Journal of Information Security, 2019, 10, 1-24
http://www.scirp.org/journal/jis
ISSN Online: 2153-1242
ISSN Print: 2153-1234

Implementation of Network Intrusion Detection System Using Soft Computing Algorithms (Self Organizing Feature Map and Genetic Algorithm)

Joël T. Hounsou, Thierry Nsabimana, Jules Degila

Institut de Mathématiques et de Sciences Physiques, Porto-Novo, Benin, Université d'Abomey Calavi
Email: joelhoun@gmail.com, thierry.nsabimana@imsp-uac.org, jules.degila@gmail.com

How to cite this paper: Hounsou, J.T., Nsabimana, T. and Degila, J. (2019) Implementation of Network Intrusion Detection System Using Soft Computing Algorithms (Self Organizing Feature Map and Genetic Algorithm). *Journal of Information Security*, **10**, 1-24.
https://doi.org/10.4236/jis.2019.101001

Received: October 14, 2018
Accepted: December 1, 2018
Published: December 4, 2018

Abstract

In today's world, computer network is evolving very rapidly. Most public or/and private companies set up their own local networks system for the purpose of promoting communication and data sharing within the companies. Unfortunately, their data and local networks system are under risks. With the advanced computer networks, the unauthorized users attempt to access their local networks system so as to compromise the integrity, confidentiality and availability of resources. Multiple methods and approaches have to be applied to protect their data and local networks system against malicious attacks. The main aim of our paper is to provide an intrusion detection system based on soft computing algorithms such as Self Organizing Feature Map Artificial Neural Network and Genetic Algorithm to network intrusion detection system. KDD Cup 99 and 1998 DARPA dataset were employed for training and testing the intrusion detection rules. However, GA's traditional Fitness Function was improved in order to evaluate the efficiency and effectiveness of the algorithm in classifying network attacks from KDD Cup 99 and 1998 DARPA dataset. SOFM ANN and GA training parameters were discussed and implemented for performance evaluation. The experimental results demonstrated that SOFM ANN achieved better performance than GA, where in SOFM ANN high attack detection rate is 99.98%, 99.89%, 100%, 100%, 100% and low false positive rate is 0.01%, 0.1%, 0%, 0%, 0% for DoS, R2L, Probe, U2R attacks, and Normal traffic respectively.

Keywords

SOFM, Intrusion Detection Systems, False Positive Rate, Detection Rate, KDD Cup 99, GA

1. General Introduction

In computer security field, an attacker or a hacker can be understood as an unauthorized user attempts to penetrate the security defenses and gain access to the network system in order to violate the integrity, confidentiality and availability of resources. The hackers constantly invent new attacks and disseminate them over the internet [1]. Securing network is a delicate step to protect a company from the most common risks emanating from the Internet as well as from its own local network system. To prevent attacks or to reduce their severity, many solutions exist but no one can be considered satisfactory and complete [2].

An Intrusion Detection System (IDS) is currently a powerful tool used in many companies, institutions, universities and so for to protect their computer systems or/and computer networks from dangerous risks. It is divided into two parts such as Host based IDS and Network based IDS. Network based IDS monitoring all the traffic packets incoming and outgoing through the Internet, whereas Host based IDS is deployed locally on each host computer and monitoring only the host on which it is installed [3]. And then, there are two general categories of IDSs [4]: misuse detection and anomaly based. Misuse detection systems are most widely used and they detect intruders with known patterns. The signatures and patterns used to identify attacks consist of various fields of a network packet, like source address, destination address, source and destination ports or even some key words of the payload of a packet. These systems exhibit a drawback in the sense that only the attacks that already exist in the attack database can be detected, so this model needs continuous updating, but they have a virtue of having very low false positive rate. Anomaly detection systems identify deviations from normal behaviour and alert to potential unknown or novel attacks without having any prior knowledge of them. They exhibit higher rate of false alarms, but they have the ability of detecting unknown attacks and perform their task of looking for deviations much faster.

Self Organizing Feature Map Artificial Neural Network (SOFMANN) is unsupervised learning algorithm. For that reason, no human assistance needed during the training process. It is used in varied domains like in computer security as detector module. It can detect both known and novel attacks. Each sample data point from the input data space is shown in parallel to all the neurons in the SOM, and the winner is chosen to be the neuron that responds best [5]. It is a delicate algorithm used for visualization and presentation of complex data. It converts the input nodes from high dimensional to the low dimensional output nodes (1-D, 2-D or 3-D).

Genetic Algorithm (GA) operates on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to the solution of the problem that GA is trying to solve and at each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness value in the problem domain and breeding them together using the operators borrowed from the genetic process performed in nature, *i.e.* crossover and mutation [6].

GA is chosen because of some of its nice properties, e.g., robust to noise, no gradient information is required to find a global optimal or sub-optimal solution, self-learning capabilities, etc. Using GAs for network intrusion detection has proven to be a cost-effective approach [7] [8] [9] [10]. Moreover, Self Organizing Feature Maps (SOFM) is chosen among the soft computing algorithms because it is proven technique for automated clustering and visual organization and anomaly detection in IDS.

Our main aim is to provide an intrusion detection system based on soft computing algorithms such as Self Organizing Feature Map Artificial Neural Network and Genetic Algorithm to network intrusion detection system so as to classify four categories of network attacks including normal traffic namely DOS, Probe, R2L, U2L and Normal. KDD Cup 99 and 1998 DARPA dataset were applied for training and testing the intrusion detection rules. The traditional Fitness Function of Genetic Algorithm was also improved for performance evaluation. The experimental results shown good detection rate and law false positive rate for different attack types trained and tested with SOFM, whereas GA has maintained very high detection rate for different attack types but with high false positive rate which leads to very poor accuracy of anomaly detection system.

The main contribution of this work is to demonstrate the performance of the SOFM artificial neural network compared with the GA in the detection rate comparison of different attack types, including normal traffic.

The remainder of the paper is articulated into seven sections: Section 2 discusses problem statement. Section 3 presents a brief overview of Self Organizing Feature Map Artificial Neural Network (SOFMANN) and Genetic Algorithm (GA). Section 4 describes related works. Section 5 demonstrates our proposed Network Intrusion Detection System Framework. Section 6 presents the experimental results and Section 7 presents the conclusion and future work.

2. Problem Statement

Most public or/and private companies, Banks, Institutions and Universities defeat against network attacks by applying varied traditional or advanced security methods namely firewalls, Antivirus and intrusion detection systems. The IT Managers of the companies, Banks, Institutions and Universities suffer from the lack of efficient security methods to secure multiple departments of their Companies, Banks and Universities before computer network attack. It is a challenging security issues for IT Managers of the companies to protect their data and network systems from attacks with rapidly growing volume of network traffic and the number of attacks. Multiple commercials and open source Network Intrusion Detection Systems are currently used in the companies, banks and so far to protect their information and local network systems from attacks. Unfortunately, the Intrusions Detection Systems (IDSs) suffer from high false positive rate generated by Anomaly detection method which leads to very poor accuracy of anomaly detection system and high false negative rate generated by Misuse or Signature detection method due to the inability of detecting unknown attacks.

3. Brief Overview of Self Organizing Feature Map Artificial Neural Network and Genetic Algorithm

3.1. Brief Overview of Self Organizing Feature Map Artificial Neural Network

Self Organizing Feature Map is a type of neural network. Neural network is classified into two types of learning namely unsupervised and supervised learning. Unsupervised learning comprises two categories of Algorithms namely Self Organizing Feature Map (SOFM) and Adaptive Resonance Theory (ART), whereas Supervised learning comprises Radial Basis Function (RBF) and Multi Layer Perceptron Algorithms. The SOFM is an unsupervised learning algorithm because no human assistance needed during the learning process. The main role of the SOFM is to transform high dimensional of the input vector randomly chosen to low dimensional (1-D or 2-D). The SOFM algorithm is first developed by Teuvo Kohonen, Finnish Professor in 1982 known as Kohonen SOFM. It is designed for visualization and representation of huge amount of data, where human brain can easly undastand. And then, the structure of the SOFM is a single feed forward network [11], where each input vector is directory connected to each neuron in the Map. But the neurons in the Kohonen Map are not linked to each other. The number of the input dimensions is usually higher than the output dimensions. It is used in various applications namely image processing, voice recognition, speech recognition, spatial data mapping, data compression, pattern recognition, text mining and so on [12].

The Self Organizing Feature Map is a competitive learning algorithm, where each neuron in the Map competes so that the neuron which responds better is selected as the Best Matching Unit (BMU) or winning neuron. The neurons which are around the winner, is called neighborhood. The weights of the neurons close to the winning neuron are also adjusted but the magnitude of the change depends on the physical distance from the winning neuron and it is also decreased with the time [13]. Six stage of the learning process are listed and briefly described below:

- Stage 1: Each neuron's weights in the network of x dimension is randomly initialized;
- Stage 2: The randomly selected input vectors from huge amount of training datasets are fred to SOFM network;
- Stage 3: The Euclidean Distance between the input vectors and neuron's weights are computed to determine the winning neuron or Best Matching Unit (BMU).

To determine the Best Matching Unit (BMU), an iterative process is performed by each node in the network and the Euclidean Distance between the weight vector of each node and the current input vector is calculated. The node with the weight vector closest to the input vector is labeled as BMU [14]. To do that, we have two points such as $I\left(I_1, I_2, \cdots, I_n\right)$ known as input vector and $W\left(W_1, W_2, \cdots, W_n\right)$ known as weight vector of each neuron. The Euclidean Distance is computed as follows:

$$D = \left\| I_n - W_n \right\| = \sqrt{\left(I_1 - W_1\right)^2 + \left(I_2 - W_2\right)^2 + \cdots + \left(I_n - W_n\right)^2} \tag{1}$$

- Stage 4: The radius of the neighborhood closer to the winning neuron or BMU has the chance to be computed but the farthest one from the Best Matching Unit has no chance to be calculated.

The size of the neighborhood decreases progressively with n Epoch. All of the neurons around this radius are labeled neighbors of the winning neuron. To determine the neighborhood in the SOFM network which are closer the BMU, radius must be computed as follows:

$$r\left(t\right) = r_0 \exp\left(-t/\lambda\right) \tag{2}$$

where $r(t)$ means the width of the lattice in time t, r_0 means the width of the lattice in time t_0, t means a time or an iteration of the epoch and λ means a constant time t (current time).

- Stage 5: Both the Best Matching Unit neuron and its neighborhood weights are adjusted in order to respond better. The closest neurons within the neighborhood are more adjusted than the furthest neurons;

$$W\left(t+1\right) = W\left(t\right) + \Theta\left(t\right) K\left(t\right)\left(I\left(t\right) - W\left(t\right)\right) \tag{3}$$

where t means the time and K means the learning rate at time t, which decreases progressively in time. And then, the weight at time $t + 1$ is set to neighborhood nodes from the current instant weight $W(t)$, plus a fraction $K(t)$, the difference between the current weight of "$W(t)$" neuron and the input vector called $I(t)$, adjusted (theta) based on distance from the BMU [15].

In Equation (3) the learning rate at time t can be calculated using an exponential decay function.

$$K\left(t\right) = K_0 \exp\left(-t/\lambda\right) \tag{4}$$

In Equation (3), represents the amount of influence that the distance from one node to the BMU has on their learning, in the current time instant is calculated using Gaussian curve [15].

$$\Theta\left(t\right) = \exp\left(-D^2/2r^2\left(t\right)\right) \tag{5}$$

where D is a distance from one neuron to the winning neuron and r is the radius of the neighborhood. The function also decreases progressively in time.

- Stage 6: Go to Stage 2 if the number of epochs is not reached.

3.2. Brief Overview of Genetic Algorithm

Genetic Algorithm is an optimization technique using an evolutionary process [16]. Network connection in dataset is represented as chromosome. Fitness Function of Genetic Algorithm is a powerful metric used to evaluate each individual population in the training dataset so as to determine the good solution and to discard the bad one. An evolutionary process of Genetic Algorithm starts where initial population is randomly generated. Each population is now evaluated to find out the best results based on the predefined criteria. For that reason, the fitness value of each population or chromosomes are determined using the Fitness Function. The best fitness values are kept and the worst are prunned from the top list. The best individual populations are selected and undergo crossover and mutation to reproduce new children. This new children knwon as new population is applied for the next generation. The process continues until the number of generations or epochs has been terminated. Figure 1 showed the Genetic Algorithm process.

As displayed in Figure 1, Genetic Algorithm (GA) consists of three operators namely selection, crossover or recombination and mutation.

3.3. Data Structure

In Genetic Algorithm data structure is represented as chromosome. The rules to match the attacks are randomly generated and are encoded as an integer array with seven attributes, as displayed in Table 1. The first six attributes of the chromosome match the condition A of an attack. The seventh attribute identifies the attack type that the first six attributes identify when they match. The same data representation is also used by Ms.Lata Jadhav [16].

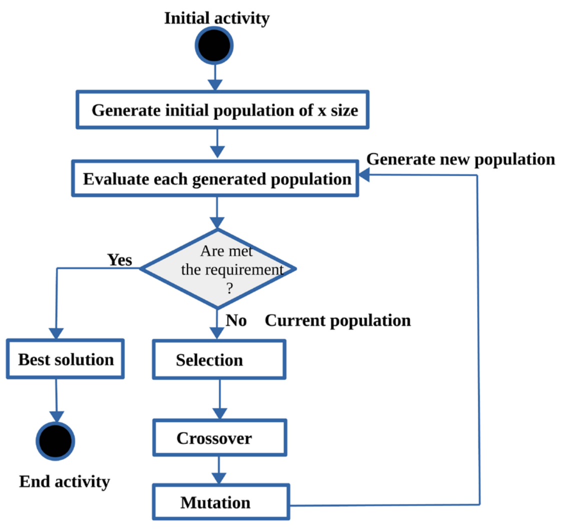


Figure 1. Genetic algorithm diagram.

Table 1. Selected attributes.

Feature Name	Format	Number of Genes
Duration	h:m:s	3
Protocol	Int	1
Source_port	Int	1
Destination_port	Int	1
Source_IP	a.b.c.d	4
Destination_IP	a.b.c.d	4
Attack_name	Int	1

Before evaluating the individual population from training dataset, the rules to match the attack types must be constructed correctly in order to classify successfull abnormal behaviours and normal traffic. And thus, Too many wildcards in the constructed rules are forbidden because they sometime block the connections and lead to very poor accuracy. The gene representation follows the simple rule if A then B, where if the first six attributes are logically and-ed together are true (A), then the rule matches the attack (B) [16]. The rule used to classify network connection in training data as well as U2R attack rlogin is explained below:

if (Duration = "0:0:14" and Protocol = "rlogin" and Source_port = 1022 and Destination_ port = 513 and Source_ip = "192.168.1.30" and Destination_ip = "192.168.0.20") then Attack_name = "rlogin".

The above rule shows that if a network packet is originated from source IP address 192.168.1.30 and source port 1022, and sent to destination IP address 192.168.0.20 and destination port 513 using the protocol rlogin, and the connection duration is 0:0:14 second, then it is a network attack of type rlogin.

3.4. The Existing and Improved Traditional Fitness Function

Traditional Fitness Function was defined by Man Leung Wong and Kwong Sak Leung (2000) [17]. It is based on the support-confidence framework and It is a performance measure used to determine the fitness value of individuals in the population. The best individuals in the population are selected according to the predefined fitness value threshhold. The fitness value is evaluated by determining how many attack connections the rule matches [18]. The traditional Fitness Function is built with two weights parameters such as w1 and w2 with default values 0.2, 0.8 respectively. Those weights are used to control the balance between the confidence and support during the training. This traditional Fitness Function generates in one hand the rules with low fitness value and the other hand with good fitness value.

To increase the fitness value of the generated rules, we have proposed two paramaters such as l1 and l2 known as lifts. The lifts l1 and l2, whose values are in the range of [−1.0.0.1.0], are used to increase the fitness value of the rules used in classifying the attack types, and used also to control the balance between the support and confidence. The generated rule with less fitness value has no chance to identify attack types. On the contrary, the rules with high fitness value have the chance in identifying attack types successful. **Figure 2** shows the existing traditional Fitness Function, whereas **Figure 3** shows the improved traditional Fitness Function. An experimental assessment of this improved traditional Fitness Function is presented in Section 6.1.

1) The existing traditional Fitness Function:

N: Total number of network connection rules in the training dataset;

|A|: Number of rules match the condition A;

|A and B|: Number of network connections match the rule if A then B;

w1, w2: weights employed to balance or control support and confidence.

(1) if A then B

(2) Support= |A and B|/N

(3) Confidence= |A and B|/A

(4) Fitness= w1*Support+w2*Confidence

Figure 2. Existing traditional fitness function perfomed on support-confidence and weights.

1) if X then Y
2) Support(X,Y)= |X and Y|/T
3) Confidence(X,Y)= |X and Y|/X
4) Fitness= |(l1-Support(X,Y))*w1+(l2-Confidence(X,Y))*w2|

Figure 3. The improved traditional fitness function perfomed on support-confidence, weights and lifts.

2) The Improved traditional Fitness Function

T: Total number of connection rules in the training dataset;

|X|: Number of connections matching the condition X;

|X and Y|: Connection matching rule if X then Y;

w1, w2: weights used to balance or control two terms: support and confidence;

l1, l2: lifts used to increase the fitness value of rules and to balance the support-confidence. lift values is in the range of [−1.0.0.1.0].

Table 2 displays a sample of a population individual rule t matching against a training dataset. We need to compute the fitness value using the two formulas carried out on Figure 2 and Figure 3. And thus, the training data and population individual rule t are displayed on Table 2. The condition part of the first 14 network features of the rule 4 and 11 in the training data match a population individual t. Moreover, the rule of 4 and 11 match the outcome part of 15 network features, where It indicates the classification of attack type.

The obtained results trained with the existing traditional Fitness Function is displayed on Figure 4 and the obtained results trained with the improved traditional Fitness Function is shown on Figure 5.

However, after computation of fitness value using both formulas, the fitness value of 90.6% found using the improved traditional Fitness Function is greater than the fitness value of 83.2% obtained using the existing traditional Fitness Function. It was concluded that the best population individual's fitness value obtained from the improved traditional Fitness Function has chance to identify attack types successfully. For that reason, this improved trational Fitness Function will be applied in our experiments to generate best rules used in identifying malicious attacks from the KDD Cup 99 and 1998 DARPA dataset.

3.5. Crossover and Mutation Operators

1) Individual Crossover Operator

(1) w1= 0.2

(2) w2= 0.8

(3) N= 12

(4) | A |= 2

(5) | A and B|= 2

(6) Support= |A and B|/N = 2/12= 0.16

(7) Confidence= |A and B|/A=2/2= 1

(8) Fitness= w1*Support+w2*Confidence

(9) Fitness= 0.2*0.16+0.8*1= 0.832= 83.2 %

Figure 4. Computation of fitness value for population individual rule t trained with the existing traditional Fitness Function.

(1) w1= 0.2

(2) w2= 0.8

(3) l1,l2= [-1 . 0.0.1.0]

(4) l1= -0.05

(5) l2= -0.06

(6) T= 12

(7) | X |= 2

(8) | X and Y|= 2

(9) Support(X,Y)= |X and Y|/T = 2/12= 0.16

(10) Confidence= |X and Y|/X=2/2= 1

(11) Fitness= |(l1-Support(X,Y))*w1+(l2-Confidence(X,Y))*w2|

(12) Fitness= |(-0.05-0.16)*0.2+(-0.06-1)*0.8|= 0.89 = 89 %|

Figure 5. Computation of fitness value for population individual rule t trained with the improved traditional Fitness Function.

Table 2. Sample of a population individual rule t matching against a training dataset.

ID	DURATION			PROTOCOL	SRC PORT	DST PORT	SRC IP				DST IP				ATTACK NAME
	H	M	S				0	1	2	3	0	1	2	3	
1	0	1	126	telnet	1754	23	192	168	1	30	192	168	0	20	-
2	0	0	0	htp	1784	80	192	168	1	30	192	168	0	40	phf
3	0	0	0	finger	240	79	192	168	1	30	192	168	0	40	port-scan
4	0	0	14	rlogin	1022	513	192	168	1	30	192	168	0	20	rlogin
5	0	0	2	rsh	1023	1021	192	168	1	30	192	168	0	20	rcp
6	0	0	38	telnet	1042	23	192	168	1	30	192	168	0	20	-
7	0	0	1	smtp	43494	25	192	168	0	40	192	168	1	30	-
8	0	0	1	smtp	43493	25	192	168	0	40	192	168	1	30	-
9	0	1	1	telnet	1769	23	192	168	1	30	192	168	0	20	-
10	0	0	22	telnet	1906	23	192	168	1	30	192	168	0	20	guess
11	0	0	14	rlogin	1022	513	192	168	1	30	192	168	0	20	rlogin
12	0	0	4	telnet	1914	23	192	168	1	30	192	168	0	20	telnet
							Population Individual rule								
t	**0**	**0**	**14**	**-1**	**-1**	**-1**	**192**	**168**	**1**	**30**	**192**	**168**	**0**	**20**	**rlogin**

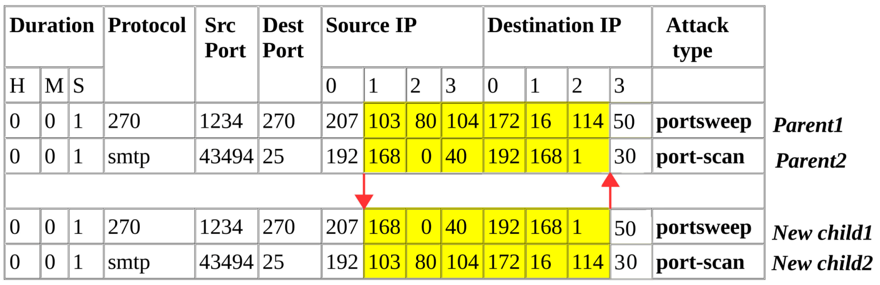
In the crossover operation, the best population individuals undergo crossover operator, where population individuals randomly selected involved in swapping genes of two parents with one an another to form two new children from each pair of parents. Crossover probability for a pair of parents were used in our experiments for performance evaluation of our implemented system. In Genetic Algorithm, there are three categories of crossover namely one point crossover, two point crossover and uniform crossover. Two point crossover technique will be used in our experiments. Figure 6 shows a sample of crossover operator.

2) Individual Mutation Operator

Mutation operator participates in modifying one or more genes value of single population individual due to the repeated use of crossover operator. For our experiment, mutation probability was employed. Figure 7 shows a sample of mutation operator carried out on single population individual.

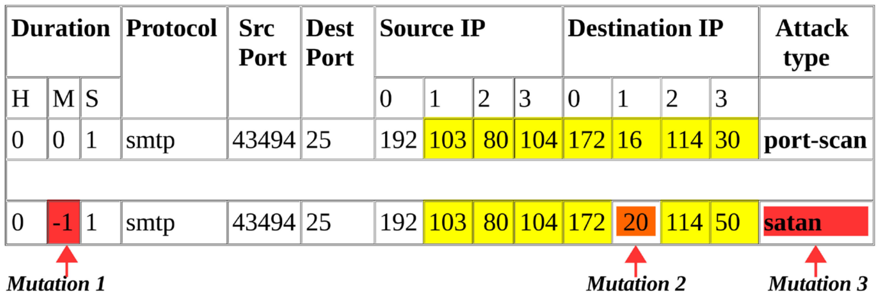
4. Related Work

An Intrusion Detection System is a powerful tool used to secure company's data from risks emanating from the public Internet as well as from their own local network system. Too many papers based on an IDS were published and some contributions were given. And thus, multiple soft computing algorithms were also used to support an Intrusion Detection System for reducing False Negative rate and False Positive rate generated by Anomaly detection method and Signature detection method. Most of them are Genetic Algorithms, Self Organizing Feature Maps Neural Network, Fuzzy Logic and so for. When used for intrusion detection, soft computing is a general term for describing a set of optimization and processing techniques that are tolerant of imprecision and uncertainty [19].



Duration			**Protocol**	**Src Port**	**Dest Port**	**Source IP**				**Destination IP**				**Attack type**	
H	M	S				0	1	2	3	0	1	2	3		
0	0	1	270	1234	270	207	103	80	104	172	16	114	50	**portsweep**	***Parent1***
0	0	1	smtp	43494	25	192	168	0	40	192	168	1	30	**port-scan**	***Parent2***
0	0	1	270	1234	270	207	168	0	40	192	168	1	50	**portsweep**	***New child1***
0	0	1	smtp	43494	25	192	103	80	104	172	16	114	30	**port-scan**	***New child2***

Figure 6. Crossover operator carried out on two parents to generate two new children.



Duration			**Protocol**	**Src Port**	**Dest Port**	**Source IP**				**Destination IP**				**Attack type**
H	M	S				0	1	2	3	0	1	2	3	
0	0	1	smtp	43494	25	192	103	80	104	172	16	114	30	**port-scan**
0	-1	1	smtp	43494	25	192	103	80	104	172	20	114	50	**satan**

Mutation 1
Mutation 2
Mutation 3

Figure 7. Mutation operator carried out on single population individual.

In 2012, Xueying Jiang and Kean Liu [20] described some of the issues of SOM neural network algorithm such as: false alarm rate, false negative rate, training time and give an improved anomaly detection SOM algorithm. They also give the intelligent detection model and the model of the training module, designed the main realization of FPSOM neural network algorithm. The simulation experiments were carried out in KDDCUP data sets. In 2009, E. J. Palomo, E. Domnguez, R. M. Luque, and J. Muoz [21] proposed a neural network model based on self organization for detecting intrusions. Growing Hierarchical SOM (GHSOM) was proposed to addresse the limitations of the SOM related to the static architecture of this model. KDD Cup 1999 benchmark was used for training the proposed GHSOM. In 2015, EMIRO DE LA HOZ FRANCO and ANDRES ORTIZ GARCIA [14] provided an implementation of an Intrusion Detection System based on Self Organizing Map and identified a methodology to validate the effectiveness of an Intrusion Detection Systems proposed in three phases (selection, training and classification) using FDR to feature selection and Self Organizing Maps to training-classification. In 2014, Kruti Choksi and Prof. Bhavin Shah [5] presented a survey which focused on IDS using Self Organizing Map and their survey shows that the existing IDS based on SOM have poor detection rate for U2R and R2L attacks. To overcome these issues, the improvement should be conducted. The normalization technique should be used to improve it. During the survey they also found that HSOM and GHSOM are advance model of SOM which have their own unique feature for better performance of IDS. In 2009, M. Bahrololum, E. Salahi and M. Khaleghi [22] proposed a new approach to design the system using a hybrid of misuse and anomaly detection for training of normal and attack packets respectively and the combination of unsupervised and supervised Neural Network (NN) for Intrusion Detection System were used. By the unsupervised NN based on Self Organizing Map (SOM), attacks will be classified into smaller categories considering their similar features. In 2006, Liberios VOKOROKOS and Anton BAL [23] present intrusion detections systems and design architecture of intrusion detection based on neural network self organizing map and described base problematic of neural network and intrusion detection system. Their work deals with specific design of intrusion detection architecture based on user anomaly behavior.

In the work done by B. Abdullah [6], he implemented an Intrusion Detction System by using Genetic Algorithm. Genetic parameters were discussed and implemented. He also applied a linear structure rule to classify normal connections and abnormal connections successfully. KDD99 Benchmark was evaluated with the implemented system. The obtained results showed high detection rate 99.87% and low False Positive rate 0.003%. His results were compared with available machine learning technique. Nitin Gupta [24] for him, he carried out an implementation of a network Intrusion Detction System using Genetic Algorithm to detect Denial of Service attacks. The implemented IDS was used to detect network intrusions. Fitness and Probabilistic approach were used to compute the optimality of the proposed new attack pattern. NSL-KDD dataset was used for training and testing purpose. Thus, Ms. Lata Jadhav [16], he implemented an IDS using Genetic Algorithm approach to derive the set of classification rules from audit data and traditional fitness function with support-confidence framework was used to measure the quality of each rule. The generated rules were used to detect or identify network intrusions. Experimental results show high detection rates based on Benchmark DARPA dataset.

5. Proposed Network Intrusion Detection System Framework

Our proposed Network Intrusion Detection System Framework based on soft computing algorithms (Self Organizing Feature Map Artificial Neural Network and Genetic Algorithm) is split into four phases namely: Datasets mining Phase, Preprocessing Phase, Training and Testing Phase. As displayed in Figure 8.

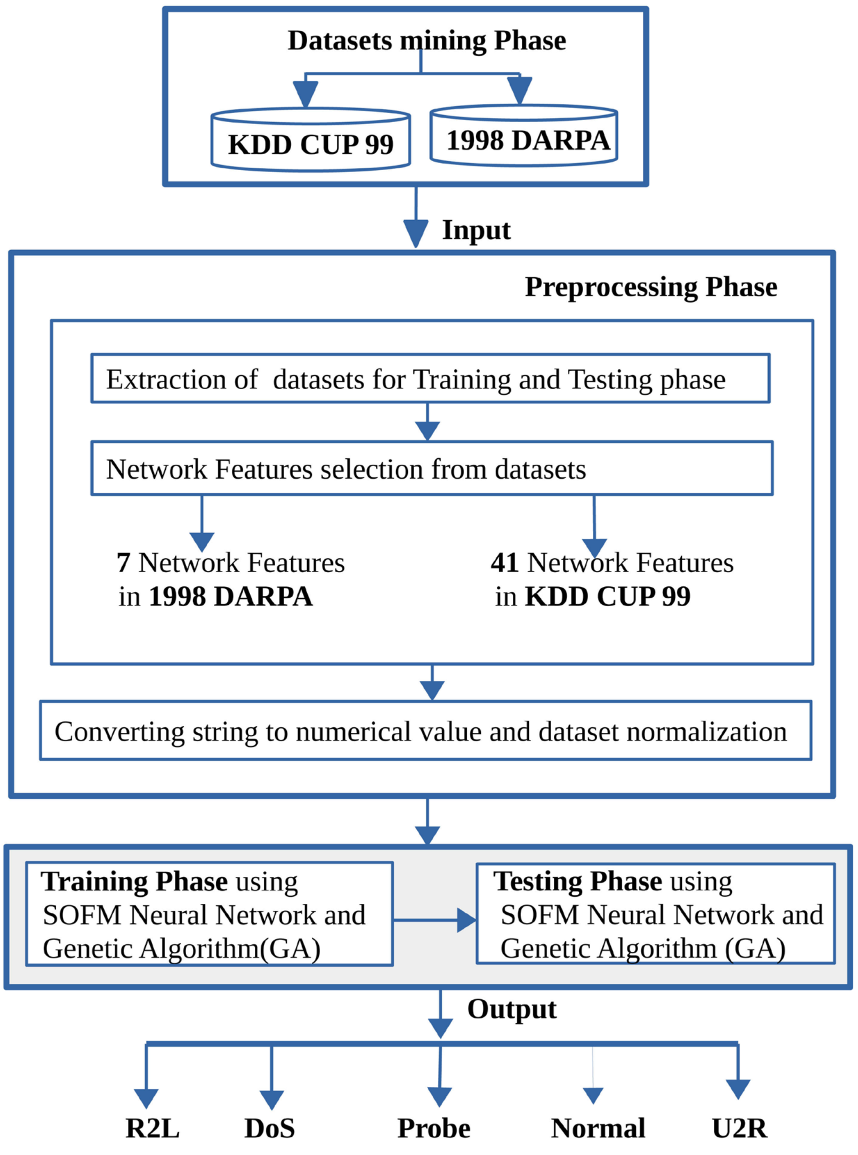

Datasets mining Phase
KDD CUP 99
1998 DARPA
Input
Preprocessing Phase
Extraction of datasets for Training and Testing phase
Network Features selection from datasets
7 Network Features in 1998 DARPA
41 Network Features in KDD CUP 99
Converting string to numerical value and dataset normalization
Training Phase using SOFM Neural Network and Genetic Algorithm(GA)
Testing Phase using SOFM Neural Network and Genetic Algorithm (GA)
Output
R2L
DoS
Probe
Normal
U2R

Figure 8. Proposed network intrusion detection system framework.

5.1. Datasets Mining Phase

For the first phase of our system, the datasets to be mined with our proposed NIDS are KDD CUP 99 and 1998 DARPA dataset. The KDD CUP 99 is one of the most datasets used to train and to test the intrusion detection rules. The KDD Cup 99 dataset is the subset of 1998 DARPA dataset that was collected by simulation of the operation of a typical USA Air Force Local Area Network (LAN) with multiple attacks classified into four categories: probe, denial of service, user to root and remote to local. KDD Cup 99 dataset records contain 41 features which fall into four categories: basic, traffic, content and host related ones [25]. And thus, 1998 DARPA Dataset (Defense Advanced Research Projects Agency) was invented in 1998 by MIT Lincoln Labs for the purpose of evaluating intrusion detection systems on offline environment. Moreover, many existing datasets such as KDD 99, DARPA, Kyoto 2006 and other public dataset, are uncontrollable, unmodifiable, and may contain old types of attack [26].

5.2. Preprocessing Phase

Genetic Algorithm (GA) and Self Organizing Feature Maps Neural Network (SOFM) cannot handle symbolic features. For that reason, two subsets training and testing dataset from KDD Cup 99 dataset were picked, the transformation of string to numerical value were carried out, the normalization of the transformed data was done and suitable network features were selected for better accuracy. The preprocessing phase of our proposed system consists of three parts, as can be seen in **Figure 8**:

- Extraction of datasets: Training and testing datasets were picked from KDD Cup 99 dataset. These two subsets data were used to measure the performance of GA and SOFM Neural Network in intrusion detection rules.
- Converting string to numerical values: The network features containing the symbolic form named protocol type, Service and Flag were converted to numerical values by affecting unique number for each network feature.
- Normalization of the numerical values: After conversion to numerical values, we then convert the numerical vector into the binary or normalized (in interval [0,1]) [27], this was done 1) to ensure that one feature did not overpower another (for those with large units) and 2) to speed up training and convergence time by using smaller numbers and the precise method is represented by this equation:

$$\text{Data\_Normalized} = \left(X - \text{mean}\left(X\right)\right) / \text{stddev}\left(X\right) \qquad (6)$$

where X represents each column in the dataset.

- Network features selection: 41 network features in the KDD Cup 99 dataset were selected and trained, tested with Genetic Algorithm and Self Organizing Feature Map Artificial Neural Network. And 7 network features in 1998 DARPA dataset was selected and trained with Genetic Algorithm for producing rules used in identifying network attacks.

5.3. Training and Testing Phase Using GA and SOFM

The proposed Network Intrusion Detection System based on soft computing algorithms SOFM Artificial Neural Network and GA comprises two modules namely Training and testing phase. In the training phase, a set of classification rules are generated from KDD Cup 99 and 1998 DARPA using the SOFM and GA in an offline environment, whereas, in the testing phase, the generated rules are used to classify incoming network connections in the real time environment and once the rules are generated, the intrusion detection is simple and efficient [6].

5.4. The Proposed Architecture of SOFM Artificial Neural Network

The proposed architecture of Self Organizing Feature Map ANN is displayed in **Figure 9**.

Our proposed architecture of Self Organizing Feature Map artificial neural network consists of two layers namely: an input node and an output node. The input nodes indicates 41 network features from KDD CUP 99 dataset, which are symbolized in short form as F1, F2, F3, ..., F40 and F41 means Feature 1, Feature 2, Feature 3 until Feature 41. In other way, we have 41 dimensions of the input nodes. The output nodes consists of five outputs N1, N2, N3, N4 and N5 known as nodes were to classify five class of attack types including normal traffic namely DoS, Probe, R2L, U2L and Normal as classification problem. Each node's map namely N1, N2, N3, N4 and N5 receives 41 coordinates (w11, w12, w13, ..., w140, w141) and each node's Map is connected to each input node but node's Map are not connected to each other in the Kohonen network. The Map dimension was initially set (25,25), epochs were initially set 100, learning rate was initially set 0.2 and neighborhood radius was initially set 16. Our proposed SOFM Neural Network was trained for each class of attack types as classification problem in order to evaluate the performance of the algorithm on individual attacks.

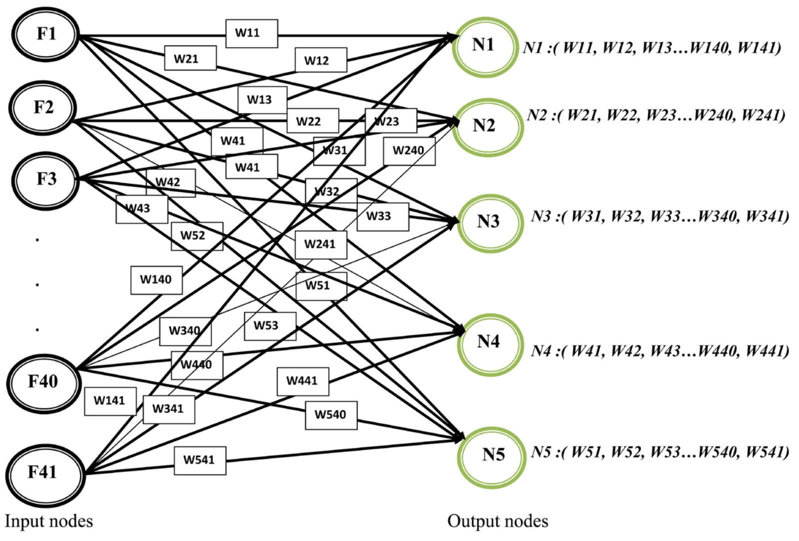

F1
F2
F3
F40
F41
N1
N2
N3
N4
N5
N1 :(W11, W12, W13…W140, W141)
N2 :(W21, W22, W23…W240, W241)
N3 :(W31, W32, W33…W340, W341)
N4 :(W41, W42, W43…W440, W441)
N5 :(W51, W52, W53…W540, W541)
W11
W21
W12
W13
W22
W23
W41
W31
W240
W41
W42
W32
W43
W33
W52
W241
W140
W51
W340
W53
W440
W441
W141
W341
W540
W541
Input nodes
Output nodes

Figure 9. The proposed architecture of SOFM artificial neural network.

6. Experimental Results

In this section, we present the obtained results using Genetic Algorithm (GA) and Self Organizing Feature Map Artificial Neural Network trained and tested on KDD Cup 99 and 1998 DARPA. The purpose of our experiments is to show the detection rate, False Positive rate and Classification time comparison for different attack types. The experiments were performed on EliteBook 6930p laptop with $Intel^{®}\ core^{TM}$ 2 Duo CPU P8600@ 2.40 GHz × 2 and 4 GB of RAM. The operating system was type 64-bit using Debian 9 stretch and hard drive 350 GB. Our system was implemented using Python 2.7.13.

6.1. Performance Measures

To evaluate our system, the two standard metrics of detection rate and false positive rate developed for network intrusions, have been used [6]: Detection rate (DR) is computed as the ratio between the number of correctly detected intrusions and the total number of intrusions, that is

$$\text{DR} = \frac{\text{True Positive}}{\text{False Negative} + \text{True Positive}} \tag{7}$$

False positive (FP) (also said false alarm) rate is computed as the ratio between the numbers of normal connections that are incorrectly classifies as intrusions and the total number of normal.

$$\text{FP} = \frac{\text{False Positive}}{\text{True Negative} + \text{False Positive}} \tag{8}$$

6.2. Self Organizing Feature Map and Genetic Algorithm Training Parameters

Table 3 displayed the used Genetic Algorithm training parameters during the traing and testing process of our system. And thus, Table 4 is shown the SOFM training parameters.

6.3. Simulation Outcomes Obtained Using GA and SOFM

Our first experiment compares our own solutions generated by our implemented GAIDS which applies the improved traditional Fitness Function as

Table 3. GA parameters for performance evaluation.

GA parameters for performance evaluation				
w1	**w2**	**Generation number**	**Population size**	**Crossover_Technique**
0.2	0.8	5000	400	two points crossover

Table 4. Self organizing feature map neural network parameters.

Simulation parameters for SOFM			
Cluster Unit	**Epochs**	**Learning rate**	**Neighborhood_Radius**
625	100	0.2	16

shown on Figure 3 with NetGA [28] which uses traditional fitness function [17] [29] [30], and both are trained with 1998 DARPA training sample dataset bsm. list [31] using the Genetic Algorithm parameters as displayed in Table 3.

Table 5 shows the rules generated by NetGA and Table 6, Table 7 show the rules produced by our Implemented GAIDS. The simulation outcomes produced by NetGA shows that this system was able to print to the standard

Table 5. Outcomes from NetGA [28].

	H	M	S	Protocol	Src Port	Dest	Source IP Address				Destination				Attack	Fitness
1	0	0	5	Telnet	−1	−1	−1	168	1	30	192	168	0	20	rcp	0
2	0	1	42	telnet	1832	513	−1	168	0	30	−1	168	0	20	rcp	0
3	0	1	19	rsh	1022	23	192	168	1	30	192	168	0	20	rlogin	0
4	0	1	20	smtp	−1	23	192	168	1	30	192	168	0	20	rcp	0
5	0	1	14	rsh	43587	23	192	168	0	30	−1	168	0	20	rcp	0
6	0	0	−1	rlogin	−1	23	192	168	1	40	192	168	0	20	guess	0
7	0	0	20	rsh	1906	−1	192	168	0	30	192	168	0	20	rsh	0
8	0	0	−1	rlogin	1906	513	192	168	1	30	192	168	0	20	rcp	0
9	0	0	20	−1	−1	513	192	168	1	30	192	168	0	20	guess	0
10	0	0	14	telnet	1832	23	192	168	0	30	192	168	0	20	rlogin	0
11	0	0	11	exec	43497	−1	192	168	1	30	−1	168	0	20	rcp	0
12	−1	1	23	rlogin	−1	−1	192	168	1	40	−1	168	0	20	rcp	0
13	0	1	48	telnet	1876	−1	192	168	1	30	−1	168	0	20	port-scan	0
14	0	0	−1	rsh	−1	−1	192	168	1	30	−1	168	0	20	rcp	0.2698
15	0	0	−1	rsh	1023	−1	192	168	1	30	−1	168	0	20	rcp	0.8031
16	−1	0	2	rsh	1023	−1	192	168	1	30	−1	168	0	20	rcp	0.8031
17	0	0	14	rlogin	−1	513	192	168	1	30	−1	168	0	20	rsh	0.8031
18	0	0	14	rlogin	−1	513	192	168	1	30	−1	168	0	20	rsh	0.8031
19	0	0	−1	−1	−1	512	192	168	1	30	192	168	0	20	port-scan	0.8031
20	0	0	−1	rsh	1023	−1	192	168	1	30	−1	168	0	20	rcp	0.8031
21	−1	0	2	rsh	1023	−1	192	168	1	30	−1	168	0	20	rcp	0.8031
22	−1	0	2	rsh	1023	−1	192	168	1	30	−1	168	0	20	rcp	0.8031
23	−1	0	2	rsh	1023	−1	192	168	1	30	−1	168	0	20	rcp	0.8031
24	0	0	23	telnet	−1	23	192	168	1	30	192	168	0	20	guess	0.8063
25	0	0	23	telnet	−1	23	192	168	1	30	192	168	0	20	guess	0.8063
26	0	0	5	−1	−1	−1	192	168	1	30	−1	168	0	20	port-scan	0.8063
27	0	0	5	−1	−1	−1	192	168	1	30	−1	168	0	20	port-scan	0.8063
28	0	0	23	telnet	−1	23	192	168	1	30	192	168	0	20	guess	0.8063
29	0	0	5	−1	−1	−1	192	168	1	30	−1	168	0	20	port-scan	0.8063
30	0	0	5	−1	−1	−1	192	168	1	30	−1	168	0	20	port-scan	0.8063

Table 6. Outcomes from our implemented GAIDS.

	H	M	S	Protocol	Src Port	Dest	Source IP Address				Destination				Attack	Fitness
0	0	0	2	rsh	1022	1021	192	168	1	30	192	168	0	20	rsh	0.8771
1	0	0	2	rsh	1022	1021	192	168	1	30	192	168	0	−1	rsh	0.8771
2	0	0	2	rsh	1022	1021	192	168	1	30	−1	168	0	20	rsh	0.8771
3	0	0	2	rsh	1022	1021	192	168	−1	30	192	168	0	20	rsh	0.8771
4	−1	0	2	rsh	1022	1021	192	168	−1	30	192	168	0	20	rsh	0.8771
5	0	0	2	rsh	1022	1021	192	168	1	30	192	168	−1	20	rsh	0.8771
6	0	0	−1	rsh	1022	1021	192	168	1	30	192	168	0	20	rsh	0.8771
7	0	0	2	rsh	1022	1021	192	−1	1	30	192	168	0	20	rsh	0.8771
8	0	0	−1	rsh	1022	1021	192	168	1	30	192	168	0	20	rsh	0.8771
9	0	0	2	rsh	1022	−1	192	168	1	30	192	168	0	20	rsh	0.8771
10	0	0	23	telnet	−1	23	192	168	1	30	192	168	0	20	guess	0.8802
11	0	0	23	telnet	−1	−1	192	168	1	30	192	168	0	20	guess	0.8802
12	0	0	23	telnet	−1	23	192	168	1	30	192	168	0	−1	guess	0.8802
13	0	−1	23	telnet	−1	23	192	168	1	30	192	168	0	20	guess	0.8802
14	0	0	23	telnet	−1	−1	192	168	1	30	192	168	0	20	guess	0.8802
15	0	0	23	telnet	−1	23	192	168	−1	30	192	168	0	20	guess	0.8802
16	0	−1	23	telnet	−1	23	192	168	1	30	192	168	0	20	guess	0.8802
17	0	0	23	telnet	−1	23	192	168	1	30	−1	168	0	20	guess	0.8802
18	0	0	23	telnet	−1	23	192	168	1	30	192	168	−1	20	guess	0.8802
19	0	0	23	telnet	−1	23	192	168	1	−1	192	168	0	20	guess	0.8802
20	0	0	5	−1	−1	−1	192	168	1	30	192	168	0	20	port-scan	0.8802
21	0	0	5	−1	−1	−1	−1	168	1	30	192	168	0	20	port-scan	0.8802
22	0	0	5	−1	−1	−1	192	−1	1	30	192	168	−1	20	port-scan	0.8802
23	0	0	5	−1	−1	−1	192	168	1	30	192	−1	0	20	port-scan	0.8802
24	0	0	5	−1	−1	−1	192	−1	1	30	192	168	0	20	port-scan	0.8802
25	0	0	5	−1	−1	−1	192	168	1	30	192	168	−1	20	port-scan	0.8802
26	0	0	5	−1	−1	−1	192	1	1	30	192	−1	0	20	port-scan	0.8802
27	0	0	5	−1	−1	−1	192	168	1	30	192	−1	0	20	port-scan	0.8802
28	0	0	5	−1	−1	−1	192	168	−1	30	192	168	0	20	port-scan	0.8802
29	−1	0	5	−1	−1	−1	192	168	1	30	192	168	0	20	port-scan	0.8802
30	0	0	2	rsh	1023	1021	192	168	1	30	−1	168	0	20	rcp	0.8771

Table 7. Outcomes from our implemented GAIDS continued.

	H	M	S	Protocol	Src Port	Dest	Source IP Address				Destination				Attack	Fitness
31	0	0	2	−1	1023	1021	192	168	1	30	−1	168	0	20	rcp	0.8771
32	0	0	2	rsh	1023	1021	192	168	1	30	−1	168	0	−1	rcp	0.8771
33	0	0	2	rsh	1023	−1	192	168	1	30	192	168	0	20	rcp	0.8771
34	0	0	2	rsh	1023	1021	192	168	1	−1	192	168	0	20	rcp	0.8771
35	0	0	2	rsh	1023	1021	−1	168	1	30	192	168	0	20	rcp	0.8771
36	0	0	2	rsh	1023	1021	192	168	1	30	192	168	0	−1	rcp	0.8771
37	−1	0	2	rsh	1023	1021	192	168	1	30	192	168	0	20	rcp	0.8771
38	0	0	2	rsh	1023	1021	192	−1	1	30	192	168	0	20	rcp	0.8771
39	0	−1	2	rsh	1023	1021	192	168	1	30	192	168	0	20	rcp	0.8771
40	0	0	14	rlogin	1023	513	192	−1	1	30	192	168	0	20	rlogin	0.8771
41	0	0	14	rlogin	1023	513	192	168	1	30	192	−1	0	20	rlogin	0.8771
42	0	0	14	rlogin	1023	513	192	168	1	30	192	168	−1	20	rlogin	0.8771
43	0	0	14	rlogin	−1	513	192	168	1	30	192	168	0	20	rlogin	0.8771
44	0	0	14	rlogin	1023	513	−1	168	1	30	192	168	0	20	rlogin	0.8771
45	0	0	14	rlogin	1023	513	192	−1	1	30	192	168	0	20	rlogin	0.8771
46	0	0	14	rlogin	1023	513	192	168	1	30	192	168	0	−1	rlogin	0.8771

output the top 30 rules. From the first rule to thirteenth rules, the fitness value is equal to 0%. These rules have no effectiveness in identifying network attacks (DoS, R2L, Probes and U2L). Thus, from fourteenth rules to thirtieth rules, fitness values vary between 26.98% and 80.63%. Such rules can identify network attacks (DoS, R2L, Probes and U2L) successful. On the contrary, the rules produced by our implemented GAIDS, was able to print to the standard output 46 top rules. From the index zero to the index 46, we have very high fitness values which vary between 87.71% and 88.24%. These rules generated by GAIDS have effectiveness in identifying network attacks than rules produced by NetGA. Hence, we can conclude that the rules produced by our GAIDS using the improved traditional Fitness Function have maintained high fitness value than NetGA. For that reason, it has a powerful in classifying network attacks (DoS, R2L, Probes and U2L).

For the last experiment simulates the test module to use the generated Network Intrusion Detection System rules against the KDD Cup 99 test to measure the quality of the rules generated from our implemented system's training module. After our implemented system has been trained with the training KDD Cup 99 dataset [32], the generated rules are then integrated to the test module. Table 8 showed the class distribution of experimental on KDD Cup 99 dataset and the class distribution of our experimental is summarized by the graph of Figure 10.

Table 8. Class distribution of experimental on KDD CUP 99 dataset.

	DoS Attacks		Probe Attacks		R2L Attacks		U2L Attacks		Normal			
	Attacks number	%	Attacks number	%	Attacks number	%	Attacks number	%	Normal number	%	SUM	%
Training Dataset	20,000	44.94	15,000	33.7	5200	11.68	4300	9.66	34	0.07	44,534	100
Testing Dataset	10,000	46.08	7000	32.25	3283	14.74	1500	6.91	498	2.23	22,281	100
Generalization Dataset	5000	0.4	3000	0.24	2000	0.16	2500	0.2	3	0.02	12,503	100

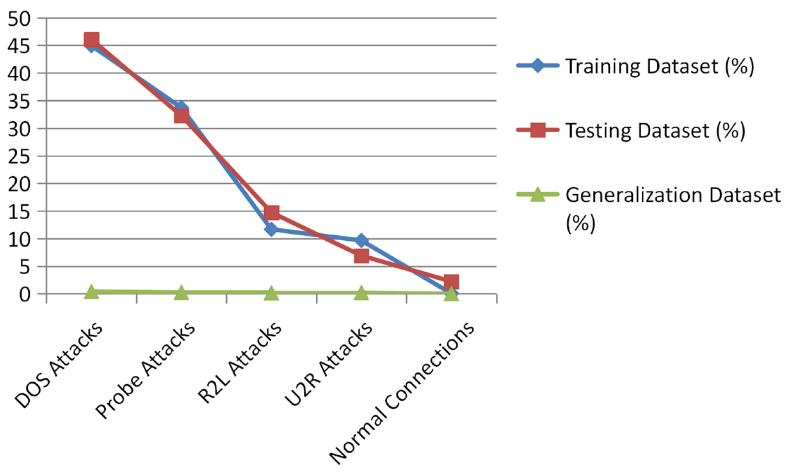


Figure 10. Graph of class distribution of experimental on KDD CUP 99 dataset.

The overall simulation outcomes and their related statistical calculations are displayed in Table 9, Table 10 and Table 11. It is concluded that the Network Intrusion Detection System rules generated by SOFM Artificial Neural Network when trained with KDD Cup 99 can detect for different attack types successful with high attack detection rate and law false positive rate.

Tables 12-14 show that the rules generated by Genetic Algorithm when trained with KDD Cup 99 can successful detect for different attack types in the training dataset namely DoS, Probe, R2L, U2R including normal traffic with very high attack detection rate but with very high false positive rate which may lead to very poor accuracy of anomaly detection system.

Table 15 compares our implemented algorithms (Genetic Algorithm and Self Organizing Feature Map Artificial Neural Network). In particular, we demonstrated the Detection rate, False Positive rate and classification rate. In training, testing and generalization detection rate, the detection rate generated by GA is better than SOFM but the false positive rate generated by SOFM is very low, better than the false positive rate in GA which may lead to very poor accuracy of Anomaly detection System. It is very difficult to compare the classification time of each attack types because the training speed depends on various factors such as implementation of the algorithms, condition of the hardware and software

Table 9. Training detection rate comparison for different attack types using SOFM on KDD Cup 99.

Attack types and Normal	Detection rate (%)	False Positive rate (%)	Classification Time (s)
DoS	99.98	0.01	7274.78
Probe	100	0	8995.55
R2L	99.89	0.10	14,490.77
U2R	100	0	10,569.26
Normal	100	0	10,916.12

Table 10. Testing detection rate comparison for different attack types using SOFM on KDD Cup 99.

Attack types and Normal connections	Detection rate (%)	False Positive rate (%)	Classification Time (s)
DOS	99.93	0.06	3654.03
Probe	99.86	0.13	3886.96
R2L	99.91	0.08	4304.89
U2R	99.84	0.15	4509.60
Normal	99.91	0.08	3876.71

Table 11. Generalization detection rate comparison for different attack types using SOFM on KDD Cup 99.

Attack types and Normal connections	Detection rate (%)	False Positive rate (%)	Classification Time (s)
DOS	99.96	0.03	2133.17
Probe	100	0	2326.36
R2L	100	0	2652.01
U2R	100	0	1901.46
Normal	100	0	2452.61

Table 12. Training detection rate comparison for different attack types using GA on KDD Cup 99.

Attack types and Normal connections	Detection rate (%)	False Positive rate (%)	Classification Time (s)
DOS	100	15.1	19,723.99
Probe	100	8.8	2716
R2L	100	11.9	1835.32
U2R	100	0	1181.53
Normal	100	3.05	3612.49

Table 13. Testing detection rate comparison for different attack types using GA on KDD Cup 99.

Attack types and Normal connections	Detection rate (%)	False Positive rate (%)	Classification Time (s)
DOS	100	13.5	1178.15
Probe	100	0.5	175.52
R2L	100	1.14	615.71
U2R	100	9.51	985.83
Normal	100	17.58	2336.81

Table 14. Generalization detection rate comparison for different attack types using GA on KDD Cup 99.

Attack types and Normal connections	Detection rate (%)	False Positive rate (%)	Classification Time (s)
DOS	100	10.05	1352.11
Probe	100	7.23	1026.07
R2L	100	15.05	458.03
U2R	100	8.10	435.84
Normal	100	3.04	485.71

Table 15. Detection rate (%) comparisons between GA and SOFM.

Classifier algorithms	Attack types and Normal	Training detection rate			Testing detection rate			Generalization detection rate		
		DR (%)	FPR (%)	Classification Time (s)	DR (%)	FPR (%)	Classification Time (s)	DR (%)	FPR (%)	Classification Time (s)
SOFM	DOS	99.98	0.01	7274.78	99.93	0.06	3654.03	99.96	0.03	2133.17
	Probe	100	0	8995.55	99.86	0.13	3886.96	100	0	2326.36
	R2L	99.89	0.10	14,490.77	99.91	0.08	4304.89	100	0	2652.01
	U2L	100	0	10,569.26	99.84	0.15	4509.60	100	0	1901.46
	Normal	100	0	10,916.12	99.91	0.08	3876.71	100	0	2452.61
GA	DOS	100	15.1	19,723.99	100	13.5	1178.15	100	10.05	1352.11
	Probe	100	8.8	2716	100	0.5	175.52	100	7.23	1026.07
	R2L	100	11.9	1835.32	100	1.14	615.71	100	15.05	458.84
	U2L	100	0	1181.53	100	9.51	985.83	100	8.10	435.84
	Normal	100	3.05	3612.49	100	17.58	2336.81	100	3.04	485.71

platform and size of the training dataset. In general the obtained results in SOFM have maintained both high detection rate and low false positive rate than GA. But GA has maintained very high detection rate but it suffer from very high false positive rate, for that reason, It leads to very poor accuracy of Anormaly detection system.

7. Conclusion and Future Work

In the experimental results obtained in the training detection rate comparison for different attack types, in the testing detection rate comparison for different attack types and in the generalization detection rate comparison for different attack types using GA and SOFM as classifier algorithms, and KDD Cup 99, 1998 DARPA as a dataset, the Self Organizing Feature Map Artificial Neural Network has maintained very high detection rate with low false positive rate than GA. The obtained results from GA are shown the best detection rate but with the highest false positive rate which can lead to very poor accuracy of Anomaly detection system. However, the highest attack detection rate and the lowest false positive rate obtained using SOFM on KDD Cup 99 demonstrated the efficiency and effectiveness of the algorithm in classifying attack types from KDD Cup 99 dataset. For that reason, the SOFM Artificial Neural Network can either classify malicious attacks or normal traffic successful. And thus, the rules produced by our GAIDS using the improved traditional Fitness Function have maintained high fitness value than NetGA. The training speed depends on various factors such as the implementation of the algorithms, conditions of the hardware and software platform and the size of the training dataset. For future work, this work will be extended for other classifier algorithms.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Ibrahim, L.M., Basheer, D.T. and Mahmod, M.S. (2013) A Comparison Study for Intrusion Database (Kdd99, Nsl-Kdd) Based on Self Organization Map (SOM) Artificial Neural Network. *Journal of Engineering Science and Technology*, **8**, 107-119.

[2] Sharma, G.D. and Kumar, V. (2016) Towards Configured Intrusion Detection Systems. Global Journals Inc., USA, Vol. 16, Version 1.0.

[3] Bunel, P. (2004) An Introduction to Intrusion Detection Systems. SANS Institute, GIAC Security Essentials, Certificate (GSEC), Practical Assignment, Version 1.4c, SANS Conference, London.

[4] Bankovi, Z., Stepanovi, D., Bojani, S. and Nieto-Taladriz, O. (2007) Improving Network Security Using Genetic Algorithm Approach. *Computers and Electrical Engineering*, **33**, 438-451. https://doi.org/10.1016/j.compeleceng.2007.05.010

[5] Ramadas, M., Ostermann, S. and Tjaden, B. (2003) Detecting Anomalous Network Traffic with Self-Organizing Maps. In: Vigna, G., Kruegel, C. and Jonsson, E., Eds., *Recent Advances in Intrusion Detection*. RAID 2003. Lecture Notes in Computer Science, Vol. 2820, Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-45248-5_3

[6] Abdullah, B., Abd-Alghafar, I., Salama, G.I. and Abd-Alhafez, A. (2009) Performance Evaluation of a Genetic Algorithm Based Approach to Network Intrusion Detection System. 13*th International Conference on Aerospace Sciences and Aviation Technology*, *ASAT*-13, Military Technical College, Cairo, Egypt, 26-28 May 2009, 1-17.

[7] Bridges, S.M. and Vaughn, R.B. (2000) Fuzzy Data Mining And Genetic Algorithms Applied to Intrusion Detection. *Proceedings of* 12*th Annual Canadian Information Technology Security Symposium*, Ottawa, Canada, 19-23 June 2000, 109-122.

[8] Chittur, A. (2001) Model Generation for an Intrusion Detection System Using Genetic Algorithms. High School Honors Thesis, Ossining High School. In cooperation with Columbia Univ.

[9] Crosbie, M. and Spafford, G. (1995) Applying Genetic Programming to Intrusion Detection. In *Working Notes for the AAAI Symposium on Genetic Programming*, MIT Press, Cambridge, MA, 1-8.

[10] Li, W. (2004) A Genetic Algorithm Approach to Network Intrusion Detection. SANS Institute, USA, Vol. 15, 209-216.

[11] Oksuz, A. (2007) Unsupervised Intrusion Detection System. Master's Thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark.

[12] Choksi, K., Shah, B. and Kale, O. (2014) Intrusion Detection System Using Self Organizing Map: A Survey. *International Journal of Engineering Research and Applications*, **4**, 11-16.

[13] Patole, V.A., Pachghare, V.K. and Kulkarni, P. (2010) Self Organizing Maps to Build Intrusion Detection Systems. *Journal of Computer Applications*, **1**.

[14] De La Hoz Franco, E., Ortiz Garcia, A., Ortega Lopera, J., De La Hoz Correa, E. and Mendoza Palechor, F. (2015) Implementation of an Intrusion Detection System Based on Self Organizing Map. *Journal of Theoretical and Applied Information Technology*, **71**.

[15] Sayad, S. (2010) Self-Organizing Maps (SOM). Bahen Centre Information Technology, Toronto, Ontario, Canada.

[16] Jadhav, M.L. and Gaikwad, C.M. (2014) Implementation of Intrusion Detection System Using GA. *International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering* (*IJIREEICE*), **2**, 1733-1736.

[17] Wong, M.L. and Leung, K.S. (2000) Data Mining Using Grammar Based Genetic Programming and Applications. Kluwer Academic Publishers, Dordrecht.

[18] Paliwal, S. and Gupta, R. (2012) Denial-of-Service, Probing and Remote to User (R2L) Attack Detection Using Genetic Algorithm. *International Journal of Computer Applications*, **60**, 57-62.

[19] Sunita, S., Chandrakanta, B.J. and Chinmayee, R. (2016) A Hybrid Approach of Intrusion Detection Using ANN and FCM. *European Journal of Advances in Engineering and Technology*, **3**, 6-14.

[20] Jiang, X., Liu, K., Yan, J. and Chen, W. (2012) Application of Improved SOM Neural Network in Anomaly Detection. *Physics Procedia*, **33**, 1093-1099. https://doi.org/10.1016/j.phpro.2012.05.179

[21] Palomo, E.J., Domnguez, E., Luque, R.M. and Muoz, J. (2009) An Intrusion Detection System Based on Hierarchical Self-Organization. In: *Proceedings of the International Workshop on Computational Intelligence in Security for Information Systems*, Springer, Berlin, Heidelberg, 139-146. https://doi.org/10.1007/978-3-540-88181-0_18

[22] Bahrololum, M., Salahi, E. and Khaleghi, M. (2009) Anomaly Intrusion Detection Design Using Hybrid of Unsupervised and Supervised Neural Network. *International Journal of Computer Networks and Communications*, **1**, 26-33.

[23] Vokorokos, L., Balaz, A. and Chovanec, M. (2006) Intrusion Detection System Using Self Organizing Map. *Acta Electrotechnica et Informatica*, **6**, 1-6.

[24] Gupta, N., Pareek, N. and Pandey, K. (2011) Genetic Algorithm Based Network Intrusion Detection System. *International Journal of Advanced Research in Computer Science*, **2**, 480-483.

[25] Proti, D.D. (2018) Review of KDD Cup'99, NSL-KDD and Kyoto 2006+ Datasets. *Vojnotehniki Glasnik*, **66**, 580-596. https://doi.org/10.5937/vojtehg66-16670

[26] Alzahrani, S. and Hong, L. (2018) Generation of DDoS Attack Dataset for Effective IDS Development and Evaluation. *Journal of Information Security*, **9**, 225-241. https://doi.org/10.4236/jis.2018.94016

[27] Amini, M., Jalili, R. and Shahriari, H.R. (2006) RT-UNNID: A Practical Solution to Real-Time Network-Based Intrusion Detection Using Unsupervised Neural Networks. *Computers and Security*, **25**, 459-468. https://doi.org/10.1016/j.cose.2006.05.003

[28] Lavender, B.E. (2010) Implementation of Genetic Algorithms into a Network Intrusion Detection System (netGA), and Integration into NProbe. Thesis Work, California State University, Sacramento.

[29] Gong, R.H., Zulkernine, M. and Abolmaesumi, P. (2005) A Software Implementation of a Genetic Algorithm Based Approach to Network Intrusion Detection. *Proceedings of the 6th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and 1st ACIS International Workshop on Self-Assembling Wireless Networks*, May 2005, 246–253.

[30] Lu, W. and Traore, I. (2004) Detecting New Forms of Network Intrusion Using Genetic Programming. *Computational Intelligence*, **20**, 475-494. https://doi.org/10.1111/j.0824-7935.2004.00247.x

[31] https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset

[32] The Third International Knowledge Discovery and Data Mining Tools Competition. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html