{1}/_{2} \sum (\sum (t_k - O_k)^2) \qquad (11)$$

Technically, when the NN has been trained adequately, the error function must be a zero value.

3.3. BPNN Algorithm

```
Allot every network input and output
Initialize each weight with lower numbers randomly, generally among - 1 and 1
repeat
  for each pattern in the training set
    Present the network pattern
//Propagate the input forward through the network:
      for every layer in the network
        for each node in the layer
          1. Evaluate the input's weighted sum for the node
          2. Add the threshold to the sum
          3. Evaluate the activation for the node
        end
      end
//Propagate the errors backward within the network
      for each output layer node
        evaluate the error signal
      end
      for every hidden layer
        for each layer node
          1. Evaluate the error signal of the node
          2. Update weight for every node in the network
        end
      end
//Compute Global Error
      Compute the Error Function
end
while ((maximum number of iterations < than determined) AND
      (Error Function is > than determined))
```

From the genetic algorithm, the detected attacks are classified using the BPNN classifier. The experiments performed in the Amazon Cloud Platform with 8 cores and 32 GB of memory. The dataset CICIDS 2017 was used for evaluating the proposed model's performance analysis.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

4.1 Dataset Description

For detecting the attacks and intrusions, CICIDS 2017 dataset is used. Commonly, various DDoS attack datasets have many constraints like out of relevant data, a redundancy that is variable. This proposed data set has network identical data. The data set was accumulated for five days with different attacks and also normal data. It has the network data with and without attacks, which was approximately the real data of the network [20]. This dataset was uneven, hence this dataset with the duplicating method as it basically affects the training of the proposed model, and then the testing was performed. This research was experimented using Keras on the Tensorflow package on 64-bit Intel Core-i5 processor with 8 GB RAM in Windows 8 system. The algorithms are executed in MATLAB.

Table.1. Class Labels of Dataset with Instances

Class Labels	Number of instances
Bot	1966
BENIGN	2359087
DDoS	41835
DoS Hulk	231072
DoSGoldenEye	10293
DoSslowloris	5796
DoSSlowhttptest	5499
FTP-Patator	7938
Heartbleed	11
Infiltration	36
PortScan	158930
SSH-Patator	5897
Web Attack – Brute Force	1507
Web Attack – SQL Injection	21
Web Attack – XSS	652

4.2. Attack Types

Bot: The attacker use Trojans to breach the security of numerous victim systems, accepting accountability for those systems and set every system in Bot network, which could be utilized and accessed by the intruders remotely.

Benign: Normal traffic action.

DDoS: The attacker uses various systems that works together to perform an attack on one victim system.

DoS Hulk: The intruder use the HULK tool for completing the DoS attack on the web server for making volumes of various and disordered traffics. Likewise, the generated traffics can avoid caching engines and attacks the resource pool of server.

DoSGoldenEye: The intruder uses the GoldenEye tool to implement a DoS attack.

DoSslowloris: The intruder uses the Slow Loris tool to implement the DoS attacks.

DoSSlowhttptest: The intruder utilize the HTTP Get request for outperform the total HTTP links allowed on the server, hindering different clients from incoming and offering the intruders the option to allow several HTTP links with the equivalent server.

FTP-Patator: The intruder uses this to implement the brute force attack for discovering the FTP access details.

Heartbleed Attack: The intruder utilizes the protocol OpenSSL to install malicious data inside OpenSSL memory, allowing the intruder with illegal access to significant data.

Infiltration: The intruder uses the infiltration procedure and softwares to penetrate and acquire unauthorized login to the networked system data.

PortScan: The intruder try to accumulate informations related to the victim system such as the OS details and services running over the packet forwarding with various destinations.

SSH-Patator: The intruder uses this to implement the brute force attack to discover the SSH access details.

Web Attack – Brute Force: The intruder tries to acquire significant information, like PIN and Password using trail-and-error.

Web Attack–SQL Injection: It was a technique of code injection used to attack applications by the data, along with odious SQL proclamation, which were installed inside a part for execution.

Web Attack – XSS: The attacker inject with commonly trusted internet-sites and benign using the web application which redirects to malicious content.

4.3. Performance Metrics

The accuracy was just the performance subset of the model. It is one of the performance metrics for evaluating the classification techniques. The accuracy computation is calculated using the following expression.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad (12)$$

Precision is a positive predictive rate. It is the ratio of properly predicted positive observation to the total predicted positive value. The calculation of precision is calculated using the following expression:

$$Precision = \frac{TP}{TP+FP} \qquad (13)$$

The recall is also termed as sensitivity. It is the ratio of properly predicted positive value to the each observation in the actual class. The computation of recall is calculated using the following expression:

$$Recall = \frac{TP}{(TP+FN)} \qquad (14)$$

The level of intrusion instances are represented by detection rate. It represents the total proper positive class predictions made as the proportion of all the predictions made. The calculation of the DR is computed using the accompanying condition:

$$DR = \frac{TP}{TP+FN} \qquad (15)$$

TP: true positive, FP: false positive, FN: false negative, TN: true negative.

From the dataset, the classes of minority attack having similar behavior and features are combined. Hence combining similar classes, the class, and the predominance ratio of various attack labels seems to be improved. As shown in the table.1, the ratio of Benign class was significant part with 83.34% and where the least attack class was Heartbleed with 0.00039%.

Table.2. Performance Analysis of Detecting Normal Attack (Benign)

Technique	Accuracy	Recall	Precision	Detection Rate
FC-ANN	90.44	89.31	89.25	89.38
NB-RF	92.15	90.12	90.05	90.24
KDBN	94.60	92.58	91.34	93.10
FCM-SVM	96.21	94.24	93.88	94.04
GA-BPNN	97.58	97.98	95.40	96.16

With this significant variation in ratio value, the primary detectors could determine benign. The analysis of the GA-BPNN was computed and compared with various detection methods like FC-ANN (Fuzzy Clustering Artificial Neural Network), NB-RF (Naive Bayes-Random Forest), KDBN (K-Dependency Bayesian Network) and FCM-SVM (Fuzzy C-Means with Support Vector Machine) as represented in table.2. The accuracy and detection rate achieved by the proposed method is 1.3% to 7% higher and 2.1% to 6.7% higher for benign attacks.

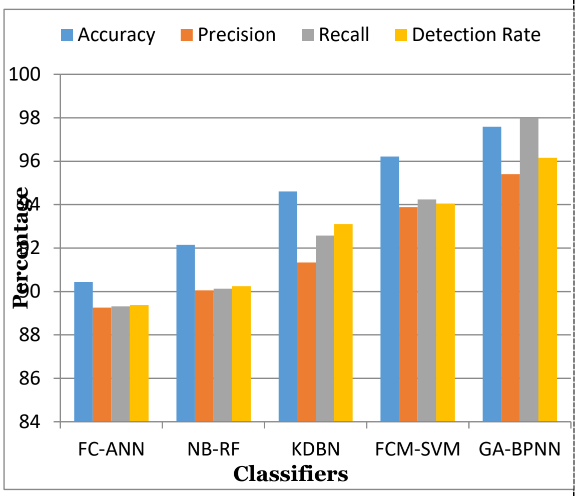

Figure.4. **Graphical Plot of Performance Metrics of Benign Detection**

The Bot class is called as Botnet ARES. This class contains 1966 instances with the ratio of 0.06%. As shown in the table.3, the proposed method achieved 1.6% to 6.7% more accuracy and 2.3% to 7.4% more detection rate than other techniques for Botnet attacks.

Table.3. **Performance Analysis of Detecting Botnet Attack**

Techniq ue	Accurac y	Reca ll	Precisio n	Detectio n Rate
FC-ANN	90.25	90.40	89.14	90.05
NB-RF	92.55	91.84	91.01	90.78
KDBN	93.97	93.25	92.42	93.89
FCM-SVM	95.33	94.99	93.98	95.12
GA-BPNN	96.98	96.38	95.16	97.50

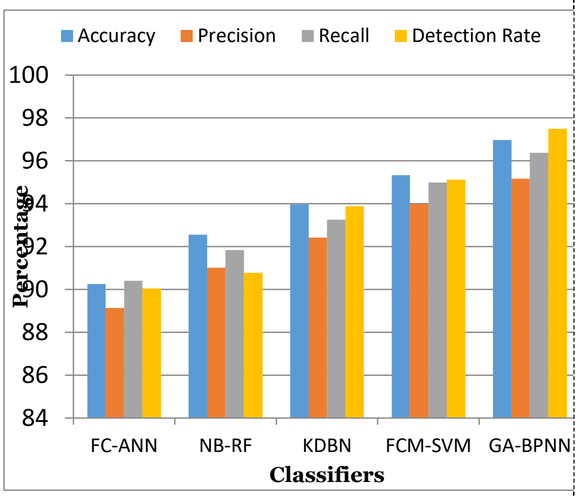

Figure.5. **Graphical Plot of Performance Metrics of Botnet Detection**

Table.4. **Performance Analysis of Detecting Brute Force Attack (FTP & SSH-Patator)**

Techniq ue	Accurac y	Reca ll	Precisio n	Detectio n Rate
FC-ANN	90.51	91.04	90.01	91.43
NB-RF	91.34	92.01	90.56	92.24
KDBN	92.97	93.25	91.33	94.00
FCM-SVM	94.64	95.11	93.80	96.18
GA-BPNN	96.89	96.95	95.45	97.89

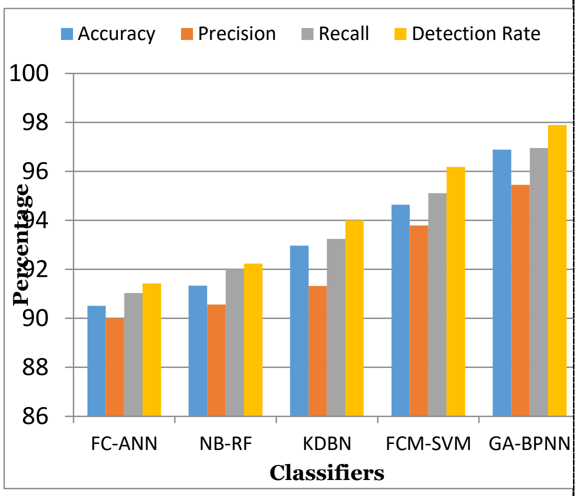

Figure.6. **Graphical Plot of Performance Metrics of Brute Force Detection**

The SSH-Patator and FTP-Patator classes are united as Brute Force class. Because both the classes have similar features and behavior, by uniting both the classes, a new class was formed with 13835 instances with 0.48% ratio. For this attack type, the proposed method's accuracy is 2.2% to 6.3% higher, and the detection rate is 1.7% and 6.4% higher than other methods, as shown in table.4.

The DoS/DDoS is a new class, which is the combinations of DDoS, DoSHulk, DoSGoldenEye, DoSslowloris, DoSSlowhttptest, and Heartbleed. By combining all these labels, it contains 294506 instances with 10.4% ratio. The GA-BPNN method achieved 0.6% to 4.9% higher accuracy and 1.2% to 6.6% higher detection rate than other techniques for this DoS/DDoS attack detection, as shown in table.5.

Table.5. **Performance Analysis of Detecing Dos/DDos Attack ("DDoS, Dos Hulk, DosGoldenEye, DoSslowloris DoSSlowhttptest, & Heartbleed")**

Techniq ue	Accurac y	Reca ll	Precisio n	Detectio n Rate
FC-ANN	92.26	92.65	91.50	91.05
NB-RF	93.40	93.97	92.04	93.71
KDBN	95.01	95.55	94.22	95.16
FCM-SVM	96.55	97.11	95.07	96.40
GA-BPNN	97.17	98.10	96.00	97.65

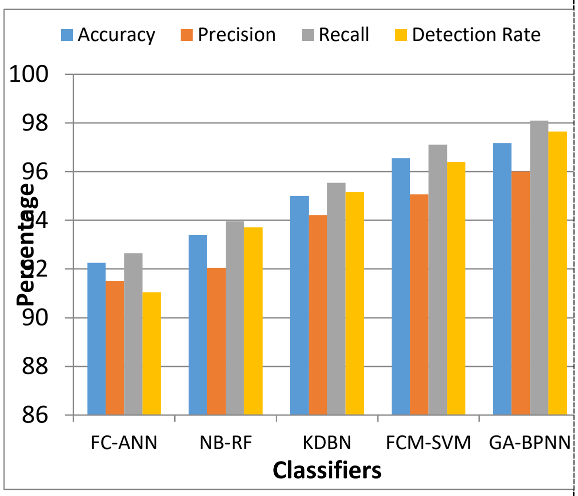

Figure.7. Graphical Plot of Performance Metrics of DoS/DDoS Detection

The performance analysis of the infiltration class and PortScan class are analyzed individually. Both labels are not similar to the features and action of the various classes. Infiltration attack has 36 instances, with a 0.001% ratio, which is the minimum attack ratio of the entire instances. For this infiltration attack detection, the achieved accuracy is 1.5% to 3.8% higher, and the detection rate is 0.7% and 6.8% higher than the other compared methods as represented below in table.6.

Table.6. Performance Analysis of Detecting Infiltration Attack

Technique	Accuracy	Recall	Precision	Detection Rate
FC-ANN	92.35	92.50	91.10	89.58
NB-RF	93.10	93.89	92.44	91.02
KDBN	93.88	94.06	92.65	93.27
FCM-SVM	94.67	94.91	93.16	95.64
GA-BPNN	96.21	96.73	94.98	96.40

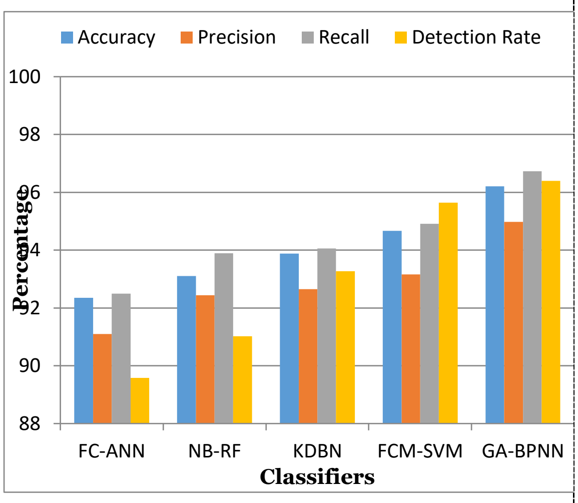

Figure.8. Graphical Plot of Performance Metrics of Infiltration Detection

The PortScan class has 158930 instances, with a 5.61% attack ratio comparing with the entire instances. The PortScan attack detection by GA-BPNN has achieved 1.2% to 7.5% higher accuracy and 1.6% to 6.6% more detection rate than compared techniques, as shown in table.7.

Table.7. Performance Analysis of Detecting PortScan Attack

Technique	Accuracy	Recall	Precision	Detection Rate
FC-ANN	90.12	90.21	89.33	89.32
NB-RF	92.40	93.06	91.10	90.86
KDBN	94.01	93.69	93.45	92.50
FCM-SVM	96.37	95.88	95.48	94.41
GA-BPNN	97.64	97.65	96.57	96.01

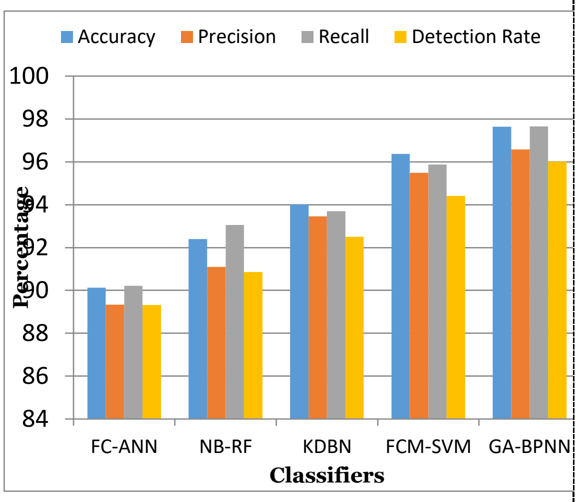

Figure.9. Graphical Plot of Performance Metrics of PortScan Detection

The Web Attack class includes Web Attack-Brute Force, SQL Injection, XSS with 2180 instances and a 0.07% attack ratio. The accuracy achieved by the proposed model for Web attack detection is 1% to 6% higher, and the detection rate is 1.7% to 7.3% higher than the other techniques as represented in table.8.

Table.8. Performance Analysis of Detecting Web Attack ("Web Attack – Brute Force, Web Attack – SQL Injection & Web Attack – XSS")

Technique	Accuracy	Recall	Precision	Detection Rate
FC-ANN	91.85	92.04	90.12	90.51
NB-RF	93.40	93.67	92.54	93.17
KDBN	95.24	96.11	94.10	94.98
FCM-SVM	96.88	96.55	96.04	96.12
GA-BPNN	97.90	97.98	96.76	97.85

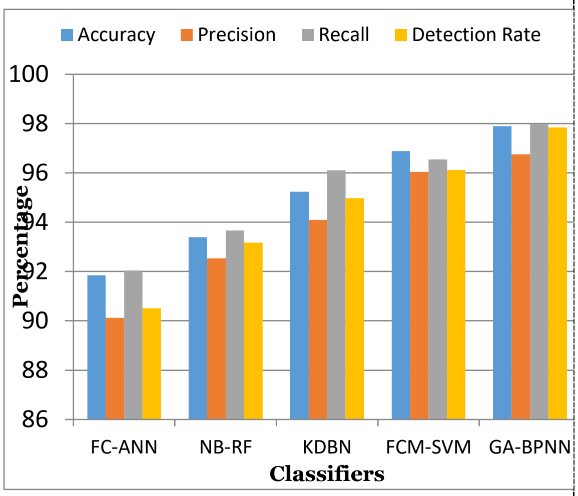

Figure.10. Graphical Plot of Performance Metrics of Web Attack Detection

Overall, the proposed GA-BPNN technique achieved higher accuracy in the detection of Web attacks with 97.90% accuracy and highest detection rate for Brute force attack detection with 97.89%.

V. CONCLUSION AND FUTURE WORK

In this research, an intrusion detection system for the cloud computing environment is proposed. Here in this model, the genetic algorithm and back propagation neural network are used for attack detection and classification. The CIC-IDS 2017 dataset was used for the evaluation of performance analysis. Initially, from the dataset, the data are preprocessed, and by using the genetic algorithm, the attack was detected. The detected attacks are classified using the back propagation neural network classifier for identifying the types of attacks. The performance analysis was executed, and the results are obtained and compared with the existing machine learning-based classifiers like FC-ANN, NB-RF, KDBN, and FCM-SVM techniques. The proposed GA-BPNN model outperforms all these classifying techniques in every performance metric, like accuracy, precision, recall, and detection rate. Overall, from the performance analysis, the best classification accuracy is achieved for Web attack detection with 97.90%, and the best detection rate is achieved for Brute force attack detection with 97.89%. In the future, the proposed research can be experimented as an IDS model for the Internet of Things (IoT) platform to detect and classify attacks.

ETHICS APPROVAL AND CONSENT TO PARTICIPATE

Not applicable.

HUMAN AND ANIMAL RIGHTS

No animals/humans were used for studies that are basis of this research.

CONSENT FOR PUBLICATION

Not applicable.

AVAILABILITY OF DATA AND MATERIALS

The authors confirm that the data supporting the findings of this research are available within the article.

FUNDING

None.

CONFLICT OF INTEREST

The authors declare no conflict of interest, financial or otherwise.

ACKNOWLEDGEMENTS

None.

REFERENCES

[1] Rajkumar Buyya, James Broberg, and Andrzej Goscinski. (2011). Cloud Computing Principles and Paradigms. John Wiley & Sons Inc.

[2] Chirag Modi et al. (2013). A survey on security issues and solutions at different layers of Cloud computing. Journal of Supercomputing. Springer. Vol.63, pp.561-592.

[3] Loubna Dali et al. (2015). A Survey of Intrusion Detection System. 2nd World Symposium on Web Applications and Networking (WSWAN). IEEE. pp.1-6.

[4] Manisha Rani and Gagandeep. (2019). A Review of Intrusion Detection System in Cloud Computing. International Conference on Sustainable Computing in Science, Technology & Management. pp.770-776.

[5] Ahmed Patel, Mona Taghavi, Kaveh Bakhtiyari, and Joaquim Celestino Junior. (2013). An intrusion detection and prevention system in cloud computing: A systematic review. Journal of Network and Computer Applications. Elsevier. Vol.36, pp.25-41.

[6] Ahmed Shawish and Maria Salama. (2014). Cloud Computing: Paradigms and Technologies. Inter-cooperative Collective Intelligence: Techniques and Applications, Studies in Computational Intelligence. Springer. Vol.495, pp.39-67.

[7] Mostapha Derfouf and Mohsine Eleuldj. (2019). Implementations of Intrusion Detection Architectures in Cloud Computing. International Conference of Cloud Computing Technologies and Applications. Springer. pp.100-124.

[8] Yasir Mehmood, Umme Habiba, Muhammad Awais Shibli, and Rahat Masood. (2013). Intrusion Detection System in Cloud Computing: Challenges and Opportunities. 2nd National Conference on Information Assurance. IEEE. pp.59-66.

[9] S N Dhage et al. (2011). Intrusion Detection System in Cloud Computing Environment. International Conference and Workshop on Emerging Trends in Technology. pp.235-239.

[10] Nureni Ayofe Azeez et al. (2019). Intrusion Detection and Prevention Systems: An Updated Review. Data Management, Analytics and Innovation. Vol.1042, pp.685-696.

[11] Nurudeen Mahmud Ibrahim and Anazida Zainal. (2020). A Distributed Intrusion Detection Scheme for Cloud Computing. International Journal of Distributed Systems and Technologies. Vol. 11, No.10, pp.68-82.

[12] Partha Ghosh, Sumit Biswas, Shivam Shakti, and Santanu Phadikar. (2020). An Improved Intrusion Detection System to Preserve Security in Cloud Environment. International Journal of Information Security and Privacy. Vol.14, No.1, pp.67-80.

[13] Ahmad Shokouh Saljoughi, Mehrdad Mehvarz, and Hamid Mirvaziri. (2017). Attacks and Intrusion Detection in Cloud Computing Using Neural Networks and Particle Swarm Optimization Algorithms. Emerging Science Journal. Vol. 1, No. 4, pp.179-191.

[14] Bahram Hajimirzaei and Nima Jafari Navimipour. (2019). Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm. ICT Express. Vol.5, No.1, pp.56-59.

[15] Mohamed Idhammad, Karim Afdel, and Mustapha Belouch. (2018). Distributed Intrusion Detection System for Cloud Environments based on Data Mining techniques. Procedia Computer Science. Elsevier. Vol.127, pp.35-41.

[16] Aws Naser Jaber and Shafiq Ul Rehman. (2020). FCM–SVM based intrusion detection system for cloud computing environment. Cluster Computing Springer. https://doi.org/10.1007/s10586-020-03082-6.

[17] Hongsheng Yin et al. (2019). Intrusion Detection Classification Model on an Improved k-Dependence Bayesian Network, Special Section on Innovation and Application of Intelligent Processing ff Data, Information and Knowledge as Resources in Edge Computing. IEEE Access. Vol.7, pp.157555-157563.

[18] Muhammad Salman Taj, Syed Irfan Ullah, Dr.Abdus Salam, and Wajid Ullah Khan. (2020). Enhancing Anomaly Based Intrusion Detection Techniques for Virtualization in Cloud Computing Using Machine Learning. International Journal of Computer Science and Information Security. Vol.18, No.5, pp.68-78.

[19] Roshani Gaidhane, C. Vaidya, and M. Raghuwanshi. (2014). Intrusion Detection and Attack Classification using Back-propagation Neural Network, International Journal of Engineering Research & Technology. Vol.3, No.3, pp.1112-1115.

[20] Razan Abdulhammed et al. (2019). Features Dimensionality Reduction Approaches for Machine Learning Based Network Intrusion Detection. Electronics. Vol.8, No.332, 2019, pp.1-27.