IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 6, 2019, accepted March 12, 2019, date of publication March 18, 2019, date of current version April 5, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2905633

A Deep Learning Method With Filter Based Feature Engineering for Wireless Intrusion Detection System

SYDNEY MAMBWE KASONGO AND YANXIA SUN
Department of Electrical and Electronic Engineering Science, University of Johannesburg, Johannesburg 2006, South Africa

Corresponding author: Sydney Mambwe Kasongo (sydneybleuops@gmail.com)

This work was supported in part by the South African National Research Foundation under Grant 112108 and Grant 112142, in part by the South African National Research Foundation Incentive Grant under 95687, in part by the Eskom Tertiary Education Support Programme Grant, and in part by the Research Grant from URC of the University of Johannesburg.

ABSTRACT In recent years, the increased use of wireless networks for the transmission of large volumes of information has generated a myriad of security threats and privacy concerns; consequently, there has been the development of a number of preventive and protective measures including intrusion detection systems (IDS). Intrusion detection mechanisms play a pivotal role in securing computer and network systems; however, for various IDS, the performance remains a major issue. Moreover, the accuracy of existing methodologies for IDS using machine learning is heavily affected when the feature space grows. In this paper, we propose a IDS based on deep learning using feed forward deep neural networks (FFDNNs) coupled with a filter-based feature selection algorithm. The FFDNN-IDS is evaluated using the well-known NSL-knowledge discovery and data mining (NSL-KDD) dataset and it is compared to the following existing machine learning methods: support vectors machines, decision tree, K-Nearest Neighbor, and Naïve Bayes. The experimental results prove that the FFDNN-IDS achieves an increase in accuracy in comparison to other methods.

INDEX TERMS Deep learning, feature extraction, intrusion detection, machine learning, wireless networks.

I. INTRODUCTION

Computer networks and wireless networks in particular are subjects to a myriad of security threats and attacks. The security challenges that have to be solved originate from the open nature, the flexibility and the mobility of the wireless communication medium [1], [2]. In an effort to secure these networks, various preventive and protective mechanisms such as intrusion detection systems (IDS) were developed [3]. Primarily, IDS can be classified as: host based intrusion detection systems (HIDS) and network based intrusion detection systems (NIDS) [4]. Furthermore, both HIDS and NIDS can be categorized into: signature-based IDS, anomaly-based IDS and hybrid IDS [5], [6]. An Anomaly based IDS analyses the network under normal circumstances and flags any deviation as an intrusion. A signature-based IDS relies on a predefined database of known intrusions to pinpoint an intrusion. In this case, a manual update of the database is performed by the system administrators.

The associate editor coordinating the review of this manuscript and approving it for publication was Shagufta Henna.

In terms of performance, an IDS is considered effective or accurate in detecting intrusions when it concurrently achieves low false alarm rates and high classification accuracy [7]; therefore, decreasing the law false alarm rate as well as increasing the detection accuracy of an IDS should be one of the crucial tasks when designing an IDS. In this paper, the terms wireless intrusion detection system (WIDS) and intrusion detection system (IDS) will be used interchangeably.

In a bid to build efficient IDS systems, Machine Learning (ML) approaches are used to identify various types of attacks. ML is the scientific study of procedures, algorithms and statistical models used by computer systems to solve complex problems and it is considered a subset Artificial Intelligence (AI) citeb8. Since the issue of intrusion detection is a classification problem, it can be modeled using ML techniques. It has been proven that developing IDS using ML methods can produce high levels of accuracy citeb5; however, citeb9 showed that the most accurate and effective IDS has not been discovered and that each IDS solution presents its own advantages and handicaps under various conditions.

The most popular ML approaches to intrusion detection include K-Nearest-Neighbors (KNN)citeb10, Decision Tree (DT)citeb11, Support Vector Machines (SVM)citeb12, Random Forest (RF) citeb13, Naive Bayes (NB) citeb14 and Multi-Layered Perceptions (MLP) associated with all Deep Learning (DL) Methodologies citeb15, b16, b17. An IDS generally treats large amount of data that causes ML techniques such as the ones in citeb10,b11,b12, b13,b14 to perform poorly; therefore is imperative to devise appropriate strategies and classification approaches to overcome the issue of under-performance. This paper focuses on DL to try to improve on the shortcomings of existing systems.

DL was first proposed by Professor Hinton [18] and it is an advanced sub-field of ML that simplifies the modeling of various complex concepts and relationships using multiple levels of representation [19]. DL has achieved a great amount of success in fields such as language identification, image processing and pharmaceutical research [20]–[22]. This has prompted researchers to explore the application of DL theory to the intrusion detection classification problem.

The major characteristic that distinguishes DL from traditional ML methods is the improved performance of DL as the amount of data increases. DL algorithms are not well suited for problems involving small volumes of data because these algorithms require a considerable amount of data to be capable of learning more efficiently [9]. Although DL can handle a high throughput in terms of data, the questions of accuracy improvement and lowering of false-positive alarm rate still remain due to the ever-growing size of datasets used for IDS research. Moreover, as the datasets dilate in terms of volume; there is also an expansion of the input space and attack classification dimension. Consequently, instances of misclassification are prevalent, which in turn trigger an increase in false positive alarm rate and impacts negatively the overall system performance. Therefore, it is crucial to implement solutions that are capable of selecting only the needed features to perform an optimal classification operation.

Feature engineering (FE) have become a key topic in many ML research domains [23]–[26]. As part of FE, the feature selection algorithms fall into the following the categories: filter model, wrapper model and hybrid model. The filter model bases itself on the intrinsic nature of the data and it is independent of the classifier used. The wrapper method evaluates the performance of the classification algorithm used on a candidate feature subset, whereas the hybrid method is a combination the wrapper and filter algorithms [27]. The methodology proposed in this paper focuses on a filter-based approach as the two latter techniques are computationally expensive [28].

The major contributions of this paper are outlined as follow:

- A Feature Extraction Unit (FEU) is introduced. By using filter-based algorithms, the FEU generates optimal subsets of features with minimum redundancy.
- We scrutinize the performance of the following existing classification algorithms applied to IDS without the FEU by using the NSL-KDD dataset: k-nearest neighbor (KNN), support vector machine (SVM), Decision Tree (DT), Random Forest (RF) and Naive Bayes (NB). Moreover, we study the performance of those algorithms coupled with the FEU.
- A feed-forward deep neural network (FFDNN) is introduced. We study its performance using the FEU and the NSL-KDD dataset. After the comparison to KNN, SVM, DT, RF and ND, the FEU-FFDNN proves to be very appropriate for intrusion detection systems. Furthermore, Experimental results demonstrate that depth and the number of neurons (nodes) used for an FFDDN classifier have a direct impact on its accuracy.

The rest of this paper is organized as follow: Section II of the paper provides a background on wireless networks. Section III gives an account of similar research with a focus on ML based IDS as well as various methods for features selection. Section IV details a background on traditional machine learning classifiers that are also explored in this work. Section V of this document provides an architecture of the proposed method for wireless intrusion detection. Section VI details the experimental setup used in this research as well as the tools used to design, implement, evaluate and test the following classifiers: SVM, DT, RF, NB, KNN and FFDNN, and the results are discussed. Section VII concludes the paper.

II. BACKGROUND: WIRELESS NETWORKS

In recent years, the growth of wireless networks has been very predominant over wired ones. Wireless communication is attractive because it does not require any wired additional infrastructure for the communication media. Today, the most popular form of wireless networks are Wireless Local Area networks (WLANs). WLANs form part of the IEEE 802.11 family and are intensively used as an effective alternative to wired communication in various areas such as industrial communication and in building communication. A myriad of security mechanisms including Wired Equivalent Protection (WEP) and WiFi Protected Access (WAP, WAP2) have been mainly used to secure and protect WLANs; however, they have shown many flaws when it comes to threats such as Denial of Service (DoS) attacks, network discovery attacks, brute force attacks, etc [2], [41], [43]. In order to reinforce WLANs security against those vulnerabilities, IDSs are generally implemented. In this research, we focus on an IDS for WLANs using DL approach. Furthermore, since wired and wireless IDS systems research go hand in hand, this work reviews strategies used both in wired and wireless IDS research using ML and DL.

III. RELATED WORK

This section provides an account of previous studies on feature selection methods in general as well as intrusion detection systems using ML and DL techniques.

The research conducted in [19] presented a deep learning based intrusion detection system that made use of non-symmetric deep auto-encoder (NDAE) for feature learning and a classification methodology using stacked NDAEs. An NDAE is an auto-encoder made of non-symmetrical multiple hidden layers. In simple terms, it is a deep neural network composed of many non-symmetrical hidden layers. The evaluation of the IDS scheme was made using two datasets: the KDDCup 99 and the NSL-KDD. The performance of the multiclass classification experiment yielded an accuracy of 85.42% over the NSL-KDD dataset and an accuracy of 97.85% on the KDDCup 99 dataset.

In [26], the researchers gave an account of a multi-objective algorithm for feature selection labeled MOMI. This approach is centered on Mutual Information (MI) and considers the features redundancy and relevancy during the feature evaluation and selection process. The experiments carried out to evaluate MOMI's performance were conducted using the WEKA tool [35] with three separate datasets. Two classifiers, namely Naive Bayes (NB) and support vector machine (SVM) were used. The results of this research suggested that MOMI was able to select only the features needed for the best performance.

Chakraborty and Pal [29] presented a feature selection (FS) algorithm using a multilayer percetron (MLP) framework with a controlled redundancy (CoR). This approach is labelled as FSMLP-CoR. An MLP is a neural network with an input layer, multiple hidden layers and an output layer [30] and it is generally used for approximation, classification, regression, and prediction in many domains [31]–[34]. In this case, an MLP was used to identify and drop those features that are not relevant in resolving the problem at hand. The FSMLP-CoR was tested using 23 datasets and the results led researchers to conclude that it was effective in selecting important features.

In [36], an ant colony optimization (ACO) technique was applied for feature selection on the KDDCup 99 dataset for intrusion detection. The KDD Cup 99 dataset has 41 features. ACO was inspired by how ants use pheromones in a bid to remember their path. ACO has different variations. In this research, the authors used the ant colony system (ACS) with two level pheromones update. The proposed solution was evaluated using the binary SVM classifier library in WEKA (LibSVM) [35]. The results revealed that a higher accuracy is obtained with an optimal feature subset of 14 inputs.

The research in [37] proposed a wrapper based feature selection algorithm for intrusion detection using the genetic algorithm (GA) as an heuristic search method and Logistic Regression (LR) as the evaluating learning algorithm. The whole approach is labeled as GA-LR. GA originates from the natural selection process and it is under the category of evolutionary based algorithms [38]. GA has the following building blocks: an initial population, a fitness function, a genetic operator (variation, crossover and selection) and a stopping criterion. The experiments conducted to evaluate the GA-LR were done using the KDD Cup 99 Dataset and the UNSW-B15 Dataset. Decision Tree classifiers were applied to candidates feature subsets and the results suggested that GA-LR is an efficient method.

Wang *et al.* [39] took a different direction in terms of the feature engineering approach by using a feature augmentation (FA) algorithm rather than a feature reduction one. The classifier used in this research was the SVM and the FA algorithm used was the logarithm marginal density ratio transformation. The goal was to obtain newly improved features that would ultimately lead to a higher performance in detection accuracy. The evaluation of the proposed scheme was conducted using the NSL-KDD dataset and the outcomes from the empirical experiments suggested the FA coupled with the SVM yielded a robust and improved overall performance in intrusion detection capacity.

In [40], an intrusion detection system (IDS) was designed and modelled based on DL using Recurrent Neural Networks (RNNs). RNNs are neural networks whereby the hidden layers act as the information storage units. The benchmark dataset used in this research was the NSL-KDD. The RNN-IDS was compared to the following commonly used classification methods: J.48, Random Forest and SVM. The accuracy (AC) was mainly used as the performance indicator during the experiments and the results suggested that RNN-IDS presented an improved accuracy of intrusion detection compared to traditional machine learning classification methods. These results reinforced the assumption that DL based intrusion detection systems are superior to classic ML algorithms. In the binary classification scheme, a model with 80 hidden nodes, a learning rate of 0.1 achieved an accuracy of 83.28% whereas in the multiclass classification using 5 classes, a model with 80 hidden neurons and learning rate of 0.5 got an accuracy of 81.29%.

The approach proposed in [41] used a deep learning approach to intrusion detection for IEEE 802.11 wireless networks using stacked auto encoders (SAE). A SAE is a neural network created by stacking together multiple layers of sparse auto encoder. The experiments undertook in this research were made using the Aegean Wireless Intrusion Dataset (AWID) that is comprised of 155 attributes with the last attribute representing the class that can take the following values: injection, flooding, impersonation and normal. According to Thing [41], this was the first work that proposed a deep learning approach applied to IEEE 802.11 networks for classification. The overall accuracy achieved in this work was 98.6688%.

Ding and Wang [42] investigated the use of DL for intrusion detection technology using the KDDCup 99 Dataset. The architecture used for the neural network model consisted of 5 hidden layers of 10-20-20-40-64 dense feed forward (fully connected layers). The activation function used in this research was the ReLU (Rectified Linear Unit) and the back-propagation method for training this model was the Adam optimizer (Ad-op). The Ad-op was used in a bid to increase the training speed and to prevent overfitting. Although this research yielded some advancements, it equally

showed no significant improvement in detecting rare attacks types (U2R and R2L) present in the dataset.

In [43], an ML approach to detect flooding Denial of Service (DoS) in IEEE 802.11 networks was proposed. The dataset used in this research was generated by the authors in a computer laboratory. The setup was made of 40 computers in which seven were designated as attackers to lunch the flooding DoS and each of the legitimate node was connected to any of the available five Access Points (APs). The obtained dataset was segmented in the following two portions: 66% for ML training and 34% for ML testing. Using the WEKA tool [35], six classifications ML learning algorithms were applied consecutively, namely: SVM, Naive Bayes, Naive Bayes Net, Ripple-DOwn Rule Learner (RIDOR), Alternating Decision Tree and Adaptive Boosting (AdaBoost). The empirical results based on the accuracy and the recall numbers suggested that AdaBoost was more efficient than the other algorithms.

In [44], a performance comparison of SVM, Extreme Learning Machine (ELM) and Random Forest (RF) for intrusion detection was investigated using the NSL-KDD as the benchmark dataset. Each of the ML algorithms used in this investigation was evaluated using the following performance metrics: Accuracy, Precision and Recall. The outcome of the experiments showed that ELM outperformed RF and SVM; consequently, the authors concluded that ELM is a viable option when designing and implementing intrusion detection systems.

IV. BACKGROUND ON TRADITIONAL MACHINE LEARNING CLASSIFIERS

A. SUPPORT VECTOR MACHINE

Support Vector Machines (SVM) is one of the most popular ML techniques applied to Big Data and used in ML research. SVM is a supervised machine learning method that is used to classify different categories of data. SVM is able to solve the complexity of both linear and non-linear problems. SVM works by generating a hyperplane or several hyperplanes within a high-dimensional space to separate data and the ones that optimally split the data per class type are selected as the best [44].

B. K-NEAREST NEIGHBOR

K-Nearest Neighbor (KNN) is another ML method used to classify data. The KNN algorithm bases itself on the standard Euclidean distance between instances in a space and can be defined as follow [45]: let x and y instances in space P, the distance between x and y, $d(x, y)$, is given the following expression:

$$d(x, y) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2} \tag{1}$$

where n represents the total number of instances. The KNN method classifies an instance x_0 within a space by calculating the Euclidean distance between x_0 and k closet samples within the training set and x_0 takes the label of k most similar neighbors [46].

C. NAIVE BAYES

Naive Bayes (NB) classifiers are simple classification algorithms based on *Bayes' Theorem* [47]. Given a dataset, an NB classifier assumes a *"naive"* independence between the features. Let X an instance with n features to be classified represented by the vector $X = (x_1, \ldots, x_n)$. In order to figure out the class C_k for X, NB does the following:

$$p(C_k|X) = \frac{p(X|C_k)p(C_k)}{P(X)} \tag{2}$$

And the class for X is assigned using the following expression:

$$y = \underset{k\in\{1,\ldots,K\}}{\operatorname{argmax}}\, p\,(C_k) \prod_{i=1}^{n} p\,(X_i|C_k) \tag{3}$$

where y is the predicted label.

D. DECISION TREE AND RANDOM FOREST

Decision Tree (DT) algorithm is widely used in data mining and ML. Given a dataset with labeled instances (training), DT algorithm generates a predictive model in a shape of a tree capable of predicting the class of unknown records [14]. A DT has three main components: a root node, internal nodes and category nodes. The classification processes happens in a top-down manner and an optimal decision is reached when the correct category of leaf node is found. A Random Forest classifier on the other hand applies multiple DTs on a given dataset for classification.

V. PROPOSED METHOD FOR WIRELESS INTRUSION DETECTION

A. FEED FORWARD DEEP NEURAL NETWORKS

Deep neural networks (DNNs) are widely used in ML and DL to solve complex problems. The most basic element of a DNN is an artificial neuron (AN) which is inspired from biological neurons within the human brain. An AN computes and forwards the sum of information received at its input side. Due the the non-linearity of real life problems and in a bid to enhance learnability and approximation, each AN applies an activation function before generating an output [48]. This activation function can be a Sigmoid, $\sigma = \frac{1}{1+e^{-t}}$; a Rectified Linear Unit (ReLU): $f(y) = max(0, y)$; or an hyperbolic tangent shown in expression (4).

$$tanh(y) = \frac{1 - e^{-2y}}{1 + e^{-2y}} \tag{4}$$

The above-mentioned activation functions have advantages and drawbacks; moreover, their optimal performance is problem specific. Traditionally, artificial neural networks (ANNs) have an input layer, one to three hidden layers and an output layer as shown in Fig. 1; whereas DNNs may contain three to tens or hundreds of hidden layers [49]. There is no general rule for determining whether an ANN is deep or not. For the

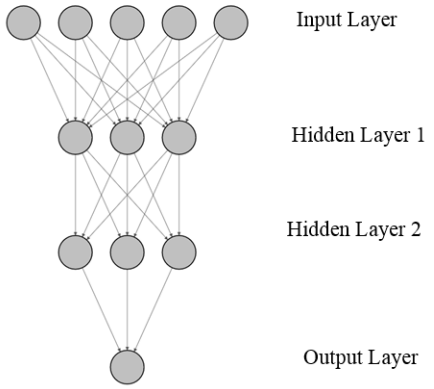


FIGURE 1. Feed forward neural network architecture.

sake of our research, we will consider a DNN to be a neural network with two or more hidden layers. In a Feed Forward DNN, the flow of information goes in one direction only: from the input layers via the hidden layers to the output layers. Neurons within the same layer do not communicate. Each AN in the current layer is fully connected to all neurons in the next layer as depicted in Fig.1.

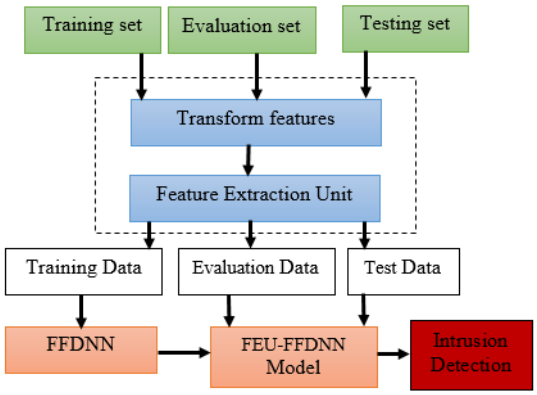


FIGURE 2. Proposed FFDNN architecture.

The block diagram in Fig. 2 presents the architecture of the proposed Feed Forward Deep Neural Network (FFDNN) IDS. In this architecture, the first step consists of the separation of raw data. It is crucial to split the main training set between two main sets: the reduced training set and the evaluation set. The evaluation dataset is used to validate the training process. The test set has a totally different data distribution from the training and evaluation (validation) sets. The second step involves a feature transformation process and a two-way normalization process of the raw data as well as a feature extraction (selection) procedure based on Information Gain. It is important to transform and to normalize the data because most of the features within a dataset come in different formats that can be numerical or nonnumberical. The last processes of the architecture are the models training and testing using the FFDNN and the FEU-FFDNN. Since the training and the validation data originate from the same data distribution, it is important to ensure that during the training process, the selected model doesn't train on the validation data because training the model on previously seen data may cause the final model to perform poorly. The next sections explain in detail the role of each of the components in Fig. 2.

B. DATASET

In the proposed research, the NSL-Knowledge Discovery and Data mining (NSL-KDD) which is an improved version of the KDDCup 99 [19] is used to train, evaluate and test the designed system shown in Fig 2. The NSL-KDD is considered a benchmark dataset in IDS research and it is used for both wired and wireless systems [19], [39], [40], [44]. The NSL-KDD comprises one class label categorized in the following major groups: Normal, Probe, Denial of Service (DoS), User to Root (U2R) and Remote to User (R2L). Furthermore, the NSL-KDD is made of 41 features of which three are nonenumeric and 38 are numeric as depicted in Table 1.

The NSL-KDD comes with two set of data: the training set (KDDTrain+ full) and the test sets (KDDTest+ full and KDDTest-21). In this research, we use the KDDTrain+ and the KDDTest+. KDDTrain+ is divided into two partitions: the KDDTrain+75, which is 75 % of the KDDTrain+ and it will be used for training, the KDDTEvaluation that is 25 % the KDDTrain+ and it will be used for evaluation after the training process. Table 2 provides a breakdown of the components in each dataset.

C. FEATURE ENGINEERING

In a dataset, features may take different forms such as numeric and nonnumeric. DNN models can only process numeric

TABLE 1. NSL-KDD Features List.

No.	Feature Name	Category	No.	Feature Name	Category
f1	duration	numeric	f22	is_guest_login	numeric
f2	protocol_type	nonnumeric	f23	count	numeric
f3	service	nonnumeric	f24	srv_count	numeric
f4	flag	nonnumeric	f25	serror_rate	numeric
f5	src_bytes	numeric	f26	srv_serror_rate	numeric
f6	dst_bytes	numeric	f27	rerror_rate	numeric
f7	land	numeric	f28	srv_rerror_rate	numeric
f8	wrong_fragment	numeric	f29	same_srv_rate	numeric
f9	urgent	numeric	f30	diff_srv_rate	numeric
f10	hot	numeric	f31	srv_diff_host_rate	numeric
f11	num_failed_logins	numeric	f32	dst_host_count	numeric
f12	logged_in	numeric	f33	dst_host_srv_count	numeric
f13	num_compromised	numeric	f34	dst_host_same_srv_rate	numeric
f14	root_shell	numeric	f35	dst_host_diff_srv_rate	numeric
f15	su_attempted	numeric	f36	dst_host_same_src_port_rate	numeric
f16	num_root	numeric	f37	dst_host_srv_diff_host_rate	numeric
f17	num_file_creations	numeric	f38	dst_host_serror_rate	numeric
f18	num_shells	numeric	f39	dst_host_srv_serror_rate	numeric
f19	num_access_files	numeric	f40	dst_host_rerror_rate	numeric
f20	num_outbound_cmds	numeric	f41	dst_host_srv_rerror_rate	numeric
f21	is_host_login	numeric			

TABLE 2. Datasets breakdown.

Dataset Name	Normal	DoS	Probe	R2L	U2R	Total
KDDTrain+ Full	67343	45927	11656	995	52	125973
KDDTrain+75	50494	34478	8717	749	42	94480
KDDTEvaluation	16849	11449	2939	246	10	31493
KDDTest+ Full	9711	7458	2754	2421	200	22544

values; therefore it is crucial to transform all nonnumeric or symbolic features into a numerical counterpart. Within the NSL-KDD, $f2$ 'protocol_type', $f3$ 'service' and $f4$ 'flag' are symbolic features. We apply a mapping process in Scikit Learn [50] whereby all symbols are mapped to a unique numerical value. Moreover, it is important to transform and normalize features as they may have an uneven distribution. For instance, taking a look at $f5$ which represents 'src_bytes' in Table 1: $f5$ has values such as 12983 and values like 20; consequently, normalization is required to keep values within the same range for optimal processing. In this research, we apply a two-step normalization procedure. We first apply a logarithmic normalization shown in expression (5) to all the features so that we keep them within acceptable range and secondly, we linearly cap the values to be in this range [0, 5] using equation in (6).

$$x_{normalized} = log(x_i + 1) \tag{5}$$

$$x_{normalized} = (b - a)\frac{x_i - min(x_i)}{max(x_i) - min(x_i)} \tag{6}$$

where $b = 5$ and $a = 0$.

After the two step normalization process, the Feature Extraction Unit (FEU) has the role to rank the features using an algorithm based on Information Gain (IG) [51] which originates from Information Theory [52]. We will compute the IG of each feature with respect to the class. Unlike the standard correlation algorithms such as Pearson Linear Correlation Coefficient [53] that is only capable of establishing linear correlations between features, IG is capable of discovering nonlinear connection as well. In information theory, the measure of uncertainty of a variable X is called entropy, $H(X)$, and it is calculated as follow:

$$H(X) = -\sum_{x \in X} P(x) log_2(x) \tag{7}$$

And the conditional entropy of two random variables X and Y is determined using the following expression:

$$H(X|Y) = -\sum_{x \in X} P(x) \sum_{y \in Y} P(x|y) log_2(P(x|y)) \tag{8}$$

where P is the probability. IG is derived from the expressions in (7) and (8) as follow:

$$IG(X|Y) = H(X) - H(X|Y) \tag{9}$$

Therefore, a given feature Y possesses a stronger correlation to feature X than feature V if $IG(X|Y) > IG(V|Y)$.

D. ALGORITHMS FOR FEATURE ENGINEERING

Given a feature vector $F(f_1, \ldots, f_n)$ with $1 < n < T$, where T is the total number of features and C the class label in the dataset, the *Transform Features* module in Fig 2. applies Algorithm 1 as follow:

After the execution of Algorithm 1, we obtained a transformed feature vector, $F_{transformed}(f_1^t, \ldots, f_n^t)$, that is fed into Algorithm 2 to generate a vector, F_{ranked}, with features that are ranked by IG with respect to C.

Algorithm 1 Normalization Algorithm

Input: $F(f_1, \ldots, f_n)$, $1 < n < T$
Output: $F_{transformed}(f_1^t, \ldots, f_n^t)$:
  for i from 1 to n **do**
    if (f_i a symbolic feature) **then**
      apply sckikit learn mapping
      Step 1 normalize using $log(f_i + 1)$
      Step 2 normalize using $(b - a)\frac{f_i - min(f_i)}{max(f_i) - min(f_i)}$
    end if
    Step 1 normalize using $log(f_i + 1)$
    Step 2 normalize using $(b - a)\frac{f_i - min(f_i)}{max(f_i) - min(f_i)}$
  end for

Algorithm 2 Features IG Ranking Algorithm

Input: $F_{transformed}(f_1^t, \ldots, f_n^t)$
Output: F_{ranked}
  for i from 1 to n **do**
    compute IG: $IG_i(f_i|C) = H(f_i) - H(f_i|C)$
    if ($IG_i >= IG_{treshold}$) **then**
      load IG_i into F_{ranked}
    end if
  end for

E. ALGORITHM FOR TRAINING FFDNNs

Training feed forward neural networks consists of the following three major steps:

1) Forward propagation.
2) Back-propagation of the computed error.
3) Updating the weights and biases.

The algorithm used to train the FFDNNs is explained in Algorithm 3. Given a set of m training sample $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ and η the learning rate. We train FFDNNs presented in this research using the back propagation algorithm backed by a stochastic gradient descent (SDG) for the weights and biases update. Additionally, the cost function used to calculate the difference between the target and the obtained output is shown in this expression:

$$C(W, b; x, y) = \frac{1}{2}\|y - output\|^2 \tag{10}$$

VI. EXPERIMENTAL SETTING, RESULTS AND DISCUSSIONS

For the purpose of this research, we have used a Python based library, Scikit-Learn [50] which is widely used in machine learning and deep learning research. Our simulations were executed on an ASUS laptop with the following specifications: Intel Core i-3-3217U CPU @1.80GHz and 4.00G of RAM. The metrics used to evaluate the performance of the FFDNNs presented in this research are the accuracy in (11), the precision in (12) and the recall in (13). These indicators are derived from the confusion matrix shown in Table 3 and they are defined as follow:

- True positive (TP): Intrusions that are successfully detected by the proposed IDS.

Algorithm 3 Forward and Back-Propagation Algorithm

Input: W, b

Output: updated W, b

1: Forward propagate x_i through layers $l = L2, L3, \ldots L_{nl}$, ($nl$ is the subscript of the last layer) using $z^{l+1} = W^l a^l + b^l$ and $a^{l+1} = f(z^{l+1})$ with f, a rectified linear unit (ReLU) of this form $f(z) = max(0, z)$

2: Compute the error term ξ for each output unit i as follow:

$$\xi_i^{nl} = \frac{d}{d(z_i^{nl})}\frac{1}{2}\|y - output\|^2 = -(y_i - a_i^{nl}).f'(z_i^{nl})$$

3: For each hidden units in $l = nl - 1, nl - 2, \ldots, 2$, compute the following for each node i in l:

$$\xi_i^l = \sum_{j=1}^{s_{l+1}} W_{ji}^l \xi_j^{l+1}.f'(z_i^l)$$

4: Calculate the required partial derivatives with respect to weights and biases for each training example as follow:

$$\frac{d}{dW_{ij}^l}C(W, b; x, y) = a_j^l \xi_i^{l+1}$$
$$\frac{d}{db_i^l}C(W, b; x, y) = \xi_i^{l+1}$$

5: Update the weight and biases as follow:

$$W_{ij}^l = W_{ij}^l - \eta a_j^l \xi_i^{l+1}$$
$$b_i^l = b_i^l - \eta \xi_i^{l+1}$$

TABLE 3. Confusion matrix.

Actual Class ↓ \Predicted Class →	Anomaly	Normal
Anomaly	TP	FN
Normal	FP	TN

- False positive (FP): Normal / non-intrusive behaviour that is wrongly classified as intrusive by the IDS.
- True Negative (TN): Normal / non-intrusive behaviour that is successfully labelled as normal/non-intrusive by the IDS.
- False Negative (FN): Intrusions that are missed by the IDS, and classified as normal / non-intrusive.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (11)$$

$$Precision = \frac{TP}{TP + FP} \quad (12)$$

$$Recall = \frac{TP}{TP + FN} \quad (13)$$

The experiments were carried out in multiple phases using the NSL-KDD dataset explained in section V. The NSL-KDD has the following classes: Normal, DoS, Probe, U2R and R2L. For binary classification, we map the DoS, Probe, U2R and R2L classes to one class called "attack" and for multi-class classification, we use the dataset with its original five major classes. In this research, the following rule applies: a classifier performs better than another one when it yields a higher accuracy on previously unseen data that can be found in the KDDTest+ set.

A. PHASE 1: BINARY CLASSIFICATION WITH 41 FEATURES

This phase uses all 41 features for binary classification. We only apply Algorithm 1 to transform the inputs. In order to select the best FFDNN, we ran models with 41 units at the input layer, two nodes at the output layer and the following hidden nodes numbers: 30, 40, 60, 80 and 150. These numbers were selected by trial and error method. Moreover, we were also varying the number hidden layers as well as the learning rate. The details are presented in Table 4. For better performance analysis and for the purpose of comparison, we also perform classification using the following classifier: SVM, KNN, RF, DT and NB. The obtained results suggested that for binary classification, a model with a learning rate of 0.05, 30 neurons spread over 3 hidden layers got an accuracy of 99.69% on the KDDEvaluation set and 86.76% on the KDDTest+. Fig. 3 shows a comparison of this model with other classification algorithms. The Random Forest classifier with an accuracy of 85.18% for the KDDTest+ came into second position after the FFDNN model and the SVM classifier produced an accuracy of 84.41% on the same test set.

TABLE 4. Accuracy during training of FFDNN - binary classification.

No. of Nodes	Learning Rate	No. of Hidden layers	KDDEvaluation	KDDTest+
30	0.001	3	99.53%	82.69%
30	0.01	3	99.64%	85.22%
30	0.05	3	99.69%	86.76%
30	0.1	3	99.41%	84.66%
40	0.001	4	99.6%	85.17%
40	0.01	4	99.73%	86.47%
40	0.05	4	99.69%	84.43%
60	0.01	3	99.74%	84.01%
60	0.02	3	99.80%	83.75%
60	0.05	3	99.85%	83.71%
80	0.01	4	99.81%	84.42%
80	0.05	4	99.61%	82.88%
150	0.01	3	99.85%	83.29%
150	0.05	3	99.84%	83.42%

B. PHASE 2: MULTICLASS CLASSIFICATION WITH 41 FEATURES

We conducted multiclass classification in this phase by using five classes of the NSL-KDD dataset with all 41 features. As described in Table 5, the FFDNN model with 60 nodes spread through three hidden layers with a learning rate of 0.05 got an accuracy of 86.62% which is a much better performance compared to other FFDNN models settings. In order to put this experiment in perspective, we also

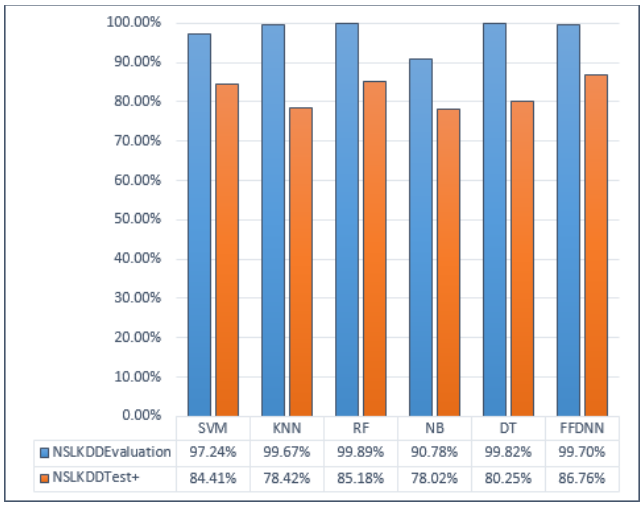


FIGURE 3. Binary classification accuracy comparison.

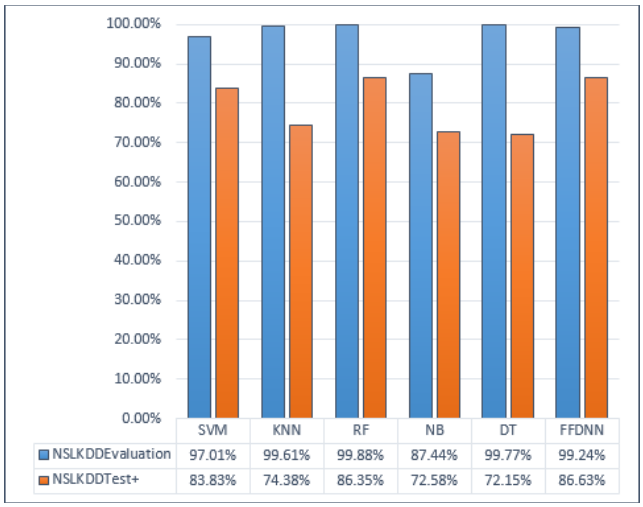


FIGURE 4. Multiclass classification accuracy comparison.

TABLE 5. Accuracy during training of FFDNN - multiclass classification.

No. of Nodes	Learning Rate	No. of Hidden layers	KDDEvaluation	KDDTest+
30	0.001	3	99.326	83.165
30	0.01	3	99.55%	82.43%
30	0.05	3	99.56%	82.00%
30	0.1	3	97.59%	82.85%
40	0.001	4	99.46%	84.07%
40	0.01	4	99.50%	84.34%
40	0.05	4	99.38%	83.52%
60	0.001	3	99.41%	83.97%
60	0.01	3	99.63%	83.43%
60	0.02	3	99.63%	84.70%
60	0.05	3	99.23%	86.62%
80	0.001	4	99.47%	84.32%
80	0.01	4	99.66%	85.73%
150	0.05	3	99.60%	84.49%

conducted a multiclass classification using SVM, KNN, RF, DT and NB classifiers. As depicted in Fig. 4, the comparison shows that FFDNN outperformed all other classifier on the test data; however, the RF classifier performed relatively well with an accuracy of 86.35% on test data and the SVM model got an accuracy of 83.83%.

C. PHASE 3: FEATURE EXTRACTION

We applied Algorithm 1 and Algorithm 2 in the FEU to the KDDTrain+ Full dataset in order to extract a reduced vector of features. The goal in this step was to select the features with enough information gain (IG) with respect to the class. The filtering process generated the features in Table 6 which represents 21 features.

In the next two phases, we repeat the experiments in Phase 1 and Phase 2; however, in these instances, a reduced feature vector F_{ranked} of 21 ranked features is used.

TABLE 6. Ranked Features.

No.	Feature Name	IG to Class
f5	src_bytes	0.57
f4	flag	0.47
f6	dst_bytes	0.44
f3	service	0.37
f30	diff_srv_rate	0.36
f29	same_srv_rate	0.36
f33	dst_host_srv_count	0.33
f34	dst_host_same_srv_rate	0.31
f38	dst_host_serror_rate	0.28
f39	dst_host_srv_serror_rate	0.28
f35	dst_host_diff_srv_rate	0.28
f12	logged_in	0.28
f23	count	0.27
f25	serror_rate	0.27
f26	srv_serror_rate	0.27
f32	dst_host_count	0.14
f36	dst_host_same_src_port_rate	0.13
f31	srv_diff_host_rate	0.097
f24	srv_count	0.065
f41	dst_host_srv_rerror_rate	0.064

D. PHASE 4: BINARY CLASSIFICATION WITH A REDUCED FEATURES VECTOR

Table 7 shows multiple FEU-FFDNN models that all have 21 inputs and two outputs. The best performing model with 30 hidden nodes, a hidden layer size of 3 and a learning rate of 0.05 got an accuracy of 99.37% over the evaluation set and 87.74% on the test data. This is an improvement over the best model using 41 inputs in Phase 1. Additionally, Fig. 5 shows an accuracy comparison between SVM, KNN, RF, DT, NB and FEU-FFDNN classifier for better contrasting. We noticed that the FEU-FFDNN outperformed other methods.

TABLE 7. Accuracy during training of FEU-FFDNN - Binary Classification.

No. of Nodes	Learning Rate	No. of Hidden layers	KDDEvaluation	KDDTest+
30	0.001	3	99.13%	87.29%
30	0.01	3	99.19%	87.18%
30	0.05	3	99.37%	87.74%
30	0.1	3	99.04%	85.81%
40	0.001	4	98.96%	85.52%
40	0.01	4	99.31%	84.48%
40	0.05	4	99.47%	83.90%
60	0.01	3	99.46%	83.60%
60	0.02	3	99.53%	86.38%
60	0.05	3	99.60%	87.26%
80	0.01	4	99.48%	85.05%
80	0.05	4	99.61%	86.31%
150	0.01	3	99.72%	87.10%

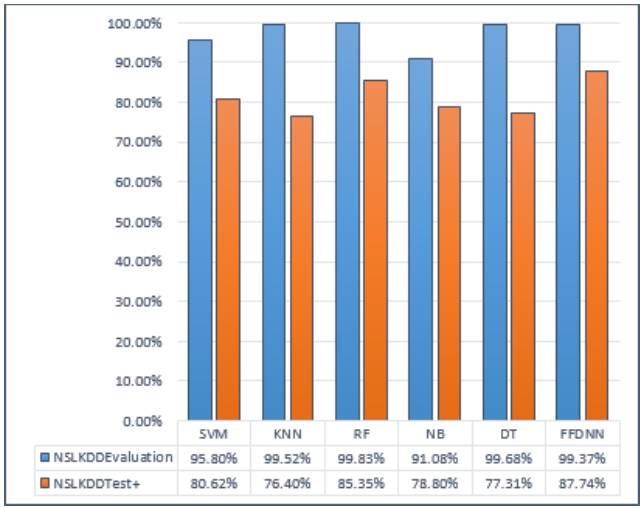


FIGURE 5. Binary classification accuracy comparison with reduced features set.

TABLE 8. Accuracy during training of FEU-FFDNN - multiclass classification.

No. of Nodes	Learning Rate	No. of Hidden layers	KDDEvaluation	KDDTest+
30	0.001	3	98.95%	83.37%
30	0.01	3	98.84%	82.83%
30	0.05	3	98.72%	84.19%
30	0.1	3	98.25%	81.72%
40	0.001	4	98.31%	85.27%
40	0.01	4	98.54%	85.39%
40	0.05	4	98.60%	79.05%
60	0.01	3	99.52%	83.36%
60	0.02	3	99.50%	85.12%
60	0.05	3	98.69%	84.78%
80	0.01	4	99.45%	84.06%
80	0.05	4	99.47%	85.83%
150	0.01	3	99.59%	85.64%
150	0.02	3	99.54%	86.19%

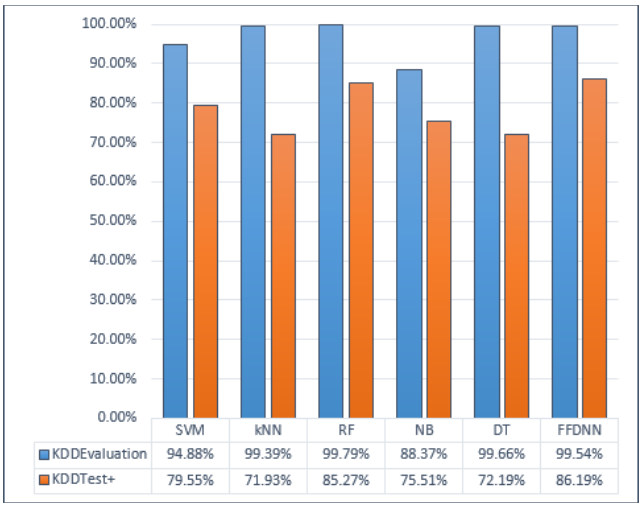


FIGURE 6. Multiclass classification accuracy comparison with reduced features set.

E. PHASE 5: MULTICLASS CLASSIFICATION WITH A REDUCED FEATURES VECTOR

In this stage of the experiments, we ran several FEU-FFDNN models and we used all classes groups present in the NSL-KDD dataset. The model that performed the best has 150 neurons, a learning rate of 0.05 and it yielded an accuracy of 99.54% on the validation data and 86.19% on the test data. In comparison to the results in Phase 2 of this research, this model needed more neurons as the feature vector dimension was reduced by the filtering process. Additionally, Fig. 6 shows a comparison of this model to existing ML models and the results showed that the FEU-FFDNN based model outperformed all other existing models.

Moreover, for the best performing model (150 neurons, three hidden layers, learning rate = 0.02), we plotted the precision and recall curve over the test dataset as seen in Fig. 7 where class 0 = 'normal', class 1 = 'R2L', class 2 = 'U2R', class 3 = 'Probe' and class 4 = 'DoS'. This curve gave us more details on how our model performed for different classes.

F. DISCUSSIONS

Our research explores in detail the application of FFDNNs to wireless intrusion detection using the NSL-KDD dataset. Experiments were carried out for both binary and multiclass classification. In the first two phases of the experimental process, the training and testing of the models were done using the entire feature vector. The results suggested that for both phases, FFDNNs outperformed other ML models. For binary classification, FFDDNNs required less neurons than for multiclass classification. In Phase 1, only 30 nodes spread

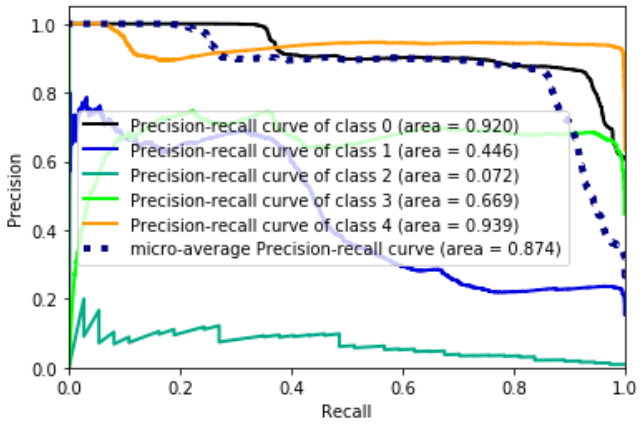


FIGURE 7. Precision-Recall curve.

over three hidden layers were needed for the generalization of our model; however, 60 neurons in three hidden layers were necessary for better approximation in the multiclass problem. Although the depth (number of hidden layers) was not affected, we can derive that the more attacks classes we have, the more neurons are needed to solve the complexity of the intrusion detection classification problem.

In Phase 3, a feature transformation and extraction procedure was executed based on IG and a feature vector with 21 ranked features was generated.

In Phase 4 and Phase 5, the experiment carried out in Phase 1 and Phase 2 were repeated respectively using a feature vector with a reduced dimension obtained from Phase 3. The results achieved in Phase 4 showed that with the same number of neurons as well as the same learning rate, the accuracy of the FEU-FFDNN model increased from 86.76% to 87.74% on the KDDTest+. For multiclass classification using the FEU in Phase 5, we obtained an overall accuracy of 86.19% with a depth of three hidden layers and 150 neurons. Here as well, the FEU-FDNN outperformed other methods as revealed in Fig.6. Moreover, we studied the intrinsic details of the classification in Phase 5 by plotting a Precision-Recall curve as depicted in Fig. 7. Based on the curve area values, Class 1 and class 2 were the classes with the most misclassifications instances because they do not appear often in both the training and test datasets.

Additionally, in comparison to other deep learning based methodologies using 41 features for multiclass classification such as stacked non-symmetric auto-encoders (S-NDAE) used in [19] that got 85.42% and recurrent neural networks (RNN) used in [40] that achieved an overall accuracy of 81.29%; the FFDNN in our research produced an accuracy of 86.62% on the test set, which is superior to the S-NDAE and RNN models.

VII. CONCLUSION

This paper presented the design, implementation and testing of a DL based intrusion detection system using FFDNNs. A literature review of ML and DL methods was conducted and it was found that the most efficient approach to intrusion detection has yet to be found. The FFDNN models used in this research were coupled to a FEU using IG in a bid to reduce the input dimension while increasing the accuracy of the classifier. The dataset used in this work is the NSL-KDD. For the binary and the multiclass classifications problems, the FFDNNs models both with a full and a FEU-reduced feature space achieved a performance that is superior to SVM, RF, NB, DT and KNN. In future work, we aim at finding a strategy to increase the detection rates of R2L and U2R attacks in the NSL-KDD dataset. Moreover, we will apply the FEU and the FFDDNs to the AWID dataset in order to investigate further the superiority of DL based methods for IDS over other ML approaches.

VIII. ACKNOWLEDGMENT

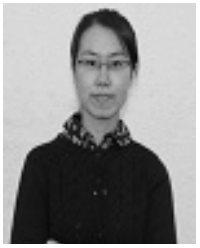
This research is partially supported by the South African National Research Foundation (Nos: 112108, 112142); SouthAfrican National Research Foundation Incentive Grant (No.95687); Eskom Tertiary Education Support Programme Grant; Research grant from URC of University of Johannes-Burg.

REFERENCES

[1] M. E. Aminanto, R. Choi, H. C. Tanuwidjaja, P. D. Yoo, and K. Kim, "Deep abstraction and weighted feature selection for Wi-Fi impersonation detection," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 3, pp. 621–636, Mar. 2018.
[2] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 184–208, 1st Quart., 2016.
[3] R. Mitchell and I.-R. Chen, "A survey of intrusion detection in wireless network applications," *Comput. Commun.*, vol. 42, no. 3, pp. 1–23, Apr. 2014. doi: 10.1016/j.comcom.2014.01.012.
[4] J. Hu, X. Yu, D. Qiu, and H. H. Chen, "A simple and efficient hidden Markov model scheme for host-based anomaly intrusion detection," *IEEE Netw.*, vol. 23, no. 1, pp. 42–47, Jan. 2009.
[5] D. A. Effendy, K. Kusrini, and S. Sudarmawan, "Classification of intrusion detection system (IDS) based on computer network," in *Proc. Int. Conf. Inf. Tech, Inf. Sys. Elec. Eng.*, Nov. 2017, pp90-94.
[6] E. Viegas, A. O. Santin, A. França, R. Jasinski, V. A. Pedroni, and L. S. Oliveira, "Towards an energy-efficient anomaly-based intrusion detection engine for embedded systems," *IEEE Trans. Comput.*, vol. 66, no. 1, pp. 163–177, Jan. 2017.
[7] S. M. H. Bamakan, B. Amiri, M. Mirzabagheri, and Y. Shi, "A new intrusion detection approach using PSO based multiple criteria linear programming," *Procedia Comput. Sci.*, vol. 55, pp. 231–237, Aug. 2015.
[8] P. Louridas and C. Ebert, "Machine learning," *IEEE Softw.*, vol. 33, no. 5, pp. 110–115, May 2016.
[9] Y. Xin *et al.*, "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018.
[10] Y. Y. Aung and M. M. Min, "Hybrid intrusion detection system using K-means and K-nearest neighbors algorithms," in *Proc. IEEE/ACIS 17th Int. Conf. Comput. Inf. Sc.*, Jun. 2018, pp. 34–38.
[11] P. Arumugam and P. Jose, "Efficient decision tree based data selection and support vector machine classification," *Mater. Today Proc.*, vol. 5, no. 1, pp. 1679–1685, 2018.
[12] A. Dastanpour, S. Ibrahim, R. Mashinchi, and A. Selamat, "Comparison of genetic algorithm optimization on artificial neural network and support vector machine in intrusion detection system," in *Proc. IEEE Conf. Open Syst. (ICOS)*, Oct. 2014, pp. 72–77.
[13] N. Farnaaz and M. A. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Comput. Sci.*, vol. 89, pp. 213–217, May 2016.
[14] F. Tian, X. Cheng, G. Meng, and Y. Xu, "Research on flight phase division based on decision tree classifier," in *Proc. Int. Conf. Comput. Intell. Appl.(ICCIA)*, Sep. 2017, pp. 372–375.

[15] A. Shenfield, D. Day, and A. Ayesh, "Intelligent intrusion detection systems using artificial neural networks," *ICT Express*, vol. 4, no. 2, pp. 95–99, Jun. 2018.
[16] L. van Efferen and A. M. Ali-Eldin, "A multi-layer perceptron approach for flow-based anomaly detection," in *Proc. Int. Symp. Netw., Comput. Commun. ISNCC*, May 2017, pp. 1–6.
[17] Z. Chiba, N. Abghour, K. Moussaid, A. El Omri, and M. Rida, "A novel architecture combined with optimal parameters for back propagation neural networks applied to anomaly network intrusion detection," *Comput. Secur.*, vol. 75, pp. 36–58, Jun. 2018.
[18] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[19] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.
[20] I. Lopez-Moreno, J. Gonzalez-Dominguez, D. Martinez, O. Plchot, and P. J. Moreno, "On the use of deep feedforward neural networks for automatic language identification," *Comput. Speech Lang.*, vol. 40, pp. 46–59, Nov. 2016.
[21] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1026–1034.
[22] S. Agatonovic-Kustrin and R. Beresford, "Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research," *J. Pharmaceutical Biomed. Anal.*, vol. 22, no. 5, pp. 717–727, 2000.
[23] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 4, pp. 491–502, Apr. 2005.
[24] S. S. Kannan and N. Ramaraj, "A novel hybrid feature selection via symmetrical uncertainty ranking based local memetic search algorithm," *Knowl.-Based Syst.*, vol. 23, no. 6, pp. 580–585, Aug. 2010.
[25] A. Taherkhani, G. Cosma, and T. M. McGinnity, "Deep-FS: A feature selection algorithm for deep boltzmann machines," *Neurocomputing*, vol. 322, pp. 22–37, Dec. 2018.
[26] M. Labani, P. Moradi, M. Jalili, and X. Yu, "An evolutionary based multi-objective filter approach for feature selection," in *Proc. World Congr. Comput. Commun. Tech. (WCCCT)*, Feb. 2017, pp. 1510–154.
[27] P. S. Tang, X. L. Tang, Z. Y. Tao, and J. P. Li, "Research on feature selection algorithm based on mutual information and genetic algorithm," in *Proc. 11th Int. Comput. Conf. Wavelet Active Media Tech. Inf. Process. (ICCWAMTIP)*, Dec. 2014, pp. 403–406.
[28] C. Liu, W. Wang, Q. Zhao, X. Shen, and M. Konan, "A new feature selection method based on a validity index of feature subset," *Pattern Recognit. Lett.*, vol. 92, pp. 1–8, Jun. 2017.
[29] R. Chakraborty and N. R. Pal, "Feature selection using a neural framework with controlled redundancy," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 1, pp. 35–50, Jan. 2015.
[30] L. Vanneschi and M. Castelli, "Multilayer perceptrons," *Encyclopedia Bioinf. Comput. Biol.*, vol. 1, pp. 612–620, Jun. 2019.
[31] F. Murtagh, "Multilayer perceptrons for classification and regression," *Neurocomputing*, vol. 2, nos. 5–6, pp. 183–197, Jul. 1991.
[32] J. George and S. G. Raj , "Leaf recognition using multi-layer perceptron," in *Proc. Int. Conf. Energy Commun. Data Analytics Soft Comput. (ICECDS)*, Aug. 2017, pp. 2216–2221.
[33] H. Amakdouf, M. E. Mallahi, A. Zouhri, A. Tahiri, and H. Qjidaa, "Classification and recognition of 3D image of charlier moments using a multilayer perceptron architecture," *Procedia Comput. Sci.*, vol. 127, pp. 226–235, Aug. 2018.
[34] A. Mondal, A. Ghosh, and S. Ghosh, "Scaled and oriented object tracking using ensemble of multilayer perceptrons," *Appl. Soft Comput.*, vol. 73, pp. 1081–1094, Dec. 2018.
[35] I. H. Witten, M. A. Hall, E. Frank, and C. J. Pal, "The WEKA workbench," in *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed. Burlington, MA, USA: Appendix, 2017, pp. 553–571.
[36] T. Mehmod and H. B. M. Rais, "Ant colony optimization and feature selection for intrusion detection," in *Advances in Machine Learning and Signal Processing*, vol. 387. New York, NY, USA: Springer, 2016, pp. 305–312.
[37] C. Khammassi and S. Krichen, "A GA-LR wrapper approach for feature selection in network intrusion detection," *Comput. Secur.*, vol. 70, pp. 255–277, Sep. 2017.
[38] J. McCall, "Genetic algorithms for modelling and optimisation," *J. Comput. Appl. Math.*, vol. 184, no. 1, pp. 205–222, Dec. 2005.
[39] H. Wang, J. Gu, and S. Wang, "An effective intrusion detection framework based on SVM with feature augmentation," *Knowl.-Based Syst.*, vol. 136, pp. 130–139, Nov. 2017.
[40] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
[41] V. L. Thing, "IEEE 802.11 network anomaly detection and attack classification: A deep learning approach," in *Proc. Wireless Commun. Netw. Conf. (WCNC)*, May 2017, pp. 1–6.
[42] S. Ding and G. Wang, "Research on intrusion detection technology based on deep learning," in *Proc. Int. Conf. Comput. Commun. (ICCC)*, Dec. 2017, pp. 1474–1478.
[43] M. Agarwal, D. Pasumarthi, S. Biswas, and S. Nandi, "Machine learning approach for detection of flooding DoS attacks in 802.11 networks and attacker localization," *Int. J. Mach. Learn. Cybern.*, vol. 7, no. 6, pp. 1035–1051, Dec. 2016.
[44] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Raheem, "Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection," *IEEE Access*, vol. 6, pp. 33789–33795, 2018.
[45] B. Trstenjak, S. Mikac, and D. Donko, "KNN with TF-IDF based framework for text categorization," *Procedia Eng.*, vol. 69, pp. 1356–1364, May 2014.
[46] S. Tan, "An effective refinement strategy for KNN text classifier," *Expert Syst. Appl.*, vol. 3, no. 2, pp. 290–298, 2006.
[47] M. O. Mughal and S. Kim, "Signal classification and jamming detection in wide-band radios using Naïve bayes classifier," *IEEE Commun. Lett.*, vol. 22, no. 7, pp. 1398–1401, Jul. 2018.
[48] W. Mo, C. L. Gutterman, Y. Li, S. Zhu, G. Zussman, and D. C. Kilper, "Deep-neural-network-based wavelength selection and switching in ROADM systems," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 10, no. 10, pp. D1–D11, 2018.
[49] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," in *Proc. Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2018, pp. 178–183.
[50] R. Garreta and G. Moncecchi, *Learning Scikit-Learn: Machine Learning in Python*. Birmingham, U.K.:Packt Publishing Ltd, 2013.
[51] Z. Gao, Y. Xu, F. Meng, F. Qi, and Z. Lin, "Improved information gain-based feature selection for text categorization," in *Proc. Int. Conf. Wireless Commun. Veh. Technol. Inf. Theory Aerosp. Electron. Sys. (VITAE)*, Aug. 2014, pp. 1–5.
[52] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 5, no. 1, pp. 3–55, 2001.
[53] H. Zhou, Z. Deng, Y. Xia, and M. Fu, "A new sampling method in particle filter based on Pearson correlation coefficient," *Neurocomputing*, vol. 216, pp. 208–215, May 2016.

SYDNEY MAMBWE KASONGO received the master's (M.Tech.) degree in computer systems from the Tshwane University of Technology, in 2017. He is currently pursuing the Ph.D. degree in electrical and electronic engineering with the University of Johannesburg. His current research interests include machine learning, deep learning, computer networks security, wireless networks, and data science.



YANXIA SUN received the joint D.Tech. degree in electrical engineering from the Tshwane University of Technology, South Africa, and the Ph.D. degree in computer science from University Paris-EST, France, in 2012. She is currently an Associate Professor or the Head of the Department of Electrical and Electronic Engineering Science, University of Johannesburg, South Africa. She has 15 years teaching and research experience. She has lectured five courses in the universities. She has supervised or co-supervised five postgraduate projects to completion. She is currently supervising four master's students and six Ph.D. students. She published 42 papers including 14 ISI master indexed journal papers. She is the Investigator or Co-Investigator for six research projects. She is the member of the South African Young Academy of Science. Her research interests include renewable energy, evolutionary optimization, neural networks, nonlinear dynamics, and control systems.