Efficient Feature Selection Technique for Network Intrusion Detection System Using Discrete Differential Evolution and Decision Tree

Ebenezer Popoola[1] and Aderemi Adewumi[2]

(Corresponding author: Aderemi Adewumi)

School of Mathematics, Statistics & Computer Science, University of KwaZulu-Natal[1]
Durban, 4000, South Africa
Email: Adewumia@ukzn.ac.za & popoolaebenezer@gmail.com

(Received June 30, 2016; revised and accepted Sept. 3 & Oct. 11, 2016)

Abstract

Network intrusion is a critical challenge in information and communication systems amongst other forms of fraud perpetrated over the Internet. Despite the various traditional techniques proposed to prevent this intrusion, the threat persists. These days, intrusion detection systems (IDS) are faced with detecting attacks in large streams of connections due to the sporadic increase in network traffics. Although machine learning (ML) has been introduced in IDS to deal with finding patterns in big data, the irrelevant features in the data tend to degrade both the speed and accuracy of detection of attacks. Also, it increases the computational resource needed during training and testing of IDS models. Therefore, in this paper, we seek to find the optimal feature set using discretized differential evolution (DDE) and C4.5 ML algorithm from NSL-KDD standard intrusion dataset. The result obtained shows a significant improvement in detection accuracy, a reduction in training and testing time using the reduced feature set. The method also buttresses the fact that differential evolution (DE) is not limited to optimization of continuous problems but work well for discrete optimization.

Keywords: C4.5; Differential Evolution; Machine Learning; Network Intrusion Detection; NSL-KDD

1 Introduction

Asides other challenges faced by computer network systems, network intrusion remains one of the unresolved issues. It has drawn the attention of researchers due to its continuous threat to the sustenance of businesses. Intrusion is described as any form of activity which tends to breach the availability, confidentiality or security of networks [13, 16]. The goal of IDS is to identify normal connections from both those using computer resources without authorization or abusing the privileges given to them. However, IDS are faced with a greater challenge as the network traffic grows [3, 21].

According to [4] over 25 billion devices will be connected by 2020 which will lead to more traffic as services are moved to the cloud due to ease of management and reduced running cost. Therefore, to sustain this technological advancement, there is a need for swift action in solving this network threat. IDS is categorized into two namely "Misuse" and "Anomaly" Detection [24]. While misuse detection deals with identifying known patterns of intrusion by comparing it with previous signatures in the database, anomaly detection identifies patterns that deviate from the standard pattern. This work focuses on anomaly detection because attackers tend to change their method of intrusion when discovered thereby making misuse detectors useless.

To detect anomaly in networks, several methods have been proposed among which ML based IDS have shown to be more efficient. Despite its advantage over other methods, it still suffers from high false positive rate (FPR), false negative rate (FNR), expensive computational cost and slow classification during training and testing. These downsides defeat the purpose of IDS since it is expected to be fast and accurate. Many works attribute this deficiency to irrelevant features in the dataset and have proposed various feature reduction techniques as the solution. Some nature inspired (NI) algorithms have shown to be effective in searching for an optimal set of features. These include Genetic Algorithm (GA) [26], Particle Swarm Optimization (PSO) [17], Ant Colony Optimization (ACO) [33], etc. but little has been done using Differential Evolution (DE). Hence, this work hybridizes DDE and C4.5 ML algorithm for finding an optimal set of feature.

2 Related Works

To address the issue of network intrusion, various steps has been taken. Some work focused on finding the best parameter settings for the classifier while others addressed removing irrelevant features from the dataset.

Due to unavailability of a standard labelled dataset, most research on IDS has used DARPA dataset known as KDDCUP'99 [19]. But investigation shows that the dataset has some fundamental issue which places a shadow of doubt on the usefulness of models designed with it [18]. Some of the problems are:

1) Redundant records in the dataset which causes the classifier to see less frequent records as noise. This leads to misclassification.

2) Duplication of records which made some models have better detection rates on the reoccurring data.

3) The above leads to having too large dataset which leads to excessive computational time. Therefore, most works randomly select a subset of the dataset which gives no basis to compare different models or performance of different feature set. Also, this leaves no basis for comparing the speed of IDS.

NSL-KDD [20] provided by [32] is a refined subset of the KDDCUP'99 dataset which solves the issues listed above. It was validated by testing it on some ML algorithms using their default parameters without a reduction in feature set. This serves as a benchmark to subsequent models. For binary classification ("Normal" or "Attack"), the following accuracies were obtained from the dataset.

This refined dataset has been validated using 7 ML algorithms on WEKA tool [14] without tuning the parameters of the learners. The resulting classification accuracy per classifier are as follows; NB-Tree: 82.67%, J48: 81.05%, Random Forest (RF): 81.59%, Multi-Layer Perceptron (MLP): 77.41%, Naive Bayes (NB): 76.56% and Support vector machine (SVM): 69.52%.

Using this dataset, a various number approach is being taken to either increase the classification accuracy or reduce the time needed during training and testing IDS. In some case, both objectives are achieved. One approach is to find optimal parameter setting of the classifier, and another is to tactically reduce the feature used to achieve faster training and testing time.

Garg and Kumar [13] reviewed various selection and classification techniques. The work tested the performance of combining two to three feature selection methods using Boolean AND operation. Out of 10 techniques tested, the combination of Symmetric and Gain Ratio for feature selection using 15 features and IBK classifier yielded the highest accuracy. However, no reason was given on why and how random data was selected from the dataset. Hence, the result can not be replicated.

Aziz et al. [5] also compared the performance of correlation-based feature selection (CFS), sequential floating selection (SFS), principal component analysis (PCA), information gain and rough sets in selecting appropriate features. SFS methods performed best using 26 features.

Al-Jarrah et al. [2] proposed two novel feature selection techniques: RandomForest-Forward Selection Ranking (RF-FSR) and RandomForest-Backward Elimination Ranking (RF-BER). Although the 15 features selected using RF-FSR achieved higher classification accuracy, it was only tested using cross-validation which does not guarantee its usability as reflected in [22].

Gaikwad and Thool [12] proposed Bagging ensemble classifier method on NSL-KDD dataset using the RepTree algorithm as the base class. The method utilized a reduced feature set of 29 features to achieve an accuracy of 81.2988% on the test set and 99.6761% using 10-fold cross-validation on the training set. Also, [27] compared the performance of selected ML algorithms where Bagging ensemble selection algorithm performed best with 97.85% accuracy after cross-validation. Ingre and Yadav [15] performed an analysis of NSL-KDD dataset using Artificial Neural Network (ANN) for both binary class classification (normal and attack) and five class classification (normal, DoS, U2R, R2L, and Probe). The method achieved 81.2% and 79.9% accuracies showing binary classification performs better than multiclass. Pervez and Farid [22] proposed an SVM-based feature selection method which achieved 91% accuracy using three features and 99% with 41 features on all the training set; the setback is that its classification accuracy when tested with an unseen dataset give 82.37%. Deshmukh et al. [10] approached intrusion detection by normalizing, discretized and selecting feature before training three classifiers (NB-Tree, NB, and AD Tree). The output showed NB algorithm performed efficiently regarding effectiveness, elegance, simplicity and robustness when compared to others on the training set. Since [10] did not provide this result when further evaluated on the test set, it is uncertain whether the model would yield high detection rate as results from previously proposed methods show that a training model may give a high accuracy, but does not guarantee the same when tested with new attacks.

Lately, nature-inspired (NI) algorithm was introduced to solving either feature selection or parameter optimization. Benaicha et al. [6], considered GA in detecting some attacks in NSL-KDD dataset using nine selected features, the focus was on detecting some DoS attacks (Neptune, smurf, teardrop, pod, back). The result was quite impressive but did not address other DoS attacks in test data and other attacks in NSL-KDD.

Likewise, [7] proposed an IDS which used GA, information gain (IG), mutual correlation and cardinality of features to achieve higher accuracy. The IG based feature selection was able to increase the accuracy to 87.54% using nine features. It was also limited to detecting Neptune, Satan, and Smurf attacks which are under the denial of service (DoS). In [5] network intrusion was approached by reviewing GA with different feature selection methods. Correlation-based feature selection techniques such

as IG, sequential floating, rough set and principal component analysis (PCA) for feature extraction were used on NSL-KDD dataset. Sequential-floating backward selection showed to be more effective with higher detection rates.

Stein et al. [28] proposed a model which considered decision tree classifier for network intrusion detection with GA-based feature selection. Their work was focused on identifying features which could separate each category of attacks (DoS, Probe, U2R, and R2L) from normal connections in a network. This led to creating four different models which were yet to be integrated as one.

Despite the superiority of DE in solving various optimization [9], there is a limited usage in its application to intrusion detection. Elsayed et al. [11] proposed DE for classification of attack but applied flexible neural tree (FNT) [8] for feature selection. Here, we propose a hybridization of DDE and C4.5 for selecting optimal features.

3 Problem Definition

Though some IDS adopts the working principle of ML for classification of connections in a network, it still faces a critical challenge which limits its use in some real life environment. Some systems proposed are either slow or raises false alarms, e.g., misclassification of normal connection as intrusive which could frustrate the experience of client or classification of intrusive connection as normal leading to a significant loss. Also, the fundamental issues about KDDCUP'99 dataset [18] give doubt on the accuracy presented by most works. For an IDS to be effective, the classification accuracy must be high, and detection rate must be fast without the usage of excessive computational resources. To achieve this goal, there is the need for removing irrelevant features from network connections during classification. Hence, the major problem is how to select the best set of feature for the right ML algorithm without loss of relevant information. Several feature reduction methods have been proposed including the use of other computational intelligence techniques such fuzzy systems, neural networks (NN) and NI algorithms. Among the various NI algorithms, GA show to be the favorite optimization technical several works has been done while little or no work has been done using DE for finding features for IDS.

Generally, GA concentrates more on exploring the search space than exploiting the environment of a solution. It is however noted that exploiting a weak solution using DE could yield a better solution than the best in the population for discrete optimization problems which in turn reduces the number of generations. Hence, we utilize DE for finding the optimal set of features where the selection criteria are based on the classification accuracy of the feature set. We also emphasize the need for using NSL-KDD dataset [32] during modelling which solves the fundamental issues of the KDDCUP'99 dataset.

4 Methodology

This section explains various concepts such as C4.5 Decision Trees (DT), DE and how they are harnessed in our proposed technique to search for the optimal feature set.

4.1 Decision Trees (C4.5)

DT designed by [23] are flow-chart-like structures which follow an IF-THEN rule have proved to be fast and efficient. It is an improvement of the ID3 algorithm which is a top-down recursive divide and conquer approach. It seeks to divide the entire dataset by first breaking the values of a feature(or Attribute) into ranges. To do this efficiently at each node, C4.5 calculates the gain ratio of each attribute and selects the one with highest the gain ratio. The expected information, Entropy, Information gain and gain ratio are calculated tho ensure the best split is achieved which can be seen as follows.

Let S be the sample data to be split which consist of $X_1, \cdots X_j$ attributes where the j^{th} represents the distinct class labels($C_i, \cdots C_m$)

$$H(S) = -\sum_{i=1}^{m} P_i log_2(P_i), \tag{1}$$

where, H is the expected information that is needed to classify a given sample data. P_i is the probability that an instance in sample S belongs to class C_i which is calculated as the sum of all instances in S which are of class C_i divided by the total instance in S ($|C_{i,s}|/|S|$).

Assuming the range of values of X_1 is subdivided into v distinct values $\{x_1, x_2, \cdots, x_v\}$ and used to divide S into $\{S_1, S_2, \cdots, S_v\}$ where S_j contains instances in D that corresponds to x_i of X_1. Hence the entropy which is the expected information needed to classify an instance from S based on splitting by X_1 in order to arrive at an exact classification can be given as:

$$info_{X_1}(S) = \sum_{j=1}^{v} \frac{|S_j|}{|S|} \times H(S_j). \tag{2}$$

It is noted that $\frac{|S_j|}{|S|}$ is the weight of the J^{th} partition and the smaller the value of $info_{X_1}(S)$, the greater the possibility that all instances in the partition belong to the same class.

Now, information gain is the expected reduction in entropy caused by partitioning the samples according to the attribute X_1. That is, the difference between splitting S based on the proportion of classes and partitioning on X_1 which is given as:

$$Gain(X_1) = H(S) - info_{X_1}(S). \tag{3}$$

Gain Ratio is used to reduce a bias towards multi-valued attributes by taking the number and size of branches into account when choosing an attribute.

$$GainRatio(X_1) = Gain(X_1)/SplitInfo(X_1). \tag{4}$$

Some other advantages of C4.5 can be extracted from [1]. Various DT classifiers including C4.5 adopts a greedy approach as it considers the immediate consequence of selecting a feature to split on at the current node without estimating the full depth future implication. Hence, a need to devise a method such that features with a high gain ratio at the initial split but more misclassification cost at the end is removed.

4.2 Differential Evolution

DE [30] which is also a population-based search technique as GA is the main optimization technique explored in this work. The sequence of operating is given as initialization, mutation, recombination/crossover and selection [25]. Although DE was originally designed for continuous problems, we see how to adapt its concept on the discrete problem through representing the solutions like that of GA. Also, using similar operators of GA but in the sequence of DE as shown in Figure 1. The details are as follows.

- **Initialization**: An initial population of chromosomes is generated for the first generation: $X_G = \{x_{1,G}, x_{2,G}, x_{3,G}, \cdots, x_{i,G}\}$ where each chromosome $x_{i,G}$ is called the ***target vector***.
- **Mutation**: The mutation is different from that of GA. Here, for each $x_{i,G}$, a ***donor vector*** $v_{i,G+1}$ is generated.

$$v_{i,G+1} = x_{r1,G} + F.(x_{r2,G} - x_{r3,G}), \tag{5}$$

  where $x_{r1,G}, x_{r2,G}, x_{r3,G}$ are randomly chosen from the population excluding the target vector. F is a user defined ***mutation or constant factor*** ($F \in [0, 2]$) which controls the amplification of the differential variation $(x_{r2,G} - x_{r3,G})$.
- **Crossover**: The mutant vector is mixed with the target vector to produce a ***trial vector*** $u_{i,G+1}$ as follows:

$$u_{i,G+1} = \begin{cases} v_{j,i,G+1} \ if \ rand_{j,i} \ \leq CR \ or \ j = I_{rand} \\ x_{j,i,G} \ if \ rand_{j,i} \ > CR \ and \ j \neq I_{rand} \end{cases}$$

  $i = 1, 2, \cdots, N; j = 1, 2, \cdots, D$, where N is the population size and D is the dimension of x_i. CR is the crossover constant ($[0, 1]$). I_{rand} ensures $v_{i,G+1} \neq x_{i,G}$.
- **Selection**: The trial vector $u_{i,G+1}$ and target vector $x_{i,G}$ are evaluated using the objective function. A comparison is then made between the two. The one with lower value is moved to the new generation.

$$x_{i,G+1} = \begin{cases} u_{i,G+1} \ if \ f(u_{i,G+1}) \ \leq f(x_{i,G}) \\ x_{i,G} \ otherwise \end{cases}$$

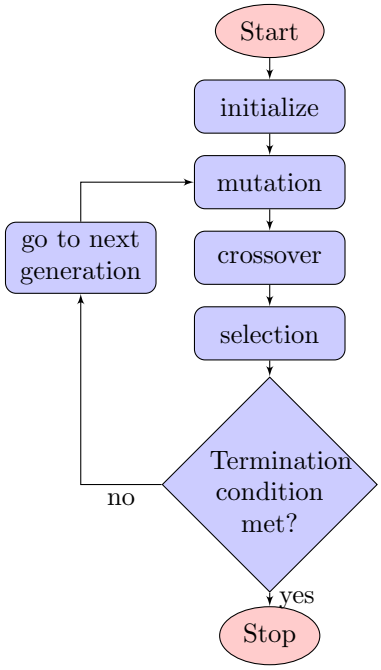

Figure 1: Differential evolution

4.3 Proposed Method

The proposed method models a DDE [31] strategy for finding the optimal set of features as follows.

4.3.1 Initialization of Population

Since the optimization problem deals with whether or not a feature is used, the solutions are expected are binary strings of 1 and 0 where a gene position having 0 means the feature is not used. However for the DE operators to have more effect, we initially encode the 41 positions as random numbers in the range of 0 and 1. Table 1 gives a picture of how a solution is represented.

Table 1: Chromosome

0.5	**0.1**	**0.9**	**0.7**	**0.55**	**0.41**	**0.1**	**0.3**

4.3.2 Mutation

To generate the mutant vector v_i for a target vector $x_{i,G}$, we do the following:

$$v_{i,G+1} = m \bigoplus F.(x_{r,G}) \tag{6}$$

where: $m = \{a, a, a, \cdots \mid |m| = |x_{r,G}|, 0 < a < 1\}$ in this case, $a = 0.3$ and $F = 0.4$ which is a scalar value.

$x_{r,G}$ is a random chromosome such that $x_{i,G} \neq x_{r,G}$. While F is the amplification factor which is preset, m gives some stability since the element of $x_{r,G}$ are random numbers in the range of 0 and 1.

4.3.3 Crossover

The mutant vector is mixed with the target vector to produce a ***trial vector*** $u_{i,G+1}$ as follows:

$$u_{i,G+1} = \begin{cases} v_{j,i,G+1} \; if \; rand_{j,i} \; \leq CR \; or \; j = I_{rand} \\ x_{j,i,G} \; if \; rand_{j,i} \; > CR \; and \; j \neq I_{rand} \end{cases}$$

$i = 1, 2, \cdots, N; j = 1, 2, \cdots, D$, where $N = 10$ is the population size and D is the dimension of x_i. CR is the crossover constant ($[0, 1]$) which is set at 0.5.

4.3.4 Selection

The following test is done in parallel with fitness evaluation in the next section to select the candidate for the next generation.

$$x_{i,G+1} = \begin{cases} x_{i,G} \; if \; u_{i,G+1} = \{0, 0, 0, \cdots\} \; or \; \{1, 1, 1, \cdots\} \\ u_{i,G+1} \; if \; f(u_{i,G+1}) \; \leq f(x_{i,G}) \\ x_{i,G} \; otherwise \end{cases}$$

If the first condition is met, it leads to an outright reject of the trial vector because it means none or all the features are selected but the goal is to find optimal features. The second selects the trial vector if the fitness is less than the target vector since we are minimizing the misclassification rate.

4.3.5 Fitness Evaluation

The fitness of each chromosome is evaluated by converting the genes to 0's and 1's based threshold values and mapping the gene position to the features in the dataset. Taking note that every gene whose value is 0 indicates the feature is not used for classification while the reverse is the case where the value is 1. This reduces the dataset column-wise before been passed to WEKA tool where classification is done using the C4.5 algorithm. The accuracy after training and testing becomes the fitness value which is calculated as follows:

$$accuracy = \frac{correctly \; classified \; instance}{total \; test \; set}$$

$$misclassification \; rate = 1 - accuracy.$$

4.3.6 Algorithm for Feature Selection

The algorithm 1 shows how features were selected.

Algorithm 1 Feature selection using discrete DE

Initialize Population
Initialize m, F, CR,Generation G and conversion threshold tr
for $i \leftarrow 1$ to G **do**
  for each chromosome **do**
    Set target vector $(x_{i,G})$ = chromosome
    Randomly select $x_{r,G} \| x_{r,G} \neq x_{i,G}$
    Mutate to get the trial mutant vector $v_{i,G+1}$
    crossover each allele of $v_{i,G+1}$ and $x_{i,G}$ with
    a Probability of CR to get $u_{i,G+1}$
    Selection
    if All element of $u_{i,G+1}$ 0's or 1's **then**
      Add target vector to new population
    else
      Convert alleles to 0 or 1 based tr
      Evaluate $f(u_{i,G+1})$ and $(x_{i,G})$
      if $f(u_{i,G+1})$ ¡ $(x_{i,G})$ **then**
        Add initial trial vector before
        conversion to new population
      else
        Add initial target vector before
        conversion to new population
      end if
    end if
  end for
end for
map the genes of the chromosome to its actual feature name

4.3.7 Performance Metric Used For The Proposed Method

There are standard performance metrics used in evaluating of an ML algorithm which can either be done through the command line or graphically using the WEKA tool. In this work, the following metrics are verified after obtaining the optimal feature set from the search technique:

- ***Sensitivity or True positive rate (TPR)***: This is the ratio of positive class (attack connections) that are correctly identified to the total positive class (P).

$$True \; positive \; rate = \frac{True \; positives \; (TP)}{P}.$$

- ***True negative rate (TNR) or specificity***: It estimates the ratio of negative class (normal connections) that are correctly identified to the total negative class (N).

$$True \; negative \; rate = \frac{True \; negative \; (TN)}{N}.$$

- ***Precision*** is the measure of exactness, i.e., the percentage of instances classified as an attack, out of the total number of cases classified as attacks.

$$Precision = \frac{TP}{TP \; + \; FP},$$

where: FP (***false positives***) are normal which are wrongly classified as attacks while the counterpart called ***false negatives*** (FN) are attack connections wrongly classified as normal.

- ***Recall*** shows the measure of completeness. It is calculated just like true positive rate.

- ***error or misclassification rate*** is calculated as 1 - *accuracy*. It can also be computed as

$$missclassification\ rate = \frac{FP + FN}{P + N}.$$

- ***Receiver Operating Characteristics (ROC) Curve*** is a plot that shows the trade-off between TPR and the rate of false positives (FPR). It is a visual representation of the rate at which the proposed model can recognize normal connections versus misclassification of attacks as normal for various sections of the dataset. Also, it leaves an **area under the curve (AUC)** which also determines how well the classifier performs. The closer the area is to 1.0, the better the classifier.

5 Experimental Setting

The implementation of the proposed method was written mainly with Python 2.7 programming language and an ML library known as WEKA, version 3.8.0 which has several learning algorithms including C4.5. With respect to the hardware used, a personal computer having a random access memory of 4GB RAM, storage size 500GB with Processor type Intel(R) Core(TM) i3 CPU @2.53GHz speed was used. Since WEKA is written in Java, its integration with python is made possible using java virtual machine wrapper for Python.

6 Dataset

The KDDCup'99 dataset contains historical data prepared by [29] for evaluating IDSs. It has been a common dataset used for training and testing models of ML algorithms, hence used as a benchmark. The dataset comprises of a training and test set. The Training set contains 22 different types of attacks which are mixed with normal connections while the test set contains both the 22 attacks and 17 news attack which total to 39 attacks. The records in the dataset consist of 41 attributes (features). Due to the inherent challenges of the dataset, NSL-KDD dataset which is an extract from the original KDDCUP'99 dataset is used. It contains about 125973 instances of TCP/IP connections which can be utilized for training and 22544 instances for testing designed models. The instances in the dataset are data connections as they flow from source to destination. For experimental purpose, each instance is given a "label" to identify it as either an "attack" or "normal" connection. The values of the features can also be divided based on their type: numeric or symbolic. One of the advantages of NSL-KDD dataset is that the quantity is reasonable enough to be trained as a whole by most algorithm as compared to KDDCUP'99 where small portions of the dataset are used for training and testing [32]. Table 2 shows the list of features and their types.

Also, both the 22 attacks in the training set and 39 attacks in the test set can be further categorized into four

Table 2: Total list of features in NSL-KDD dataset

No.	Feature	Type
1	Duration	Numeric
2	protocol_type	Symbolic
3	Service	Symbolic
4	flag	Symbolic
5	src_bytes	Numeric
6	dst_bytes	Numeric
7	land	Numeric
8	wrong_fragment	Numeric
9	urgent	Numeric
10	hot	Numeric
11	num_failed_logins	Numeric
12	logged_in	Numeric
13	num_compromised	Numeric
14	root_shell	Numeric
15	su_attempted	Numeric
16	num_root	Numeric
17	num_file_creations	Numeric
18	num_shells	Numeric
19	num_access_files	Numeric
20	num_outbound_cmds	Numeric
21	is_host_login	Numeric
22	is_guest_login	Numeric
23	count	Numeric
24	srv_count	Numeric
25	serror_rate	Numeric
26	srv_serror_rate	Numeric
27	rerror_rate	Numeric
28	srv_rerror_rate	Numeric
29	same_srv_rate	Numeric
30	diff_srv_rate	Numeric
31	srv_diff_host_rate	Numeric
32	dst_host_count	Numeric
33	dst_host_srv_count	Numeric
34	dst_host_same_srv_rate	Numeric
35	dst_host_diff_srv_rate	Numeric
36	dst_host_same_src_port_rate	Numeric
37	dst_host_srv_diff_host_rate	Numeric
38	dst_host_serror_rate	Numeric

39	dst_host_srv_serror_rate	Numeric
40	dst_host_rerror_rate	Numeric
41	dst_host_srv_rerror_rate	Numeric

broader groups which are Dos, Probe, U2R and R2L as described below also in Tables 3 and 4:

- **Denial of Service(DoS)**: An attack which creates excessive computational request to the responding system such that there are no more resources on the destination side to respond to a request from legitimate users which is the goal of the attacker.
- **Probe attacks**: The attacker gets sensitive information about a network by scanning data without access permission with the aim of breaking its security control.
- **Remote-to-Local (R2L)**: uses the vulnerability of a system to get a normal user account from a remote location to gain local access to a system.
- **User-to-Root(U2R)**: It is when an attacker uses means such as social engineering, password sniffing to get the password of a normal user of a system, from there he can exploit some vulnerability of the system to gain root access.

Table 3: Categories of attacks in NSL-KDD training and test dataset

Category	**Actual Attacks in Training**	**Additional Attacks in Test set**
DoS	Neptune, smurf, teardrop, pod, land, back	apache2, mailbomb, processtable, udpstorm
Probing	satan, ipsweep, nmap, portsweep	mscan, saint
R2L	imap, warezmaster, phf, multihop, guess_passwd,spy, warezclient, ftp_write	httptunnel, named, sendmail, snmpgetattack, xlock, xsnoop
U2R	loadmodule, buffer_overflow, rootkit, perl	ps, snmpguess, sqlattack, worm, xterm

Table 4: Categories of attacks in NSL-KDD training and test dataset

Training Dataset		**Testing dataset**	
Attack Class	**Quantity**	**Attack Class**	**Quantity**
Normal	67343	Normal	9711
DoS	54927	DoS	7458
Probe	11656	Probe	2421
R2l	995	R2L	2754
U2R	52	U2R	200
Total	125973	**Total**	**22544**

7 Results and Discussion

This section provides experimental results of the proposed feature selection search technique. It uses the standard ML metric to evaluate the strength of the resulting classifier. These metric includes classification rate, FPR, TPR, precision, recall, F-measure and Auc. Also, it is compared to recent ML techniques to see how well it performs.

The performance of the classifier is evaluated both on the training set and more importantly the test set provided by NSL-KDD. Table 5 shows a detailed performance of the proposed feature set on the C4.5 algorithm. Comparing the result of Table 5 with Table 6 where Bagging ensembles was proposed by [12], it is clear that our method performs better both regarding accuracy and FPR. Also, Table 8 presents results from [22] which indicate the proposed model is better when matched. Likewise, It performs better in terms of accuracy than both multi and binary classification using ANN proposed by [15] as shown in Table 7.

Furthermore, the area under the curve (AUC) after feature selection in Figure 3 shows an appreciable increase (Auc = 0.9172) compared to Figure 2 (Auc = 0.84) which shows the classifier is balanced in its classification.

Table 5: Performance of proposed model on NSL-KDD dataset

Datasets	**Training (100%)**	**Testing**
Classification rate	99.81%	88.73%
Error rate	0.1873%	11.27%
TP Rate (Weighted Avg.)	0.998	0.89
FP Rate (Weighted Avg.)	0.002	0.093
Precision (Weighted Avg.)	0.998	0.90
Recall (Weighted Avg.)	0.998	0.89
F-Measure (Weighted Avg.)	0.998	0.89

Table 6: Performance on NSL-KDD dataset [12]

Performance Metric	On Training Set	On Test Set
Classifier Accuracy	99.6761%	81.2988%
False Positive Rate	0.003	0.148

Table 7: Performance on NSL-KDD dataset [15]

Classifier Method	Detection Accuracy	FPR (attack)
SOM	75.49	5.77
binary	81.2	4.23
five class	79.9	-

Table 8: Performance on NSL-KDD dataset [22]

Datasets	Training (100%)	Testing
Classification rate	99.01%	82.37%
Error rate	0.98%	17.62%
TP Rate (Weighted Avg.)	0.99	0.82
FP Rate (Weighted Avg.)	0.007	0.15
Precision (Weighted Avg.)	0.99	0.74
Recall (Weighted Avg.)	0.99	0.82
F-Measure (Weighted Avg.)	0.99	0.77

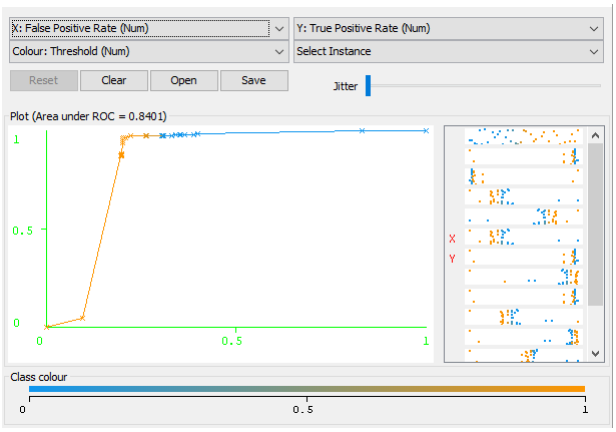

Figure 2: ROC before feature selection

We can also see the effect of the feature reduction in the training time of C4.5, as shown in Figure 4, which shows a 76% decrease in the training time.

In general, the achievements of this classifier is attributed to the choice of a feature set. The DDE algorithm helped in obtaining the right set of features as

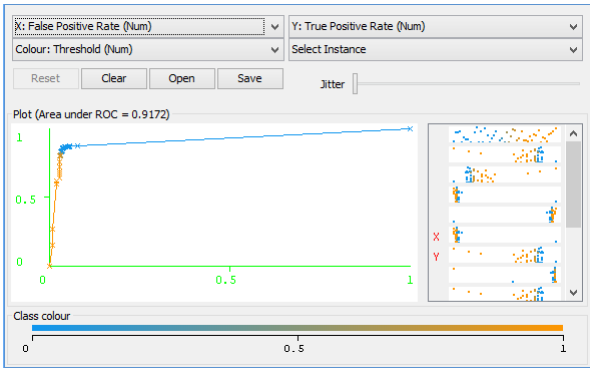

Figure 3: ROC after feature selection

Table 9: List of features used

No.	Feature
1	Duration
2	Service
3	src_bytes
4	land
5	hot
6	num_compromised
7	root_shell
8	is_guest_login
9	serror_rate
10	rerror_rate
11	srv_rerror_rate
12	dst_host_count
13	dst_host_srv_count
14	dst_host_same_srv_rate
15	dst_host_srv_serror_rate
16	dst_host_rerror_rate

shown in Table 9. It was observed that the feature used by C4.5 as the root node of the tree, before and after the proposed feature selection technique was "src_bytes" which tags the feature as the most important for their initial split. On the other hand, the "service" feature contributed most in making the final split. Also, despite the reduction of features, the important basic TCP features (1 - 4), content-based features (4 - 8), time-related traffic features (9 - 11), and host-based traffic features (12 - 16) were captured without loss in accuracy.

8 Conclusion

This work proposed an efficient feature selection technique for NID using DDE. Evaluation of the proposed

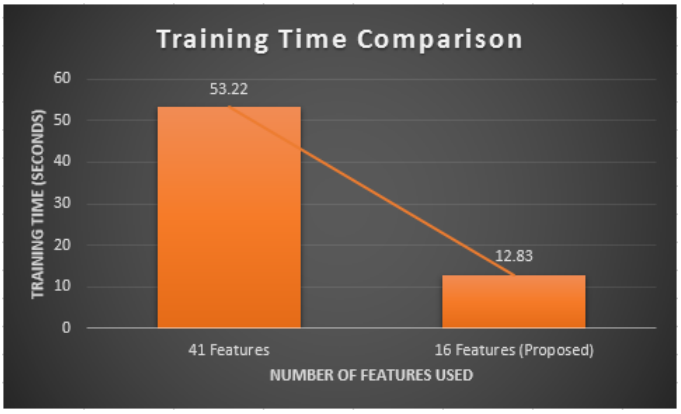

Figure 4: Comparison of 41 vs 16 features training times

method is done to both on the training and test set to rate the resulting classifier compared to other existing classifiers using standard ML performance metric. The computational result shows that this technique is able to identify 16 features capable of classifying the connections in the NSL-KDD dataset with high accuracy, low error and FPR. While it achieves 99.92% classification accuracy on the training set using 10-fold cross-validation, it is able to classify new attacks from in the test set with 88.73% accuracy. Aside the result above, the reduced feature set helps in reducing both the training and testing time used by the classifier (C4.5). Hence, the proposed method is greatly encouraged.

In future, we intend to extract connections from live networks having these set of features but with recent forms of attack to further test the model, because other datasets which have been provided by some authors do no have the exact features as in the NSL-KDD dataset.

Acknowledgments

The University of KwaZulu-Natal, South Africa supported this study. The authors gratefully acknowledge the anonymous reviewers for their valuable comments.

References

[1] G. L. Agrawal, H. Gupta, "Optimization of C4. 5 decision tree algorithm for data mining application," *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, no. 3, pp. 341–345, 2013.

[2] O. Y. Al-Jarrah, A. E. M. Siddiqui, P. D. Yoo, S. Muhaidat, K. Kim, "Machine-learning-based feature selection techniques for large-scale network intrusion detection," in *Proceedings of The 34th International Conference on Distributed Computing Systems Workshops (ICDCSW'14)*, pp. 177–181, Madrid, Spain, June-July 2014.

[3] A. Anurag, "Network neutrality: Developing business model and evidence based net neutrality regulation," *International Journal of Electronics and Information Engineering*, vol. 3, no. 1, pp. 1–9, 2015.

[4] L. Atzori, A. Iera, G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[5] A. S. A. Aziz, A. T. Azar, M. A. Salama, A. E. Hassanien, S. E. O. Hanafy, "Genetic algorithm with different feature selection techniques for anomaly detectors generation," in *Proceedings of The Federated Conference on Computer Science and Information Systems (FedCSIS'13)*, pp. 769–774, Krakow, Poland, Sept. 2013.

[6] S. E. Benaicha, L. Saoudi, S. E. B. Guermeche, O. Lounis, "Intrusion detection system using genetic algorithm," in *Proceedings of The Conference on Science and Information Conference (SAI'14)*, pp. 564–568, London, UK, Aug. 2014.

[7] N. Cleetus, K. A. Dhanya, "Genetic algorithm with different feature selection method for intrusion detection," in *Proceedings of The First International Conference on Computational Systems and Communications (ICCSC'14)*, pp. 220–225, Dec. 2014.

[8] Y. Chen, A. Abraham, B. Yang, "Feature selection and classification using flexible neural tree," *Neurocomputing*, vol. 70, no. 1, pp. 305–313, 2006.

[9] S. Das, P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[10] D. H. Deshmukh, T. Ghorpade, P. Padiya, "Intrusion detection system by improved preprocessing methods and naïve bayes classifier using NSL-KDD 99 dataset," in *Proceedings of The International Conference on Electronics and Communication Systems (ICECS'14)*, pp. 1–7, Feb. 2014.

[11] S. Elsayed, R. Sarker, J. Slay, "Evaluating the performance of a differential evolution algorithm in anomaly detection," in *Proceedings of The Congress on Evolutionary Computation (CEC'15)*, pp. 2490–2497, Sendai, Japan, May 2015.

[12] D. P. Gaikwad, R. C. Thool, "Intrusion detection system using bagging ensemble method of machine learning," in *Proceedings of The International Conference on Computing Communication Control and Automation (ICCUBEA'15)*, pp. 291–295, Pune, Maharashtra, India, Feb. 2015.

[13] T. Garg, Y. Kumar, "Combinational feature selection approach for network intrusion detection system," in *Proceedings of The Third International Conference on Parallel, Distributed and Grid Computing (PDGC'14)*, pp. 82–87, India, Dec. 2014.

[14] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[15] B. Ingre, A. Yadav, "Performance analysis of NSL-KDD dataset using ann," in *Processdings of The*

International Conference on Signal Processing And Communication Engineering Systems (SPACES'15), pp. 92–96, Jan. 2015.

[16] M. Kumar, K. Dutta, I. Chopra, "Impact of wormhole attack on data aggregation in hieracrchical WSN," *International Journal of Electronics and Information Engineering*, vol. 1, no. 2, pp. 70–77, 2014.

[17] A. J. Malik, W. Shahzad, F. A. Khan, "Network intrusion detection using hybrid binary pso and random forests algorithm," *Security and Communication Networks*, vol. 8, no. 16, pp. 2646–2660, 2015.

[18] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Transactions on Information System Security*, vol. 3, no. 4, pp. 262–294, 2000.

[19] MIT Lincoln Labs, "Darpa intrusion detection evaluation," June 2015. (`https://www.ll.mit.edu/ideval/data/`)

[20] NSL-KDD, "NSL-KDD data set for network-based intrusion detection systems," June 2015. (`https://web.archive.org/web/20150205070216/http://nsl.cs.unb.ca/NSL-KDD/`)

[21] E. U. Opara, O. A. Soluade, "Straddling the next cyber frontier: The empirical analysis on network security, exploits, and vulnerabilities," *International Journal of Electronics and Information Engineering*, vol. 3, no. 1, pp. 10–18, 2015.

[22] M. S. Pervez, D. M. Farid, "Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing svms," in *Proceedings of 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA'14)*, pp. 1–6, Dhaka, Bangladesh, Dec. 2014.

[23] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[24] G. P. Rout, S. N. Mohanty, "A hybrid approach for network intrusion detection," in *Proceedings of The Fifth International Conference on Communication Systems and Network Technologies (CSNT'15)*, pp. 614–617, Gwalior, India, Apr. 2015.

[25] Z. Salek, F. M. Madani, R. Azmi, "Intrusion detection using neuarl networks trained by differential evaluation algorithm," in *Proceedings of The 10th International Conference on Information Security and Cryptology (ISCISC'13)*, pp. 1–6, Yazd, Iran, Aug. 2013.

[26] S. Samira, M. Zaiton, A. Idawaty, B. Mehdi, "GA and SVM algorithms for selection of hybrid feature in intrusion detection systems," *International Review on Computers and Software*, vol. 10, no. 3, pp. 2, 2015.

[27] M. Sreenath, J. Udhayan, "Intrusion detection system using bagging ensemble selection," in *Proceedings of The International Conference on Engineering and Technology (ICETECH'15)*, pp. 1–4, India, Mar. 2015.

[28] G. Stein, B. Chen, A. S. Wu, K. A. Hua, "Decision tree classifier for network intrusion detection with ga-based feature selection," in *Proceedings of the 43rd Annual Southeast Regional Conference*, vol. 2, pp. 136–141, New York, USA, 2005.

[29] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, P. K. Chan, "Cost-based modeling for fraud and intrusion detection: Results from the jam project," in *Proceedings of The Conference on DARPA Information Survivability Conference and Exposition (DISCEX'00)*, vol. 2, pp. 130–144, California, USA, Jan. 2000.

[30] R. Storn, K. Price, "Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[31] M. F. Tasgetiren, Q. K. Pan, P. . Suganthan, Y. C. Liang, "A discrete differential evolution algorithm for the no-wait flowshop scheduling problem with total flowtime criterion," in *Proceedings of The Symposium on Computational Intelligence in Scheduling*, pp. 251–258, Apr. 2007.

[32] M. Tavallaee, E. Bagheri, W. Lu, A. A. Ghorbani, "A detailed analysis of the KDD cup 99 data set," in *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications (CISDA'09)*, pp. 53–58, Piscataway, NJ, USA, 2009.

[33] X. Z. Wang, "ACO and SVM selection feature weighting of network intrusion detection method," *Analysis*, vol. 9, no. 4, pp. 129–270, 2015.

Biography

Ebenezer O. Popoola received B.Sc. in Electronic and Electrical Engineering from Obafemi Awolowo University, Ile-Ife, Nigeria. He worked in the telecommunication industry before proceeding for his M.Sc. Degree in Computer Science at the University of KwaZulu-Natal, South Africa. His primary interests are communication security, computer vision, machine learning and optimization.

Aderemi O. Adewumi received the B.Sc. and M.Sc. degrees in Computer Science from the University of Lagos, Nigeria, and PhD in Computational & Applied Mathematics from the University of Witwatersrand, South Africa, with a specialty in optimization and computational intelligence. He is currently with the University of KwaZulu-Natal, Durban, South Africa, where he leads the Optimization and Modeling Research Group in the School of Mathematics, Statistics and Computer Science. His current research interests include optimization and artificial intelligence, with a particular interest in computational intelligence, machine learning, and intelligent solutions to real-world global optimization problems.