A Deep Learning Method With Wrapper Based Feature Extraction For Wireless Intrusion Detection System

Sydney Mambwe Kasongo, Yanxia Sun

PII: S0167-4048(20)30036-5
DOI: https://doi.org/10.1016/j.cose.2020.101752
Reference: COSE 101752

To appear in: *Computers & Security*

Received date: 21 August 2019
Revised date: 10 January 2020
Accepted date: 6 February 2020

Please cite this article as: Sydney Mambwe Kasongo, Yanxia Sun, A Deep Learning Method With Wrapper Based Feature Extraction For Wireless Intrusion Detection System, *Computers & Security* (2020), doi: https://doi.org/10.1016/j.cose.2020.101752

A Deep Learning Method With Wrapper Based Feature Extraction For Wireless Intrusion Detection System

Sydney Mambwe Kasongo, Yanxia Sun*

Department of Electrical & Electronic Engineering Science, University of Johannesburg, South Africa

Abstract

In the past decade, wired and wireless computer networks have substantially evolved because of the rapid development of technologies such as the Internet of Things (IoT), wireless handled devices, vehicular networks, 4G and 5G, cyber-physical systems, etc. These technologies exchange large amount of data, and as a result, they are prone to several malicious actions, attacks and security threats that can compromise the availability and integrity of information or services. Therefore, the security and protection of the various communication infrastructures using an intrusion detection system (IDS) is of critical importance. In this research, we propose a Feed-Forward Deep Neural Network (FFDNN) wireless IDS system using a Wrapper Based Feature Extraction Unit (WFEU). The extraction method of the WFEU uses the Extra Trees algorithm in order to generate a reduced optimal feature vector. The effectiveness and efficiency of the WFEU-FFDNN is studied based on the UNSW-NB15 and the AWID intrusion detection datasets. Furthermore, the WFEU-FFDNN is compared to standard machine learning (ML) algorithms that include Random Forest (RF), Support Vector Machine (SVM), Naïve Bayes (NB), Decision Tree (DT) and k-Nearest Neighbor (kNN). The experimental studies include binary and multiclass types of attacks. The results suggested that the proposed WFEU-FFDNN has greater detection accuracy than other approaches. In the instance of the UNSW-NB15, the WFEU generated an optimal feature vector consisting of 22 attributes. Using this input vector; our approach achieved overall accuracies of 87.10% and 77.16% for the binary and multiclass classification schemes, respectively. In the instance of the AWID, a reduced input vector of 26 attributes was generated by the WFEU, and the experiments demonstrated that our method obtained overall accuracies of 99.66% and 99.77% for the binary and the multiclass classification configurations, respectively.

Keywords: Machine Learning, Deep Learning, Intrusion Detection, Wireless Networks, Feature Extraction

1. Introduction

The emergence of wireless communication technologies, wireless handheld devices, wireless computer networks (WCN) has allowed for processing large volumes of data at a given moment. As a result, these systems are subjected to a myriad of malicious attacks and security vulnerabilities. It is therefore imperative to develop wireless intrusion detection systems (WIDS) that are proactive, efficient and accurate in mitigating these attacks [1]. An Intrusion Detection System (IDS) is the first line of defense in a computer network. At the highest level, IDSs are categorized as: Host-based IDS (H-IDS) and Network-based IDS (N-IDS). The N-IDS operates in a distributed manner within a network system, and the H-IDS runs on a host computer or system. [2]. Additionally, both types of IDSs operate using two methods, namely, anomaly-based detection and signature-based recognition [3].

Traditional IDSs suffer from shortcomings such as a high false alarm rate, a reduced efficiency in detecting novel forms of attacks and a low detection accuracy [3, 4]. It is therefore imperative to design robust IDSs that can improve the detection accuracy, decrease the false alarm rate and increase the efficiency in the discovery of new types of attacks. To fulfill network security requirements, Machine Learning (ML) methods have been heavily investigated in order to devise IDSs capable of performing optimally. ML is an evolving field of computational algorithms that emulate human intelligence. ML methods use statistical models to dis-

*Corresponding author:
Email address: ysun@uj.ac.za (Yanxia Sun)

cover patterns in large volumes of data [5].

In the context of IDS research and development, the following ML techniques have been extensively used: k-Nearest Neighbors (kNN) [6], Support Vector Machine (SVM) [7], Decision Tree (DT) [8], Artificial Neural Networks (ANNs) [9, 10], Random Forest (RF) [11] and Naive Bayes (NB) [12]. Our research focuses on Deep Learning (DL). DL is a branch of ML that has to do with algorithms inspired by the structure, the function and the operation of the biological neurons within the human brain [10, 13]. DL has shown a lot of success in various fields such as natural language processing [14], medical research [15], speech recognition [16], image recognition [17], aerospace and defense [18], etc.

In comparison to traditional ML algorithms, DL based algorithms deal with large datasets that often have a myriad of features (inputs). Some of the features are relevant and required for solving a specific classification problem and others are not required and are redundant. Furthermore, datasets that have a high feature vector dimension tend to be complex to train and test. Therefore, for an increased performance, it is imperative to perform Feature Engineering (FE) over a particular multidimensional dataset. FE has gained a great deal of momentum in ML and DL research [19, 20, 21]. FE is defined as the process of combining or extracting inputs in order to increase the performance of particular classifier. Our research focuses on the feature extraction process. At the highest level, feature extraction methods are categorized as wrapper-based and filter-based [22].

The wrapper-based feature extraction approach is used to compute the inputs weights or importance by using a classification model to measure the performance of those features [23]. In the wrapper-based methodology, the input subset extraction algorithm acts as a wrapper layer around the classification method. Further, the input subset extraction algorithm carries out a search in order to select the best (optimal) feature subset that is ultimately used in the final classification process. This search is based on a desired classification metric. In this way, the classification (induction) algorithm is embedded in the feature selection procedure [24].

The filter-based approach uses the property of the data in order to select the most appropriate inputs without the intervention of a classifier model. In other words, the filter-based method picks the input subsets by using a pre-processing step. The drawback with this algorithm is the fact that it does not consider the impact of the selected input subset over the chosen classification algorithm [25].

In this paper, we focus on the wrapper-based approach. We have selected the Extra Trees (ET) algorithm in order to conduct the feature extraction process. The advantage of using the ET resides in its versatility whereby it is capable of generating the feature subsets using different conditions (in terms of the target class). Moreover, the input subsets obtained from the ET extraction process can be used for classification using a different estimator [26]. Further, the ET is a tree based algorithm and more details are provided in section 4.3.1.

The methodology of this paper is an addendum to the work we conducted in [27]. The key differences reside in the technique used for the feature engineering process and the dataset used to assess the performance of the various ML and DL models. In [27], the experiments were executed on the NSL-KDD intrusion dataset. Moreover, the feature extraction unit uses a two stage normalization procedure in conjunction with a filtering algorithm that is fully independent from any classifier so as to pick the required inputs needed for the classification process. However, in the current work, the datasets used during the experiments are the UNSW-NB15 and the AWID. Furthermore, the feature extraction method, in this instance, is a wrapper approach whereby the feature ranking and extraction process is dependent on a classifier algorithm. The benefit of using the strategy proposed in this research is the fact that we are able to use two separate classifiers at different levels. Firstly, we extract a richer representation of the feature vector using one classifier. This allows for an optimal model tuning. Secondly, we perform the intrusion detection using a more potent ML or DL classifier.

The main contributions of this research are:

- A wrapper-based feature extraction unit (WFEU) is developed. This method is built using the Extra-Trees classifier as the foundation.
- The WFEU is applied to the UNSW-NB15 and the AWID datasets in order to compute the respective features importance measures with respect to the target class.
- We apply the full set of features of the UNSW-NB15 dataset, to the following traditional ML classification algorithms: SVM, DT, RF, NB and kNN. We further apply full set of features of the UNSW-NB15 to Feed Forward Deep Neural Networks (FFDNNs) and we compare the results.
- After the application of the WFEU to the UNSW-NB15 dataset, a candidate reduced feature vector was generated. The attack classification process was done using the SVM, FFDNNs, DT, RF, NB

and kNN. The outcomes demonstrate that a DL approach yields improved results for both the binary and the multiclass attacks detection tasks.

- We implement the approach presented in [27] and we compare the results that are obtained using the methods proposed in this paper.
- We implement the binary and the multiclass classification configurations using the full set of features of the AWID.
- Using the WFEU on the AWID, we generated an optimal feature vector consisting of 26 attributes. This input array was used in the evaluation of FFDNNs. The results showed that our proposed method achieved an outstanding performance in comparison to the existing works.

The rest of this paper is structured as follows: Section 2 provides an overview of existing ML methods and Section 3 provides a review of related work. Section 4 presents the proposed approach for wireless intrusion detection. In Section 5, we conduct the experiments and hold the discussions, and in Section 6 we conclude the paper.

2. A review of commonly used ML Classifiers

2.1. Naive Bayes

NB algorithms constitute a group of supervised learning methods built on the Bayes' theorem [28]. For a particular data entity, the NB approach assumes a certain level of autonomous *"naivety"* amongst the features. Given $V = (v_1, ..., v_n)$, an input vector with n inputs destined for classification. In order to predict the class C_k for an attribute in V, the algorithm operates as follow:

$$p(C_k|V) = \frac{p(V|C_k)p(C_k)}{P(V)} \tag{1}$$

And the category (class) of v_n is computed using Equation (2) as follows:

$$\Gamma = \underset{k\in\{1,...,K\}}{\arg\max}\, p(C_k) \prod_{i=1}^{n} p(v_i|C_k) \tag{2}$$

where Γ represents the predicted label. In this research, the experiments are based on the Gaussian Naive Bayes (GaussianNB) algorithm where by the likelihood of inputs is assumed to be of a Gaussian nature as follows [29]:

$$p(v_n|C_k) = \frac{1}{\sqrt{2\pi\sigma^2_{C_k}}} exp\left(- \frac{(v_n - \mu_{C_k})^2}{2\sigma^2_{C_k}} \right) \tag{3}$$

where the estimation of σ_{C_k} and μ_{C_k} is done using the maximum likelihood.

2.2. K-Nearest Neighbor

The kNN method is a classification algorithm used to categorize various types of data. This approach is inspired from the Standard Euclidean Distance (SED) [30] that exists between two points in the same space. Given m and z, two points within a space S, the measure of the distance separating v and z, $\delta(m, z)$, is calculated using Equation (4).

$$\delta(m, z) = \sqrt{\sum_{t=1}^{n} (m_k - z_k)^2} \tag{4}$$

n is the maximum number of attributes in S. The kNN algorithm estimates the class of an instance m_0 in S by calculating the label of m_0 is dictated by the class of k related neighbors [31].

2.3. Support Vector Machine

One of the most common and simple ML algorithms used to categorize different types of data is SVM. It is a non-probabilistic method and it can solve the complexity of noisy datasets. An SVM creates a hyper-plane or a multiple hyper-planes in a boundless dimensional input vector in order to categorize its instances. The SVM that ideally divides the instances according to their label is the most efficient one [32].

2.4. Decision Tree and Random Forest

Another supervised ML algorithm that is commonly used to classify data is the Decision Tree (DT). For a training set composed of labeled objects, the DT approach creates a computing model that resembles a tree. This model is able to predict the label of a given sample of instances [8, 28]. A DT has a simple structure which comprises the category nodes, the internal nodes and a root node. In order to label the input features of a dataset, a DT computation process occurs from the root node to a category node passing through several internal nodes depending on the dimension of the dataset. The best decision is attained when a leaf node is correctly labeled.

DTs are the building blocks of the Random Forest (RF) procedure. The RF algorithm is a meta estimator that uses several DTs on a dataset in order to predict the class of an instance [11].

3. Related Work

In previous studies, various methods based on both ML and DL were explored and applied to the wireless intrusion detection problem. This section provides a survey of existing ML and DL based solutions for IDSs.

The authors in [27] proposed a methodology for wireless networks attacks detection using DL and a filter based method for the features space reduction. The features selection algorithm used in this work uses the information gain (IG) theory as its foundation [33]. A comparison study was conducted between the feed forward deep neural networks IDS (FFDNN-IDS) and the following algorithms: DT, SVM, kNN, RF and NB. The experiments were conducted for the multiclass and the binary types of attacks using the NSL-KDD intrusion detection dataset. The results suggested that the IG-FFDNN-IDS attained an increased performance in comparison to the other models that were also implemented. Based on the reduced (filtered) NSL-KDD set of features, the FFDNN-IDS achieved an accuracy of 99.37% on the training data and 87.74% on the test data for binary classification. For the multiclass classification configuration, the FFDNN-IDS yielded an accuracy of 94.54% on the training set and 86.19% over the testing set.

In [34], an application of Deep Neural Networks (DNN) for intelligent intrusion detection was presented. In this instance, the authors did not use any feature extraction methods. Several experiments were carried out and the DNN model was compared to classical ML methods. The experiments were conducted on publicly available datasets including the KDD Cup 99, NSL-KDD, WSN-DS and UNSW-NB15. The results suggested that the DNN surpassed other models for the multiclass and the binary classification setups. Using the KDDCup 99, a DNN with 5 layers achieved an accuracy of 92.7% with regards to the binary categorization and 92.5% for the multiclass setup. On the NSL-KDD, a DNN with 5 layers yielded a detection accuracy of 78.9% for the binary categorization setup and 78.5% for the multiclass classification. On the WSN-DS, a DNN with 5 layers got an accuracy of 98.2% for binary classification and 96.4% for multiclass classification. On the UNSW-NB15, a DNN with 5 layers got a detection accuracy of 76.1% for the binary classification scheme and 65.1% for the multiclass setting.

In [35], the authors presented an IDS based on the ANN classifier with a reduced set of features that was obtained using the correlations. These correlations are used in order to rank the features with respect to the class and determine whether or not a feature is impactful during the classification process. After the training and testing processes over the KDD Cup 99, the outputs suggested that the proposed method showed an increased model's accuracy and a decrease in the false alarm rate.

In [36], a network intrusion detection based on SVM and RF was presented. RFs were applied for the features selection process based on a variable importance score algorithm. The dataset used to evaluate the above approach is the KDD Cup 99. Using SVM as the classifier, the performance results suggested that a reduced set of features including 14 attributes achieved an accuracy of 93% on the training data versus 90% on the full set of 41 features.

In [37], the authors investigated the application DNNs for anomaly detection. Deep Convolutional Neural networks (DCNN), various Autoencoders and Long-Short Term Memory (LSTM) Recurrent Neural Network (RNN) were implemented. These DNN models were compared to classical ML such as Extreme Learning Machine (ELM), kNN, DT, NB, SVM and RF. The dataset used to train and test the above models is the NSL-KDD. The results confirmed that the DNN based IDSs outperformed standard ML based IDSs. The experiments showed, for instance, that DCNN achieved an accuracy of 85% on the test data which was a superior performance in comparison to the standard ML methods. In conclusion, the authors deducted that DL is a promising technology for information security (InfoSec) applications.

In [38], the authors proposed an evolutionary feed forward neural networks based IDS using the locust swarm optimization (FFNN-LSO) technique. In a bid to derive the performance of the FFNN-LSO, both the NSL-KDD and UNSW-NB15 entities were utilized. Moreover, the FFNN-LSO was compared to traditional evolutionary algorithms such as the FFNN coupled with the Genetic Algorithm (GA) and the FFNN using the Particle Swarm Optimizer (PSO). The experimental results on the UNSW-NB15 showed that FFNN-LSO had a detection accuracy (DA) of 95.24% and a false alarm rate (FAR) of 9.40%. The FFNN-GA achieved a DA of 86.44% and a FAR of 29.1%. The FFNN-PSO got a DA of 94.42% and a FAR of 14.78%. On the NSL-KDD, FFNN-LSO yielded a DA of 94.04% and a FAR of 2.21%. The FFNN-GA achieved a DA of 87.69% and a FAR of 1.16%. The FFNN-PSO got a DA of 89.09% and a FAR of 5.62%. Based on the above results, the FFNN-LSO outperformed the FFNN-PSO and the FFNN-GA.

In [39], an IDS based on the LSTM classifier was proposed. The authors used the CIDDS dataset in a bid

to assess the performance of their methodology. The detection accuracy was the main metric used to determine the efficiency of the LSTM IDS. Additionally, the model was compared to various other approaches such as SVM, NB and the Multilayer Perceptron (MLP). The results demonstrated that the LSTM IDS outperformed its peers with a detection accuracy of 84.83% on the validation data.

In [40], a bidirectional LSTM based approach was used to classify attacks within the UNSW-NB15 dataset. An LSTM is a special type of Recurrent Neural Networks. The LSTM IDS showed that it was capable of detecting malicious activities effectively; however, its performance suffered with the class imbalance issue found in the UNSW-NB15 dataset. The authors concluded that the class imbalance problem was the next phase to be resolved in their research.

The work in [41] presented a wrapper-based feature extraction approach based on a GA which is an evolutionary technique. Additionally, the Logistic Regression (LR) ML methodology is used in order to select the best subset of features. Their experiments were conducted on the KDD Cup99 dataset as well as the UNSW-NB15 dataset using three DT classifiers in order to perform the network intrusion detection. For the KDD Cup99 [42, 43], the best feature vector had 18 attributes and yielded an accuracy of 99.90%, a detection rate of 99.81% and a FAR of 0.105%. In the instance of the UNSW-NB15 dataset, the best feature vector included 20 attributes and achieved an accuracy of 81.42% and FAR of 6.39%.

The researchers in [44] implemented an IDS based on the Incremental Extreme Learning Machine (I-ELM) coupled with an Adaptive Principal Component Analysis (APCA). An ELM is a form of DNN that contains multiple hidden layers consisting of a number of neurons. ELMs are generally built using a feed-forward configuration. In this work, the APCA is used as a wrapper-based feature selection technique whereby the best subsets of features are progressively learned and extracted before being assessed by the I-ELM engine. The researchers conducted their experiments using the NSL-KDD and the UNSW-NB15 datasets. The results showed that the APCA-I-ELM IDS achieved an overall test accuracy of 81.22% over the NSL-KDD dataset. In the instance of the UNSW-NB15, their system obtained an overall accuracy 70.51%.

In [45], the authors performed an extensive performance investigation on the KDDCup 99 as well as the UNSW-NB15 datasets using five different classifiers. These methods include: NB, DT, Artificial Neural Networks (ANNs), LR and EM Clustering. The performance metrics used were the accuracy and the FAR. Focusing on ANNs, the experiments showed that the overall accuracy of detection and the FAR were 97.04% and 1.48% on the KDD Cup 99 dataset. In the case of the UNSW-NB15, ANNs yielded an accuracy of 81.34% and a FAR of 21.13%. Based on the ANNs results as well as those of the four remaining classifiers, the authors concluded that the UNSW-NB15 is a much more complex dataset in comparison to the KDD Cup 99 dataset. This complexity reflects that of the real world wireless networks traffic.

The IDS approach proposed in [46] is based on Deep Feed-Forward Neural Networks(DFFNNs) and Deep Auto-Encoders (DEAs). The datasets used in the experiments were the NSL-KDD and the UNSW-NB15. The DEAs were used to extract a deeper representation of the data. These representations are then fed to the DFFNNs for the classification process. The presented DAE-DFFNN method showed that it can successfully construct and extract crucial features which improve the model's effectiveness. The results suggested that DAE-DFFNN got an overall accuracy of 98.6% on the NSL-KDD dataset and 92.4% on the UNSW-NB15 dataset.

In [47], the researchers implemented a WIDS based on a semi-supervised DL method using the AWID dataset. The technique presented in this work employs a ladder network approach in order to select the optimal inputs that were required to detect attacks effectively. The architecture of the ladder network consisted of a Stacked Auto-Encoder (SAE) in conjunction with a clean decoder unit, a noisy encoder unit and a decoder unit. This wrapper based architecture was capable of self-learning the features and it was able to pick the attributes that were optimal for an increased classification accuracy. Moreover, a focal loss function was employed in the aim of improving the generalization aptitude of the proposed framework. The simulations were carried out for the multiclass scheme. The results demonstrated that this framework achieved 89.32%, 73.41%, 82.79% and 99.77% the following classes, respectively: impersonation, flooding, injection and normal. The overall accuracy was 98.54%.

The work conducted in [48] introduced a DL based approach for WIDS. In this research, the authors used a fully unsupervised DL method based on the k-means clustering and SAE. The SAE is used as a wrapper-based feature extraction method in order to extract the necessary features that will be used by the clustering algorithm in the aim to generate different clusters consisting of each type of network traffic (malicious and legitimate). The following performance metrics were employed with the goal to assess the performance of

the proposed system: the detection rate, the FAR, the accuracy, the precision as well as the F1 score. The framework implemented here achieved, 92.18%, 4.40%, 94.81%, 86.15% and 89.06% for each performance metric, respectively.

In cite[49], the authors implemented a WIDS using the AWID using an Ant Colony Optimization (ACO) based algorithm in order to extract classification rules. In their experiments, they implemented a filter-based feature extraction methodology using a correlation-based algorithm. This procedure resulted in the full feature space of the AWID being reduced to an array of 35 attributes. The results of their simulations showed that the RF classifier obtained overall accuracies of 98.87% and 99.10% for the multiclass and binary classification schemes, respectively.

Table 1 provides a summary of the most important methods discussed in this paper. In this table, we list the classification methods as well as their year of publication. We also indicate the dataset that was used, the feature selection approach that was used as well as the overall accuracy on the test data. In the "Feature Selection Method" column, we use the keyword *None* to specify that no feature extraction method was used.

Table 1
Summary of Discussed Methods

Method Year	Dataset	Feature Selection Method	Overall Accuracy Multiclass
FFDNN [25] 2019	NSL-KDD	Filter	81.19%
DNN [26] 2019	NSL-KDD	None	78.50%
DNN [26] 2019	UNSW-NB15	None	65.10%
DFFN [35] 2018	NSL-KDD	Wrapper	98.60%
DEA [35] 2018	UNSW-NB15	Wrapper	92.40%
DCNN [29] 2018	NSL-KDD	None	85.00%
DT-GA-LR [33] 2017	UNSW-NB15	Wrapper	81.42%
ANN [34] 2016	UNSW-NB15	Filter	81.34%
APCA I-ELM [56] 2019	NSL-KDD	Wrapper	81.22%
APCA I-ELM [56] 2019	UNSW-NB15	Wrapper	70.51%
LSTM [31] 2018	CIDDS	None	84.83%
SVM [31] 2017	KDD Cup 99	Wrapper	93.00%
SAE [52] 2018	AWID	Wrapper	98.57%
SAE [53] 2017	AWID	Wrapper	94.81%
ACO [54] 2017	AWID	Filter	98.87%

4. The Proposed Approach for Wireless intrusion Detection

4.1. UNSW-NB15 Dataset

The clean UNSW-NB15 dataset [50] used in this research has 42 features (inputs) as described in Table 2. In this dataset, three inputs are nominal features and 39 are numeric features with the following data types: *integer*, *float* and *binary*. The UNSW-NB15 includes two sets, namely, a training set and a testing set. In this research, we further split the training set in two subsets: the UNSW-NB15-25 which represents 25% of the full training set and the UNSW-NB15-75 which is 75% of the full training set. The UNSW-NB15-25 will serve as a validation set after training a model on the UNSW-NB15-75. This strategy prohibits a model to train through the validation or the test sets. The strategy also guarantees that the results obtained on the validation and the test data are independent from any bias and interference during the model's learning phase.

The UNSW-NB15 includes the following nine types of intrusions: Shellcode, Generic, Worms, Fuzzers, DoS, Reconnaissance, Analysis, Backdoor and Exploits. The values distribution of the UNSW-NB15-75, UNSW-NB15-25 and UNSW-NB15-TEST are depicted in Table 3.

Table 2
UNSW-NB15 Features List

No.	Name	Category	No.	Name	Category
f1	dur	float	f22	dtcpb	integer
f2	proto	categorical	f23	dwin	integer
f3	service	categorical	f24	tcprtt	float
f4	state	categorical	f25	synack	float
f5	spkts	integer	f26	ackdat	float
f6	dpkts	integer	f27	smean	integer
f7	sbytes	integer	f28	dmean	integer
f8	dbytes	integer	f29	trans_depth	integer
f9	rate	float	f30	response_body_len	integer
f10	sttl	integer	f31	ct_srv_src	integer
f11	dttl	integer	f32	ct_state_ttl	integer
f12	sload	float	f33	ct_dst_ltm	integer
f13	dload	float	f34	ct_src_dport_ltm	integer
f14	sloss	integer	f35	ct_dst_sport_ltm	integer
f15	dloss	integer	f36	ct_dst_src_ltm	integer
f16	sinpkt	float	f37	is_ftp_login	binary
f17	dinpkt	float	f38	ct_ftp_cmd	integer
f18	sjit	float	f39	ct_flw_http_mthd	integer
f19	djit	float	f40	ct_src_ltm	integer
f20	swin	integer	f41	ct_srv_dst	integer
f21	stcpb	integer	f42	is_sm_ips_ports	binary

Table 3
UNSW-NB15 values distribution

Attack Type	UNSW-NB15	UNSW-NB15-75	UNSW-NB15-25	UNSW-NB15-TEST
Normal	56000	41911	14089	37000
Generic	40000	30081	9919	18871
Exploits	33393	25034	8359	11132
Fuzzers	18184	13608	4576	6062
DoS	12264	9237	3027	4089
Reconnaissance	10491	7875	2616	3496
Analysis	2000	1477	523	677
Backdoor	1746	1330	416	583
Shellcode	1133	854	279	378
Worms	130	99	31	44

4.2. The Aegean WiFi Intrusion Dataset (AWID)

The AWID was introduced due to the lack of publicly accessible datasets designated for WIDS design and development [51]. In comparison to the KDD Cup 99, the NSL-KDD and the UNSW-NB15 datasets which are all general purpose datasets used for IDS research as a whole, the AWID is among the first attempts to create a dataset that is solely generated from a wireless network traffic [52]. The traces in the AWID were not artificially generated but they were naturally produced from a Wireless Local Area Network (WLAN) that was secured by the Wired Equivalent Protocol (WEP). The authors in [51] argue that the AWID is a crucial contribution to WIDS research and they also claim that this is one of the first dataset of its kind to be accessible by the public.

The AWID is composed of two categories of datasets: the Full (F) AWID dataset and the Reduced (R) AWID dataset. The F and the R datasets contain two subsets: the training subset (Trn) and the test subset (Tst). As listed in Table 4 and Table 5, each of the AWID subsets have 155 attributes that include 154 inputs and one class instance that indicates whether a trace exhibits a legitimate or an intrusive behavior. In this paper, we have picked the following two subsets: the AWID-CLS-R-Trn utilized for the training and validation processes. The AWID-CLS-R-Tst employed for the testing procedure. The AWID-CLS-R-Trn and the AWID-CLS-R-Tst subsets includes four classes: normal, flooding, impersonation and injection. Table 6 shows the values distribution of the reduced AWID (AWID-CLS-R-Trn and AWID-CLS-Tst).

In this paper, we used 20% of the AWID-CLS-R-Trn (AWID-CLS-R-Trn-20) that has 359115 records and 20% of the AWID-CLS-R-Tst (AWID-Min-Tst) that contains 115128 records. We generated the AWID-Min-Trn (75% of the AWID-CLS-R-Trn-20) and the AWID-Min-Val (25% of the AWID-CLS-R-Trn-20) from the AWID-CLS-R-Trn-20. The AWID-Min-Trn is used during the training process, the AWID-Min-Val is used during the validation procedure and the AWID-Min-Tst is used for the testing process. Table 7 shows the values distribution of the minimal AWID.

Table 4
AWID Attributes- Part 1

No.	Name	No.	Name
f1	frame.interface_id	f33	radiotap.present.vht
f2	frame.dlt	f34	radiotap.present.reserved
f3	frame.offset_shift	f35	radiotap.present.rtap_ns
f4	frame.time_epoch	f36	radiotap.present.vendor_ns
f5	frame.time_delta	f37	radiotap.present.ext
f6	frame.time_delta_displayed	f38	radiotap.mactime
f7	frame.time_relative	f39	radiotap.flags.cfp
f8	frame.len	f40	radiotap.flags.preamble
f9	frame.cap_len	f41	radiotap.flags.wep
f10	frame.marke	f42	radiotap.flags.frag
f11	frame.ignored	f43	radiotap.flags.fcs
f12	radiotap.version	f44	radiotap.flags.datapad
f13	radiotap.pad	f45	radiotap.flags.badfcs
f14	radiotap.length	f46	radiotap.flags.shortgi
f15	radiotap.present.tsft	f47	radiotap.datarate
f16	radiotap.present.flags	f48	radiotap.channel.freq
f17	radiotap.present.rate	f49	radiotap.channel.type.turbo
f18	radiotap.present.channel	f50	fradiotap.channel.type.cck
f19	radiotap.present.fhss	f51	radiotap.channel.type.ofdm
f20	radiotap.present.dbm_antsignal	f52	radiotap.channel.type.2ghz
f21	radiotap.present.dbm_antnoise	f53	radiotap.channel.type.5ghz
f22	radiotap.present.lock_quality	f54	radiotap.channel.type.passive
f23	radiotap.present.tx_attenuation	f55	radiotap.channel.type.dynamic
f24	radiotap.present.db_tx_attenuation	f56	radiotap.channel.type.gfsk
f25	radiotap.present.dbm_tx_power	f57	radiotap.channel.type.gsm
f26	radiotap.present.antenna	f58	radiotap.channel.type.sturbo
f27	radiotap.present.db_antsignal	f59	radiotap.channel.type.half
f28	radiotap.present.db_antnoise	f60	radiotap.channel.type.quarter
f29	radiotap.present.rxflags	f61	radiotap.dbm_antsignal
f30	radiotap.present.xchanne	f62	radiotap.antenna
f31	radiotap.present.mcs	f63	radiotap.rxflags.badplcp
f32	radiotap.present.ampdu	f64	wlan.fc.type_subtype

4.3. Feature Extraction

4.3.1. Extra Trees ML method

The ET is an algorithm similar to the RF approach because ET also fits several DTs in order to perform classification or regression. The major differences between ET and RF lie in the fact that ET separates the nodes by picking cut-points randomly and that instead of using sub-samples, ET uses the whole dataset in order to grow the required trees [26].

4.3.2. Feature Extraction Algorithm

The algorithms listed in this section of the research provide an overview of how we first normalize all the inputs before applying the ET algorithm for the purpose of extracting the most important features of the UNSW-NB15 dataset.

Table 5
AWID Attributes - Part 2

No.	Name	No.	Name
f65	wlan.fc.version	f97	wlan_mgt.fixed.capabilities.spec_man
f66	wlan.fc.type	f98	wlan_mgt.fixed.capabilities. short_slot_time
f67	wlan.fc.subtype	f99	wlan_mgt.fixed.capabilities.apsd
f68	wlan.fc.ds	f100	wlan_mgt.fixed.capabilities. radio_measurement
f69	wlan.fc.frag	f101	wlan_mgt.fixed.capabilities.dsss_ofdm
f70	wlan.fc.retry	f102	wlan_mgt.fixed.capabilities.del_blk_ack
f71	wlan.fc.pwrmgt	f103	wlan_mgt.fixed.capabilities.imm_blk_ack
f72	wlan.fc.moredata	f104	wlan_mgt.fixed.listen_ival
f73	wlan.fc.protected	f105	wlan_mgt.fixed.current_ap
f74	wlan.fc.order	f106	wlan_mgt.fixed.status_code
f75	wlan.duration	f107	wlan_mgt.fixed.timestamp
f76	wlan.ra	f108	wlan_mgt.fixed.beacon
f77	wlan.da	f109	wlan_mgt.fixed.aid
f78	wlan.ta	f110	wlan_mgt.fixed.reason_code
f79	wlan.sa	f111	wlan_mgt.fixed.auth.alg
f80	wlan.bssid	f112	wlan_mgt.fixed.auth_seq
f81	wlan.frag	f113	wlan_mgt.fixed.category_code
f82	wlan.seq	f114	wlan_mgt.fixed.htact
f83	wlan.bar.type	f115	wlan_mgt.fixed.chanwidth
f84	wlan.ba.control.ackpolicy	f116	wlan_mgt.fixed.fragment
f85	wlan.ba.control.multitid	f117	wlan_mgt.fixed.sequence
f86	wlan.ba.control.cbitmap	f118	wlan_mgt.tagged.all
f87	wlan.bar.compressed.tidinfo	f119	wlan_mgt.ssid
f88	wlan.ba.bm	f120	wlan_mgt.ds.current_channel
f89	wlan.fcs_good	f121	wlan_mgt.tim.dtim_count
f90	wlan_mgt.fixed.capabilities.ess	f122	wlan_mgt.tim.dtim_period
f91	wlan_mgt.fixed.capabilities.ibss	f123	wlan_mgt.tim.bmapctl.multicast
f92	wlan_mgt.fixed.capabilities. cfpoll.ap	f124	wlan_mgt.tim.bmapctl.offset
f93	wlan_mgt.fixed.capabilities. privacy	f125	wlan_mgt.country_info.environment
f94	wlan_mgt.fixed.capabilities. preamble	f126	wlan_mgt.rsn.version
f95	wlan_mgt.fixed.capabilities.pbcc	f127	wlan_mgt.rsn.gcs.type
f96	wlan_mgt.fixed.capabilities.agility	f128	wlan_mgt.rsn.pcs.count
f129	wlan_mgt.rsn.akms.count	f145	wlan.qos.tid
f130	wlan_mgt.rsn.akms.type	f146	wlan.qos.priority
f131	wlan_mgt.rsn.capabilities.preauth	f147	wlan.qos.eosp
f132	wlan_mgt.rsn.capabilities.no_pairwise	f148	wlan.qos.ack
f133	wlan_mgt.rsn.capabilities. ptksa_replay_counter	f149	wlan.qos.amsdupresent
f134	wlan_mgt.rsn.capabilities. gtksa_replay_counter	f150	wlan.qos.buf_state_indicated
f135	wlan_mgt.rsn.capabilities.mfpr	f151	wlan.qos.bit4
f136	wlan_mgt.rsn.capabilities.mfpc	f152	wlan.qos.txop_dur_req
f137	wlan_mgt.rsn.capabilities.peerkey	f153	wlan.qos.buf_state_indicated
f138	wlan_mgt.tcprep.trsmt_pow	f154	data.len
f139	wlan_mgt.tcprep.link_mrg	f155	class
f140	wlan.wep.iv		
f141	wlan.wep.key		
f142	wlan.wep.icv		
f143	wlan.tkip.extiv		
f144	wlan.ccmp.extiv		

Feature normalization is an extremely crucial step when pre-processing data to be fed into a ML model. Inputs have different data formats that could be numbers (integers, floats, doubles, binary, etc.) or they could be nominal (strings, chars, text, etc.). Furthermore, the numbers in a dataset can be very large or too small. Given the fact that ML models can only compute numbers, non-numeric inputs need to be mapped to numbers or to be converted in one-hot-encoded values. One-hot-encoding is a procedure whereby the nominal attributes of a dataset are transformed into a numerical format that can be processed by ML or DL algorithm [53]. Moreover, the values are capped within a chosen range for an optimal processing using Equation (5).

$$x_{normalized} = (b-a)\frac{x_i - min(x_i)}{max(x_i) - min(x_i)} \quad (5)$$

where b and a are chosen based on preference.

For a feature vector represented by $F_v(f_1, ..., f_k)$, $1 < k < Z$, where Z represents the maximum number of features and C the class label in the dataset, the normalization process happens as follow:

Algorithm 1 Data Normalization Process

Input: $F_v(f_1, ..., f_k)$, $1 < k < K$
Output: $F_{transformed}(f_1^t, ..., f_k^t)$:
  for i from 1 to k **do**
    if (f_i a nominal input) **then**
      Step 1: use one-hot-encoding or scikit-learn mapping
      Step 2: data normalization with $(b-a)\frac{f_i - min(f_i)}{max(f_i) - min(f_i)}$
    end if
    Step 1: data normalization with $(b-a)\frac{f_i - min(f_i)}{max(f_i) - min(f_i)}$
  end for

Table 6
The reduced AWID Values Distribution

Attack Type	AWID-CLS-R-Trn	AWID-CLS-R-Tst
Normal	1633190	530785
injection	65379	16682
impersonation	48522	20079
flooding	48484	8097

Table 7
The minimal AWID Values Distribution

Attack Type	AWID-CLS-R-Trn-20	AWID-Min-Trn	AWID-Min-Val	AWID-Min-Tst
Normal	326311	244733	81578	104670
injection	13220	9915	3305	4294
impersonation	9853	7390	2463	3083
flooding	9731	7298	2433	3081

After the normalization process, the features are ranked using a measure called: Feature Importance (FI). The FI of a particular input in relation to the class attribute is computed using the ET method in Algorithm 2.

4.4. Deep Neural Networks

ANNs composed of an input layer, multiple hidden layers and an output layer are considered to be *deep neural networks* (DNNs) [9]. The primary building block of

Algorithm 2 Features FI Ranking Algorithm

Input: $F_{transformed}(f_1^t, ..., f_k^t)$
Output: F_{ranked}
Step 1: Initialize ET Model
Step 2: Fit ET Model
Step 3: Save importance scores in FI_{temp}
Step 4:
for i from FI_{temp} **do**
  if $(FI_i \geqslant FI_{threshold})$ **then**
    load FI_i into F_{ranked}
  end if
end for

a DNN is a neuron[10]. In this research we refer to neurons as nodes interchangeably. A simple DNN is shown in Fig. 1 whereby the input layer has 8 nodes. The first hidden layer has 5 nodes and the second hidden layer contains 3 nodes. The output layer has a single node. In order to improve the approximation ability of a DNN, one of the following activation functions can be applied at each node [54, 55]:

$$\text{Sigmoid: } \sigma = \frac{1}{1 + e^{-t}} \tag{6}$$

$$\text{Rectified Linear Unit (ReLU): } f(t) = max(0, t) \tag{7}$$

$$\text{Hyperbolic Tangent: } tanh(t) = \frac{1 - e^{-2t}}{1 + e^{-2t}} \tag{8}$$

There exist different categories of neural networks including Feed Forward Neural Networks [27, 38, 46], RNNs [40], Convolutional Neural Networks [18], Autoencoders[18], etc. In this paper, we consider FFDNNs for our experiments. In a FFDNN, as shown in Fig. 1, the flow of information is unidirectional, from the input layer to the output layer. There is no consensus in the literature for the definition of "deep"; however, in this work, a FFDNN is labeled as deep if it has more then two hidden layers.

4.5. The Proposed FFDNN IDS with a wrapper based Feature Extraction Unit

The architecture depicted in Fig. 2 represents the the WFEU-FFDNN-IDS. This architecture gives an overview of our proposed system whereby the first step consists of preparing the data for computation. In this step, we use the ET algorithm wrapper based features selection process. The full UNSW-NB15 dataset is normalized and the Extra Trees method is used over the dataset to generate an array, FI_{temp}, of features importance with respect to the label attribute. From this array, a secondary array, FI_{ranked}, is generated based on a feature importance threshold, $FI_{threshold}$. This threshold is selected during an experimental trial iteration.

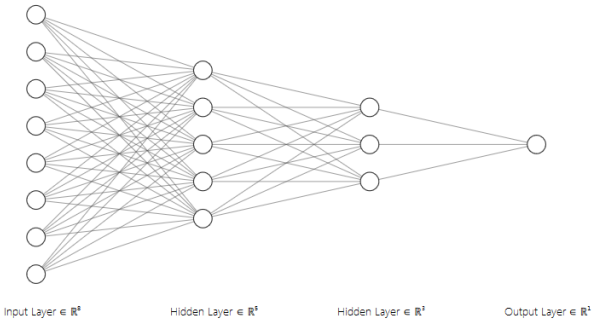


Fig. 1. A simple DNN

In the second step, we use a strategy inspired from [27] by dividing the training set in two partitions as explained in section 4.1.: the UNSW-NB15-75 for training and UNSW-NB15-25 for validation. Furthermore, we normalize all the attributes of the input vector to generate the features vector with normalized values. Moreover, the array from the wrapper-based features selection procedure will dictate which features will be processed during the training and validation steps.

In the last step, an FFDNN model is trained using: Forward Propagation, Back-Propagation of the calculated errors and the updates of weights and corresponding biases[27, 56]. The final output is the WFEU-FFDNN Model that is evaluated and tested.

5. Experiments and Discussions

5.1. Hardware and Software setups

The experiments carried out in this work were conducted using Scikit-Learn[57] which is a Python ML library that we ran on Windows 8.1 64-bit Operating System. This ML library is extensively used for ML, DL and data science research. In terms of hardware, our simulations were carried out using an ASUS Notebook Intel Core i3-3217U CPU @1.80GHz with 4G RAM.

5.2. Performance Measure

The performance metrics used for classification problems are derived from the following four possibilities:

- True positive (TP): attacks that are correctly classified as intrusions.

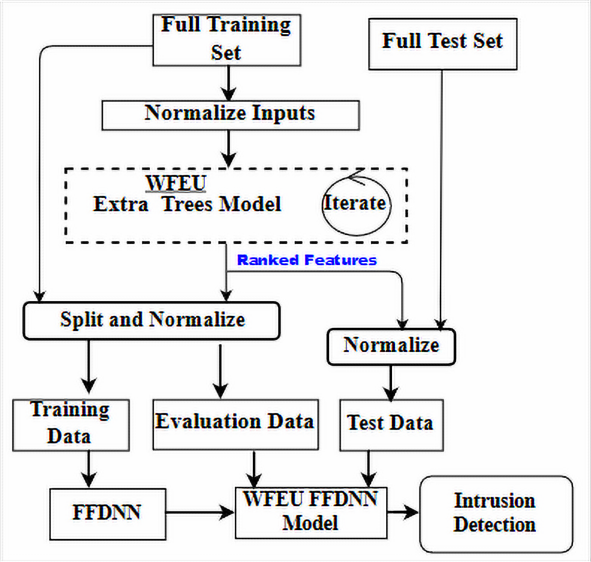


Fig. 2. The Proposed IDS Architecture

- True Negative (TN): genuine activities that are successfully classified as normal (legitimate).
- False positive (FP): non-intrusive activities that are wrongly labeled as attacks by the IDS.
- False Negative (FN): suspicious network activities that are classified as normal.

The accuracy (AC), the Precision and the Recall are derived from the above conditions as follow:

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

The AC on the test data is the main metric that we will consider in this research.

5.3. Wrapper Based Feature extraction

The features listed in Table 8 and Table 9 are the results from the application of Algorithm 1 and Algorithm 2 on the UNSW-NB15 dataset and the AWID, respectively. In the instance of the UNSW-NB15, the ET Algorithm in the WFEU generated an array with the feature importance factors (IF) that enabled us to pick the most suitable candidate features. This wrapper based feature extraction process produced a set of 22 optimal features whereby feature 31 (f31: ct_srv_src) with an IF = 0.125573 is the most relevant input. In the instance of the AWID, the WFEU generated a set of 26 optimal features whereby feature 75 (f75: wlan.duration) with an IF = 0.099840 is the most important attribute.

Table 8
UNSW-NB15 Reduced Features Set

No.	Feature Name	Importance Factor
f31	ct_srv_src	0.125573
f10	sttl	0.094520
f9	rate	0.076320
f12	sload	0.070620
f8	dbytes	0.041252
f15	dloss	0.039846
f11	dttl	0.037874
f40	ct_src_ltm	0.030439
f30	response_body_len	0.029490
f3	service	0.026663
f27	smean	0.025744
f6	dpkts	0.025600
f7	sbytes	0.024240
f26	ackdat	0.021015
f35	ct_dst_sport_ltm	0.019919
f5	spkts	0.019777
f19	djit	0.019592
f16	sinpkt	0.019439
f34	ct_src_dport_ltm	0.019201
f24	tcprtt	0.017525
f23	dwin	0.017200
f18	sjit	0.017169

5.4. Experiments and Discussions

This section details all the experiments conducted in this research. In order to put the results obtained by the WFEU-FFDNN-IDS model in perspective, we also conducted experiments on the following standard ML methodologies: kNN, SVM, DT, RF and NB. Furthermore, we implement both the binary classification where there are only two classes (*normal* = 0 and *attack* = 1) and the multiclass classification with 10 classes (*Normal* = 0, *Generic* = 1, *Exploits* = 2, *Fuzzers* = 3, *DoS* = 4, *Reconnaissance* = 5, *Analysis* = 6, *Backdoor* = 7, *Shellcode* = 8, *Worms* = 9). Moreover, the simulations were carried in two folds. Firstly, we applied the UNSW-NB15 with the full set of features on the kNN, SVM, DT, NB, RF and FFDNN. Secondly, we implement the wrapper based feature extraction method of the WFEU and with the generated input vector, we re-run the experiments performed in the first phase. Moreover, we implement the binary and the muticlass schemes using the full set of features of the AWID. Additionally,

Table 9
AWID Reduced Set of Features

No.	Feature Name	Importance Factor
f75	wlan.duration	0.099840
f38	radiotap.mactime	0.070337
f68	wlan.fc.ds	0.068456
f71	wlan.fc.pwrmgt	0.067939
f67	wlan.fc.subtype	0.067175
f140	wlan.wep.iv	0.064220
f47	radiotap.datarate	0.056081
f7	frame.time_relative	0.041923
f51	radiotap.channel.type.ofdm	0.037753
f110	wlan_mgt.fixed.reason_code	0.037529
f50	radiotap.channel.type.cck	0.037394
f9	frame.cap_len	0.032410
f8	frame.len	0.031858
f73	wlan.fc.protected	0.031522
f64	wlan.fc.type_subtype	0.030453
f4	frame.time_epoch	0.028216
f70	wlan.fc.retry	0.025550
f142	wlan.wep.icv	0.024775
f164	wlan.fc.type	0.023719
f61	radiotap.dbm_antsignal	0.021252
f154	data.len	0.017520
f107	wlan_mgt.fixed.timestamp	0.009610
f97	wlan_mgt.fixed.capabilities.spec_man	0.007806
f112	wlan_mgt.fixed.auth_seq	0.007080
f89	wlan.fcs_good	0.007039
f108	wlan_mgt.fixed.beacon	0.005803

using the reduced set of features of the AWID, we also carry out experiments for the binary and the multiclass classification processes. In the multiclass category, we considered all four classes of the AWID (*normal* = 0, *flooding*= 1, *impersonation* = 2 and *injection* = 3) .

5.4.1. *UNSW-NB15: Experiments with the full Feature Set*

In this section, the experiments were achieved using the full set of features (42 in total) of the clean UNSW-NB15 dataset. Table 10 shows the training and testing of various FFDNN models with 3 hidden layers each. The aim of this procedure is to uncover which model performs relatively well compared to other models. Although we represented the models configured with 30, 40, 80 and 150 nodes; the experiments were performed using many more models. However, these models were the ones that showed significant changes when the testing process occurred. The procedure described in this section is followed throughout our experiments. Moreover, in each of the FFDNN experiment, the learning rate is a value (a parameter) that controls how fast and by how much a model's error is updated. The learning rate is selected empirically [57]. As depicted in Table 10, for the binary classification with the full set of features in the UNSW-NB15 dataset, the FFDNN model that ran optimally possessed 3 hidden layers composed of 150 nodes, a learning rate of 0.02 and a test accuracy (accuracy computed on the test data) of 87.48%.

Fig. 3. Binary Classification Accuracy Comparison -Full Set of Features

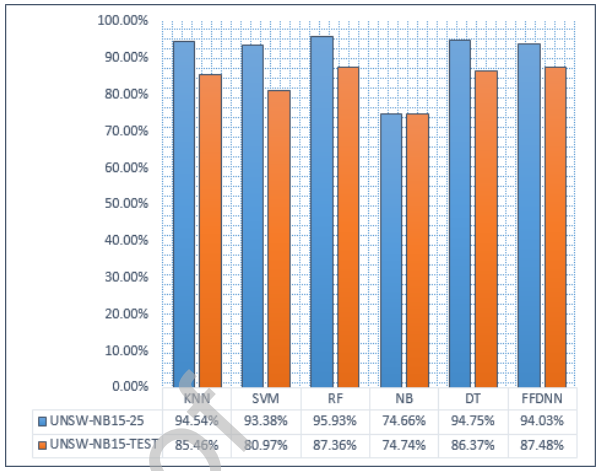


Table 10
FFDNN Training - Binary Classification- Full Set of Features

No. of Nodes	Learning Rate	No. of Hidden Layers	UNSW-NB15-25 AC.	UNSW-NB15-TEST AC.
30	0.01	3	94.14%	84.47%
30	0.02	3	93.79%	84.75%
30	0.05	3	93.73%	81.51%
40	0.01	3	94.06%	84.78%
40	0.02	3	93.70%	81.43%
40	0.05	3	93.64%	82.45%
80	0.01	3	94.46%	85.01%
80	0.02	3	94.29%	84.94%
80	0.05	3	93.82%	82.76%
150	0.01	3	94.44%	84.71%
150	0.02	3	94.34%	87.48%
150	0.05	3	94.13%	87.37%

Additionally, we ran experiments on the existing ML algorithms (kNN, SVM, RF, NB and DT) using the full set of features of the UNSW-NB15 dataset. A result analysis is presented in Fig. 3 whereby the FFDNN model outperformed all other presented ML models. RF and DT ML algorithms came second with the tests accuracy of 87.36% and 86.37%.

The FFDNN models training and testing outcomes listed in Table 11 represent those of the multiclass clas-

Table 11
FFDNN Training - Multiclass Classification- Full Set of Features

No. of Nodes	Learning Rate	No. of Hidden Layers	UNSW-NB15-25 AC.	UNSW-NB15-TEST AC.
30	0.01	3	78.94%	70.93%
30	0.02	3	71.42%	65.69%
30	0.05	3	53.48%	62.05%
40	0.01	3	79.60%	69.86%
40	0.02	3	75.89%	71.54%
40	0.05	3	32.14%	44.93%
80	0.01	3	80.80%	75.83%
80	0.02	3	79.36%	73.18%
80	0.05	3	38.61%	45.36%
150	0.01	3	81.15%	73.45%
150	0.02	3	80.03%	68.66%
150	0.05	3	60.92%	67.59%

sification process. The model we selected had an accuracy of 75.83% on the test data with 80 neurons spread over 3 hidden layers and a learning rate of 0.01. Furthermore, in Fig. 4, a performance comparison in terms of the test accuracy was conducted against the traditional ML methods that were studied. The FFDNN was leading in terms of effectiveness.

5.4.2. UNSW-NB15: Experiments with WFEU Generated Feature Set

Table 12
FFDNN Training - Binary Classification - Reduced Set of Features

No. of Nodes	Learning Rate	No. of Hidden Layers	UNSW-NB15-25 AC.	UNSW-NB15-TEST AC.
30	0.01	3	93.26%	81.01%
30	0.02	3	93.02%	80.18%
30	0.05	3	92.79%	79.13%
40	0.01	3	92.38%	87.10%
40	0.02	3	93.71%	82.81%
40	0.05	3	93.63%	82.60%
80	0.01	3	92.96%	82.55%
80	0.02	3	72.26%	75.08%
80	0.05	3	93.32%	80.46%
150	0.01	3	93.32%	80.46%
150	0.02	3	92.63%	78.78%
150	0.05	3	92.79%	79.94%

In this section, the experiments were performed using

Fig. 4. Multiclass Classification Accuracy Comparison - Full Set of Features

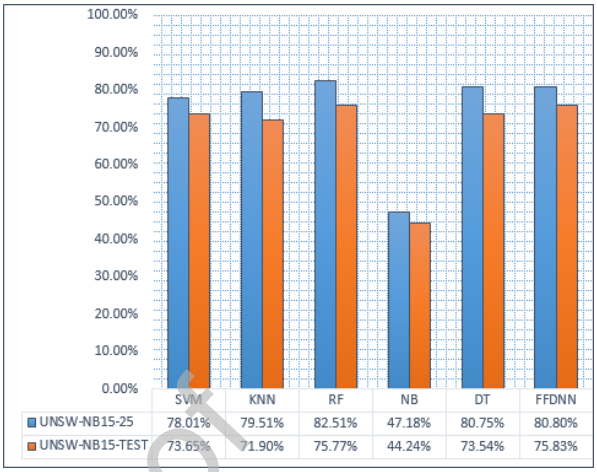
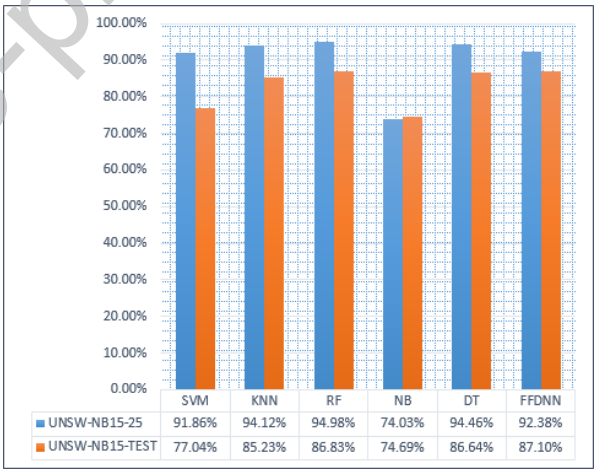


Fig. 5. Binary Classification Accuracy Comparison -Reduced Set of Features



the reduced set of features (22 in total) of the UNSW-NB15 dataset generated by the WFEU. This set of features is listed in Table 8.

We first implemented the WFEU-FFDNN method for the binary classification by training all the models with the strategy used in Table 10. The results are depicted in Table 12. The best performing model had 40 nodes in 3 hidden layers and it yielded a test accuracy of 87.10%. In order to benchmark our performance results, we further applied the WFEU to the following ML methods: SVM, kNN, RF, NB and DT. As shown in Fig. 5, the FFDNN based model outmatches other models.

Table 13 depicts the results of the multiclass classification process ran on FFDNN models with a reduced

Table 13
FFDNN Training - Multiclass Classification - Reduced Set of Features

No. of Nodes	Learning Rate	No. of Hidden Layers	UNSW-NB15-25 AC.	UNSW-NB15-TEST AC.
30	0.01	3	70.25%	58.96%
30	0.02	3	77.14%	69.51%
30	0.05	3	32.00%	44.93%
40	0.01	3	76.77%	69.28%
40	0.02	3	77.26%	68.01%
40	0.05	3	31.93%	44.93%
80	0.01	3	77.07%	70.00%
80	0.02	3	76.16%	71.77%
80	0.05	3	63.30%	58.54%
150	0.01	3	79.59%	72.53%
150	0.02	3	80.92%	77.16%
150	0.05	3	67.14%	63.66%

feature vector whereby the model with 150 nodes spread over 3 middle layers at a learning speed of 0.02 and produced a test accuracy of 77.16%. In comparison to the standard ML methodologies as illustrated in Fig. 6, the results demonstrated the superiority of the FFDNN seconded by RF and DT ML methods with 76.82% and 74.35% on test data.

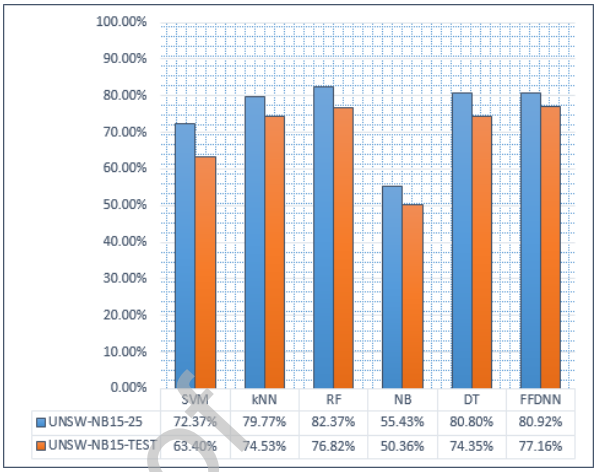
5.4.3. *UNSW-NB15: Experiments with WFEU (IG) Generated Feature Set*

In this phase, the experiments were performed using a reduced feature vector of the UNSW-NB15. This array was generated using the IG filter-based feature selection algorithm that was presented in [27]. Table 14 shows the features that were selected as well as their IG values. The classification models used in this instance are FFDNNs. The results for the binary classification are shown in Table 15 whereby the best model obtained an overall test accuracy of 85.48% with a learning rate of 0.01 using 60 neurons distributed over 3 hidden layers. Table 16 highlights the results obtained from the multiclass classification process. The model that performed optimally achieved an overall accuracy of 74.78% with a learning rate of 0.01 and 150 neurons spread over 3 hidden layers.

5.4.4. *AWID: Experiments with the Full Set of Features versus the WFEU Generated Set of Features*

In the first step, we implemented a number FFDNNs using the full set of features of the AWID (154 features). The results are shown in Table 17 and Table 18. In the

Fig. 6. Multiclass Classification Accuracy Comparison - Reduced Set of Features



instance of the binary classification configuration; the best FFDNN achieved an accuracy of 98.62% on the validation data and 98.69% on the test data. This model used 40 neurons distributed in 3 hidden layers and a learning rate of 0.001. For the multiclass classification scheme, the best model obtained an accuracy of 98.47% over the validation data and 98.59% on the test data. This model was configured with 40 neurons spread over 3 hidden layers and with a learning rate of 0.001.

In the second step, we implemented the framework described in Figure 2 using the AWID with a reduced set of features. The results described in Table 19 depict the training, the validation and the testing processes of several FFDNN models. During the experiments, we considered the optimal feature vector (26 features) generated by the WFEU as listed in Table 9 . The classification scheme used in this phase of our experimental process is the binary classification. The performance metrics considered were the number of neurons, the learning rate, the size of the hidden layers, the accuracy of validation data and the accuracy on test data. The most important metric that was considered for decision making is the accuracy on test data. As depicted in Table 19, the FFDNN model that achieved the best performance got 99.67% on validation data and 99.66% on test data. This model was configured using 60 neurons spread over 3 hidden layers with a learning rate of 0.001. In comparison to the results obtained by the best model using the full set of features; this represents an increase of 1.05% in accuracy over the validation data and 0.97% on the test data. Moreover, the use of the reduced set of features has allowed for more models to achieve an improved performance. The models are highlighted in

Table 14
UNSW-NB15 IG-Reduced Features Set

No.	Feature Name	Info. Gain Factor
f7	sbytes	0.466848717
f10	sttl	0.386406914
f8	dbytes	0.372411927
f11	dttl	0.355313624
f32	ct_state_ttl	0.353079544
f9	rate	0.351145878
f12	sload	0.345455257
f27	smean	0.337612213
f1	dur	0.337184455
f28	dmean	0.314022615
f17	dinpkt	0.296980629
f13	dload	0.280154486
f6	dpkts	0.27310762
f16	sinpkt	0.244757347
f24	tcprtt	0.235729867
f25	synack	0.23345685
f26	ackdat	0.232389399
f18	sjit	0.204299072
f5	spkts	0.182391446
f4	state	0.18111681
f19	djit	0.175168184

yellow in Table 19 and Table 20.

The results shown in Table 20 describe the performance of the multiclass classification process using the optimal features vector. In this instance, all the 4 classes of the AWID were considered. As depicted in Table 20, the greatest performance was obtained by a FFDNN model configured with a learning rate of 0.001 and built using 30 neurons distributed over 3 hidden layers. This model achieved an accuracy 99.78% on validation data and an accuracy of 99.77% on test data. These results represent an increase in comparison to the best model that we obtained using the full set of features.

5.4.5. Discussions

The work presented in this research investigated the feasibility of applying FFDNNs to solve the issue of wireless network intrusions detection using the UNSW-NB15 and the AWID. Several experimental processes were conducted whereby we mainly studied the binary classification and the multiclass classification types of attacks. In the instance of the UNSW-NB15; for binary classification using the full feature vector (FFV), the FFDNN outperformed other models by achieving a detection accuracy of 87.48% over the test dataset using 150 neurons. For the multiclass classification using the FFV, the FFDNN achieved a test accuracy of 75.83%;

Table 15
UNSW-NB15: Binary Classification - Accuracy Comparison - IG Reduced Set of Features

No. of Nodes	Learning Rate	No. of Hidden Layers	UNSW-NB15-25 AC.	UNSW-NB15-TEST AC.
30	0.001	3	93.29%	80.75 %
30	0.01	3	93.47%	81.55 %
30	0.05	3	92.77%	79.38 %
30	0.1	3	68.07%	55.06 %
40	0.001	4	93.41%	80.97%
40	0.01	4	93.65%	83.36%
40	0.05	4	93.37%	80.88%
60	0.01	3	93.55%	85.48%
60	0.02	3	93.58%	83.43%
60	0.05	3	93.47%	81.29%
80	0.01	4	93.43%	84.45%
80	0.05	4	73.93%	75.28%
150	0.01	3	93.80%	82.05%

however, after applying the WFEU, the accuracy increased by 1.29% to 77.16%. On the validation data, the FFV-FFDNN achieved 94.34% and the WFEU-FFDNN got an accuracy of 92.38%.

Furthermore, After the implementation of the filter-based feature extraction method presented in [27], we generated a reduced set of features composed of 21 attributes. Using the extracted array with FFDNNs, we implemented the binary classification process whereby the best model obtained an accuracy of 85.48%. However, the optimal model configured using our proposed method (in Table 12) achieved an accuracy of 87.10%. For the multiclass classification setting; the model using our proposed approach (in Table 13) obtained an accuracy of 77.16% whereas the WFEU (IG) FFDNN model (in Table 16) achieved an accuracy of 74.78%. All the experiments in this paragraph were carried out within the same environment. Therefore, in this instance, the ET wrapper-based is a more optimal option in comparison to the IG filter-based method when using the UNSW-NB15 dataset

Moreover, in comparison to the ANN method presented in [45] that yielded an accuracy of 81.34% on the UNSW-NB15 validation set, our proposed method performed much better. Furthermore, using the FFV of the UNSW-NB15, the DEA-DFFNN IDS presented in [46] achieved a detection accuracy of 92.4% on validation data which is 1.94% less than our proposed approach. Additionally, our proposed method excelled in comparison to the work presented in [34] whereby a Deep Neu-

Table 16
UNSW-NB15: Multiclass Classification - Accuracy Comparison - IG Reduced Set of Features

No. of Nodes	Learning Rate	No. of Hidden Layers	UNSW-NB15-25 AC.	UNSW-NB15-TEST AC.
30	0.001	3	72.21%	66.29%
30	0.01	3	65.94%	61.86%
30	0.05	3	53.09%	61.72%
30	0.1	3	52.95%	61.49%
40	0.001	4	73.43%	70.03%
40	0.01	4	73.59%	69.23%
40	0.05	4	32.18%	44.93%
60	0.01	3	67.27%	65.95%
60	0.02	3	64.77%	60.96%
60	0.05	3	53.37%	61.75%
80	0.01	4	67.62%	66.18%
80	0.05	4	67.48%	65.88%
150	0.01	3	78.84%	74.78%
150	0.02	3	67.93%	67.11%

ral Network(DNN) with 5 layers got 76.1% for the binary classification and 65.1% for the multiclass classification on the UNSW-NB15 test set. Additionally, our proposed technique yielded an improvement of 5.68% in performance in comparison to the work presented in [41] where the DT-GA-LR achieved an overall accuracy of 81.42% for the binary classification. Moreover, in contrast with the research presented in [44] whereby a test accuracy of 70.51% was obtained for the multiclass classification scheme using the UNSW-NB15; our research registered an increase of 6.65% in accuracy for the same configuration.

In terms of the speed of convergence, we computed and compared WFEU-FFDNN models with 3 layers of 30 (WFEU-FFDNN_1), 40 (WFEU-FFDNN_2), 80 (WFEU-FFDNN_3) and 150 (WFEU-FFDNN_4) nodes, all with a learning rate of 0.02 over 100 epochs. As depicted in Fig.7, WFEU-FFDNN_4 started converging at around 25 epochs without over-fitting. This procedure was used to verify that the WFEU-FFDNN models increase in accuracy from the standard FFV-FFDNN models can be justified using the loss as a performance metric.

Furthermore, during our experiments, we have discovered that the performance of Naive Bayes (NB) ML method decreases dramatically for multiclass classification on the UNSW-NB15 dataset. Using the FFV, NB achieved a test accuracy of 44.24%; however, after the reduction of the input vector to 22 optimal features, NB

Table 17
FFDNN Training (AWID): Binary Classification - Accuracy Comparison - Full Set of Features

No. of Nodes	Learning Rate	No. of Hidden Layers	AWID-Min-Val AC.	AWID-Min-Tst AC.
15	0.001	3	97.17%	97.15%
15	0.01	3	97.76%	97.79%
15	0.02	3	98.18%	98.20%
30	0.001	3	97.30%	97.31%
30	0.01	3	97.85%	97.81%
30	0.02	3	97.41%	97.40%
40	0.001	3	98.62%	98.69%
40	0.01	3	98.14%	98.15%
40	0.02	3	97.60%	97.59%
60	0.001	3	98.52%	98.54%
60	0.01	3	96.74%	96.76%

Table 18
FFDNN Training (AWID): Multiclass Classification - Accuracy Comparison - Full Set of Features

No. of Nodes	Learning Rate	No. of Hidden Layers	AWID-Min-Val AC.	AWID-Min-Tst AC.
15	0.001	3	94.79%	94.83%
15	0.01	3	92.95%	92.88%
15	0.02	3	94.50%	94.55%
30	0.001	3	98.46%	98.49%
30	0.01	3	90.93%	90.91%
30	0.02	3	90.96%	90.99%
40	0.001	3	98.47%	98.59%
40	0.01	3	91.92%	91.87%
40	0.02	3	92.94%	92.96%
40	0.05	3	93.98%	93.95%
60	0.001	3	98.35%	98.36%

achieved 50.36%. This represents a significant increase of 6.12%. Additionally, the feature engineering procedure has also facilitated an improvement in accuracy on the test data of the kNN algorithm with an increase of 2.6% from 71.90% to 74.53%. The RF approach advanced from 75.77% to 76.82%, which is an increase of 1.05% and the DT algorithm got a step-up of 0.81% from 73.54% to 74.35%.

Moreover, in the case of the AWID, we investigated the possibility of using FFDNNs in the aim to design an effective and robust WIDS. We used the WFEU in order to generate an optimal input vector. This array was composed of 26 most relevant attributes with respect to the class attribute. This reduced array is about 17% of the full feature vector containing 154 features. We carried

Table 19
FFDNN Training (AWID): Binary Classification - Accuracy Comparison - Reduced Set of Features

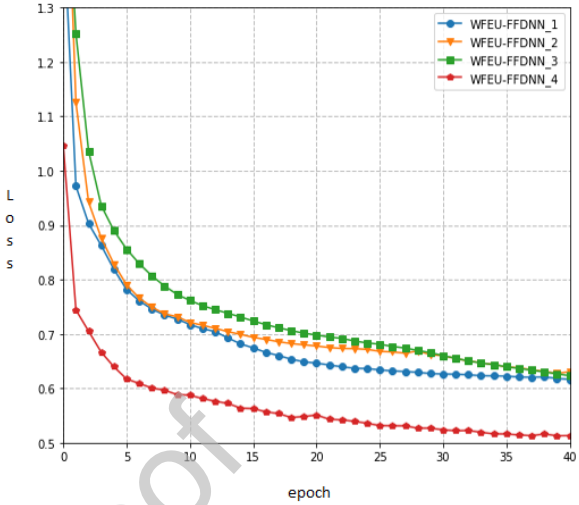
No. of Nodes	Learning Rate	No. of Hidden Layers	AWID-Min-Val AC.	AWID-Min-Tst AC.
15	0.001	3	90.89%	90.91%
15	0.01	3	96.37%	96.39%
15	0.02	3	90.89%	90.91%
30	0.001	3	99.43%	99.42%
30	0.01	3	97.28%	97.29%
30	0.02	3	90.98%	90.91%
40	0.001	3	99.38%	99.38%
40	0.01	3	94.56%	94.57%
40	0.02	3	95.00%	95.03%
60	0.001	3	99.67%	99.66%
60	0.01	3	95.95%	95.90%

Table 20
FFDNN Training (AWID): Multiclass Classification - Accuracy Comparison - Reduced Set of Features

No. of Nodes	Learning Rate	No. of Hidden Layers	AWID-Min-Val AC.	AWID-Min-Tst AC.
15	0.001	3	94.84%	94.81%
15	0.01	3	93.71%	93.72%
15	0.02	3	97.85%	97.85%
30	0.001	3	99.78%	99.77%
30	0.01	3	90.94%	90.91%
30	0.02	3	90.96%	90.91%
40	0.001	3	98.72%	98.70%
40	0.01	3	96.57%	96.60%
40	0.02	3	90.94%	90.91%
40	0.05	3	91.01%	90.91%
60	0.001	3	99.70%	99.71%

out simulations for the binary and the multiclass classification schemes. In contrast with the work presented in [49] whereby 35 attributes of the AWID were considered during the classification process; our proposed framework achieved a test accuracy of 99.66% for the binary classification configuration whereas their system got 99.10%. In the instance of the multiclass classification scheme, our system achieved 99.77% on test data whereas their method yielded 98.87%. In the work in [47], the overall accuracy that was obtained is 98.54%. In contrast to the research conducted in [47], our framework achieved 99.99%, 93.04%, 82.35%, 99.93% for the following classes, respectively: normal, flooding, impersonation and injection. The overall accuracy in our case is 99.77% as depicted in Table 20.

Fig. 7. WFEU-FFDNN Models Convergence



This research demonstrates that the usage of a feature selection method in conjunction with FFDNN increases the performance of a model's capacity to classify various types of attacks when applied to test data. It is crucial to note that the true performance of a model is dictated by its performance on new data (data that the model has never seen before). In this research, the new datasets that were employed are the following: UNSW-NB15-TEST and the AWID-Min-Tst. On the UNSW-NB15, both a filter based (IG) and a wrapper based (ET) feature selection methods were applied. The results demonstrated that these methods had a positive impact by increasing the accuracy of detection of FFDNNs. However, the wrapper-based approach is a better alternative in comparison to the filter-based. In the instance of the AWID, experiments were ran using the full set of features (154) as well as the WFEU generated set of features (26). The results show that the usage a reduced set of features enabled multiple models to perform efficiently on the test data.

Moreover, during the course of our experiments; the training, the validation and the testing processes of various DL and ML models were computationally expensive and time consuming. This is one of the limitations of our research. Based on these factors, it could be of substantial benefit to consider implementing our research on computers with more resources in terms of computing power as well as on GPU enabled hardware [58] for better scalability.

6. Conclusion

In this paper, an IDS for wireless network traffic coupled to a wrapper based feature extraction technique was proposed. The IDS was built using Feed Forward Deep Neural Networks and the feature extraction method was implemented using the Extra Trees (ET) algorithm. The ET generated an optimal feature vector. The experiments were carried out using the UNSW-NB15 intrusion dataset. The full UNSW-NB15 was divided into two separate sets namely, the UNSW-NB15-75 (75% of the UNSW-NB15) for training and the UNSW-NB15-25 (25% of the UNSW-NB15) for validation. The models were tested on the UNSW-NB15-TEST dataset which is independent from the full UNSW-NB15 dataset. In the first phase, we implemented FFDNNs over the full feature space and compared the outcome to regular ML methods including NB, SVM, RF, kNN and DT. The outcome suggested that the FFDNN approach was superior to other techniques with an accuracy of 94.34% on validation data and 87.48% on test data for binary classification. In the second phase, we implemented the WFEU with the FFDNNs models. Moreover, we also implemented the WFEU with traditional ML algorithms. The results showed that the best WFEU-FFDNN outperformed classical ML approaches both for the binary classification as well as the multiclass classification settings. In the third phase of the experimental process, we compared the accuracy of our proposed method to that of our previously implemented method in [27]. The results demonstrated that in comparison to the filter-based feature selection method; the wrapper-based feature extraction method is a better option when using the UNSW-NB15 dataset. In the case of the AWID, we first implemented FFDNN models using the full set of features. Secondly, we conducted our experiments using a reduced set of features consisting of 26 features out of the 154 available attributes. The simulations showed that our proposed approach obtained overall accuracies of 99.66% and 99.77% for the binary and the multiclass classification respectively. These results represent an improved performance in contrast with the FFDNNs that used the full set of features. Additionally, the usage of the AWID in our research contributes significantly to the limited literature with regards to its analysis. Moreover, the utilization of the AWID in our research has demonstrated that our proposed method is applicable not only to wired networks but it can also be adapted for application to wireless networks.

In future work, we aim to investigate the detection rates of individual classes of the UNSW-NB15 and the AWID as well as the impact on the performance after applying a wrapper based feature extraction method.

Acknowledgment

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

References

[1] J. A. Nada, M. R. Al-Mosa, A proposed wireless intrusion detection prevention and attack system, in: 2018 International Arab Conference on Information Technology (ACIT), IEEE, 2018, pp. 1–5.

[2] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, S. Venkatraman, Deep learning approach for intelligent intrusion detection system, IEEE Access 7 (2019) 41525–41550.

[3] R. Samrin, D. Vasumathi, Review on anomaly based network intrusion detection system, in: 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), IEEE, 2017, pp. 141–147.

[4] T. Mehmood, H. B. M. Rais, Machine learning algorithms in context of intrusion detection, in: 2016 3rd International Conference on Computer and Information Sciences (ICCOINS), IEEE, 2016, pp. 369–373.

[5] I. E. N. I, M. Murphy, What is machine learning?, Machine Learning in Radiation Oncology 7 (2015) 3–11.

[6] A. A. Aburomman, M. B. I. Reaz, A novel svm-knn-pso ensemble method for intrusion detection system, Applied Soft Computing 38 (2016) 360–372.

[7] T. Yerong, S. Sai, X. Ke, L. Zhe, Intrusion detection based on support vector machine using heuristic genetic algorithm, in: 2014 Fourth International Conference on Communication Systems and Network Technologies, IEEE, 2014, pp. 681–684.

[8] S. Sahu, B. M. Mehtre, Network intrusion detection system using j48 decision tree, in: 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, 2015, pp. 2023–2026.

[9] M. Saber, I. El Farissi, S. Chadli, M. Emharraf, M. G. Belkasmi, Performance analysis of an intrusion detection systems based of artificial neural network, in: Europe and MENA Cooperation Advances in Information and Communication Technologies, Springer, 2017, pp. 511–521.

[10] I. A. Basheer, M. Hajmeer, Artificial neural networks: fundamentals, computing, design, and application, Journal of microbiological methods 43 (1) (2000) 3–31.

[11] N. Farnaaz, M. Jabbar, Random forest modeling for network intrusion detection system, Procedia Computer Science 89 (2016) 213–217.

[12] J. Yang, Z. Ye, L. Yan, W. Gu, R. Wang, Modified naive bayes algorithm for network intrusion detection based on artificial bee colony algorithm, in: 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), IEEE, 2018, pp. 35–40.

[13] W. Zhang, G. Yang, Y. Lin, C. Ji, M. M. Gupta, On definition of deep learning, in: 2018 World Automation Congress (WAC), IEEE, 2018, pp. 1–5.

[14] G. Tur, A. Celikyilmaz, X. He, D. Hakkani-Tür, L. Deng, Deep learning in conversational language understanding, in: Deep Learning in Natural Language Processing, Springer, 2018, pp. 23–48.

[15] J. Latif, C. Xiao, A. Imran, S. Tu, Medical imaging using machine learning and deep learning algorithms: A review, in: 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), IEEE, 2019, pp. 1–5.

[16] T. Ogunfunmi, R. P. Ramachandran, R. Togneri, Y. Zhao, X. Xia, A primer on deep learning architectures and applications in speech processing, Circuits, Systems, and Signal Processing (2019) 1–27.

[17] S. Kumar, A. Pandey, K. S. R. Satwik, S. Kumar, S. K. Singh, A. K. Singh, A. Mohan, Deep learning framework for recognition of cattle using muzzle point image pattern, Measurement 116 (2018) 1–17.

[18] Z. Yan, X. Song, H. Zhong, Spacecraft detection based on deep convolutional neural network, in: 2018 IEEE 3rd International Conference on Signal and Image Processing (ICSIP), IEEE, 2018, pp. 148–153.

[19] F. Gottwalt, E. Chang, T. Dillon, Corrcorr: A feature selection method for multivariate correlation network anomaly detection techniques, Computers & Security 83 (2019) 234–245.

[20] H. Liu, L. Yu, Toward integrating feature selection algorithms for classification and clustering, IEEE Transactions on Knowledge & Data Engineering (4) (2005) 491–502.

[21] S. S. Saha, S. Rahman, M. J. Rasna, T. B. Zahid, A. M. Islam, M. A. R. Ahad, Feature extraction, performance analysis and system design using the du mobility dataset, IEEE Access 6 (2018) 44776–44786.

[22] I. Manzoor, N. Kumar, et al., A feature reduced intrusion detection system using ann classifier, Expert Systems with Applications 88 (2017) 249–257.

[23] R. Panthong, A. Srivihok, Wrapper feature subset selection for dimension reduction based on ensemble learning algorithm, Procedia Computer Science 72 (2015) 162–169.

[24] R. Kohavi, G. H. John, Wrappers for feature subset selection, Artificial intelligence 97 (1-2) (1997) 273–324.

[25] W. Ke, C. Wu, Y. Wu, N. N. Xiong, A new filter feature selection based on criteria fusion for gene microarray data, IEEE Access 6 (2018) 61065–61076.

[26] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Machine learning 63 (1) (2006) 3–42.

[27] S. M. Kasongo, Y. Sun, A deep learning method with filter based feature engineering for wireless intrusion detection system, IEEE Access 7 (2019) 38597–38607.

[28] D. M. Farid, L. Zhang, C. M. Rahman, M. A. Hossain, R. Strachan, Hybrid decision tree and naïve bayes classifiers for multi-class classification tasks, Expert Systems with Applications 41 (4) (2014) 1937–1946.

[29] S. Learn, Naive bayes (2019).
URL https://scikit-learn.org/stable/modules/naive_bayes.html

[30] T. E. Schouten, E. L. Van den Broek, Fast exact euclidean distance (feed): A new class of adaptable distance transforms, IEEE transactions on pattern analysis and machine intelligence 36 (11) (2014) 2159–2172.

[31] Z. Abu-Aisheh, R. Raveaux, J.-Y. Ramel, Efficient k-nearest neighbors search in graph space, Pattern Recognition Letters.

[32] R. Gholami, N. Fakhari, Support vector machine: Principles, parameters, and applications, in: Handbook of Neural Computation, Elsevier, 2017, pp. 515–535.

[33] S. M. Kasongo, Y. Sun, A deep long short-term memory based classifier for wireless intrusion detection system, ICT Express.

[34] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, S. Venkatraman, Deep learning approach for intelligent intrusion detection system, IEEE Access 7 (2019) 41525–41550.

[35] I. Manzoor, N. Kumar, et al., A feature reduced intrusion detection system using ann classifier, Expert Systems with Applications 88 (2017) 249–257.

[36] Y. Chang, W. Li, Z. Yang, Network intrusion detection based on random forest and support vector machine, in: 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Vol. 1, IEEE, 2017, pp. 635–638.

[37] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, K. Han, Enhanced network anomaly detection based on deep neural networks, IEEE Access 6 (2018) 48231–48246.

[38] I. Benmessahel, K. Xie, M. Chellal, T. Semong, A new evolutionary neural networks based on intrusion detection systems using locust swarm optimization, Evolutionary Intelligence 12 (2) (2019) 131–146.

[39] S. A. Althubiti, E. M. Jones, K. Roy, Lstm for anomaly-based network intrusion detection, in: 2018 28th International Telecommunication Networks and Applications Conference (ITNAC), IEEE, 2018, pp. 1–3.

[40] S. Yang, Research on network behavior anomaly analysis based on bidirectional lstm, in: 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), IEEE, 2019, pp. 798–802.

[41] C. Khammassi, S. Krichen, A ga-lr wrapper approach for feature selection in network intrusion detection, computers & security 70 (2017) 255–277.

[42] M. Tavallaee, E. Bagheri, W. Lu, A. A. Ghorbani, A detailed analysis of the kdd cup 99 data set, in: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, IEEE, 2009, pp. 1–6.

[43] M. Tavallaee, E. Bagheri, W. Lu, A. A. Ghorbani, A detailed analysis of the kdd cup 99 data set, in: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, IEEE, 2009, pp. 1–6.

[44] J. Gao, S. Chai, B. Zhang, Y. Xia, Research on network intrusion detection based on incremental extreme learning machine and adaptive principal component analysis, Energies 12 (7) (2019) 1223.

[45] N. Moustafa, J. Slay, The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set, Information Security Journal: A Global Perspective 25 (1-3) (2016) 18–31.

[46] A.-H. Muna, N. Moustafa, E. Sitnikova, Identification of malicious activities in industrial internet of things based on deep learning models, Journal of Information Security and Applications 41 (2018) 1–11.

[47] J. Ran, Y. Ji, B. Tang, A semi-supervised learning approach to ieee 802.11 network anomaly detection, in: 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring), IEEE, 2019, pp. 1–5.

[48] M. E. Aminanto, K. Kim, Improving detection of wi-fi impersonation by fully unsupervised deep learning, in: International Workshop on Information Security Applications, Springer, 2017, pp. 212–223.

[49] F. D. Vaca, Q. Niyaz, An ensemble learning based wi-fi network intrusion detection system (wnids), in: 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), IEEE, 2018, pp. 1–5.

[50] N. Moustafa, J. Slay, Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set), in: 2015 military communications and information systems

conference (MilCIS), IEEE, 2015, pp. 1–6.
[51] C. Kolias, G. Kambourakis, A. Stavrou, S. Gritzalis, Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset, IEEE Communications Surveys & Tutorials 18 (1) (2015) 184–208.
[52] C. Kolias, V. Kolias, G. Kambourakis, Termid: A distributed swarm intelligence-based approach for wireless intrusion detection, International Journal of Information Security 16 (4) (2017) 401–416.
[53] W. Zhang, T. Du, J. Wang, Deep learning over multi-field categorical data, in: European conference on information retrieval, Springer, 2016, pp. 45–57.
[54] Y. Li, Y. Yuan, Convergence analysis of two-layer neural networks with relu activation, in: Advances in Neural Information Processing Systems, 2017, pp. 597–607.
[55] M. M. Lau, K. H. Lim, Investigation of activation functions in deep belief network, in: 2017 2nd international conference on control and robotics engineering (ICCRE), IEEE, 2017, pp. 201–206.
[56] M. Puig-Arnavat, J. C. Bruno, Artificial neural networks for thermochemical conversion of biomass, in: Recent Advances in Thermo-Chemical Conversion of Biomass, Elsevier, 2015, pp. 133–156.
[57] S. Learn, Scikit-learn: Machine learning in python (2019). URL https://scikit-learn.org/stable/
[58] S. Mittal, S. Vaishay, A survey of techniques for optimizing deep learning on gpus, Journal of Systems Architecture (2019) 101635.

Sydney Mambwe Kasongo received a master's degree (M.Tech.) in Computer Systems from the Tshwane University of Technology in 2017. He is currently pursuing a Ph.D. degree in Electrical and Electronic Engineering at the University of Johannesburg. His main research interests include Machine Learning, Deep Learning, Computer Networks Security, Wireless Networks, and Data Science.

Yanxia Sun got her joint qualification: D-Tech in Electrical Engineering, Tshwane University of Technology, South Africa and PhD in Computer Science, University Paris-EST, France in 2012. Yanxia Sun is currently working as Associate Professor/Head of 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. Currently she is supervising six PhD students and four master 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 (SAYAS). Here research interests include Renewable Energy, Evolutionary Optimization, Neural Network, Nonlinear Dynamics and Control Systems

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.