AN IMPROVED CYBERATTACK PREDICTION TECHNIQUE WITH INTELLIGENT CLUSTERING AND DEEP NEURAL NETWORK

Supported by tetfund

Ayei E. Ibor[1]*, Florence A. Oladeji[2], Olusoji B. Okunoye[2], and Obeten O. Ekabua[1]
[1]Department of Computer Science, University of Calabar, Calabar, Nigeria
[2]Department of Computer Sciences, University of Lagos, Lagos, Nigeria
*Corresponding author: ayei.ibor@gmail.com

Received: October 27, 2019 **Accepted:** January 11, 2020

Abstract: One of the most damaging security threats on the Internet today is cyberattacks. As new paradigms emerge, new vulnerabilities and flaws are discovered on a daily basis. These vulnerabilities have been consistently exploited by malicious users to stage cyberattacks, which erode the confidentiality, integrity and availability of critical data, and other computing resources. In recent times, the research focus has been on signature based and anomaly detection approaches. However, the challenges of using known attack signatures and profiles have made the prediction of attacks an elusive and cumbersome activity. The use of task specific algorithms has also created more setbacks in cyberattack prediction, hence the need for new approaches that exploit the learning of data representations. Therefore, this paper presents a combination of Principal Component Analysis (PCA) and Expectation Maximization (EM) for intelligent clustering, and a supervised Deep Neural Network (DNN) for the training of the model to make predictions on attack data. In the hybrid model, PCA and EM algorithm perform dimensionality reduction and clustering of the attack data during the unsupervised pre-training stage of the model building. The output of the unsupervised pre-training is fed into the DNN for supervised training, at which point rectified linear units (RELUs) in the hidden layers are used to generate a cascade of concepts for making accurate predictions on the modeled dataset. For experimentation, we use a Python environment test bed to fully assess the performance of the model and report its accuracy, false positive rate, precision rate, recall rate, F-measure and entropy. The results obtained show a 99.8% accuracy for predicting the modeled attack types.

Keywords: Cyberattacks, cybersecurity, cyberattack prediction, expectation-maximisation, deep learning, PCA

Introduction

Cyberattacks are becoming more pervasive as new paradigms emerge and big data becomes more accessible. With an increase in the use of the Internet as devices and objects are configured to seamlessly connect to each other, the attacker is finding new ways to deliver malicious payloads to network perimeters and their internal topologies. Attack surfaces are expanding every year and the ability of most attackers to craft packets that evade network defences is gaining a new momentum. Tobiyam *et al*. (2016), Pai *et al*. (2017), Asaju *et al*. (2017) posit that malicious network traffic is affecting millions of resources as many of the attacks involve malware and other forms of attacks such as denial of service and probe.

Recent researches in machine learning have been helpful in predicting attacks, however, the use of task specific algorithms limits the extent to which predictions can be made hence the need for representation learning. With representation learning as discussed in Goodfellow *et al*. (2016), the intrinsic features of connection vectors can be extracted to have a cascade of concepts for the interpretation of different attack scenarios. In this way, the accuracy of cyberattack prediction can be enhanced. To achieve the proposed technique, a hybrid technique that combines unsupervised and supervised learning is used.

The model in this report uses Principal Component Analysis (PCA) and Expectation Maximisation (EM) algorithm for unsupervised learning. PCA participates in feature selection at the first phase of dimensionality reduction. This generates a set of uncorrelated principal components while maintaining the variability in the data. EM generates clusters from the dimensionally reduced dataset. The clusters are used to train the Supervised Deep Neural Network (DNN) for making predictions on the modeled dataset.

In the hidden layers of the DNN, the transformation of the feature space is enhanced to realise a structured interpretation of network traffic. Through this process, diverse attack scenarios can be predicted using learned attack patterns. Further deviations from the learned attack patterns can be flagged as new patterns to predict novel attacks. The model is evaluated for accuracy, false positive rate, precision rate, recall rate, F-measure and entropy using NSL-KDD dataset on a python environment test bed.

Advances in machine learning have yielded tremendous results in the field of deep learning. Deep learning architectures basically rely on non-linear activation functions on the hidden layers. Längkvist *et al*. (2014) posits that the notion of non-linearity creates a model that is able to learn more abstract representations of the feature space. That is, the lower layers are used for compressed feature representation, and higher layers are used to learn these representations for better generalisation of the feature space (Deng and Yu, 2014; LeCun *et al*., 2015).

In deep learning, a composition of many layers is used to define parameterised functions such as sigmoid and rectified linear units (RELUs) from inputs to outputs (Abadi *et al*., 2016). These parameterised functions are subsequently trained such that we can fit any finite set of input and output examples. A loss function is also defined to represent the cost of mismatching on the training data. Furthermore, Cho (2014) and Goodfellow *et al*. (2016) give the conditions for a deep neural network as follows:

i) We can extend the network by adding layers made up of multiple units
ii) In each and every layer, the parameters are trainable.

Tobiyama *et al*. (2016) mentioned that a deep neural network (DNN) is a neural network with several hidden layers. DNN typically learns data representations rather than perform task specific functions. In learning data representations, DNN relies on several layers of non-linear information processing. These layers can be adapted for supervised or unsupervised automatic feature learning and abstraction on several architectures such as deep neural networks, deep belief networks and recurrent neural networks (Deng and Yu, 2014; LeCun *et al*., 2015; Schmidhuber, 2015). In Tobiyama *et al*. (2016), a stepwise application of deep neural network is used to classify malware processes. The authors combined the effect of RNN and Convolutional Neural Network (CNN) to extract and classify features of malware process behaviour to report the presence or absence of a malware in a network. The

approach achieved an accuracy of 96% in detecting malware though it used 150 instances of process behaviour log files. It is yet unclear how the model will behave given a larger dataset.

Wei *et al*. (2016) applied a deep learning model to mitigate injection attacks in smart grids. The technique could identify and mitigate information corruption on Wide Area Monitoring Systems (WAMSs). Belanger and McCallum (2016) studied the use of deep learning architectures to perform representation learning for structured objects. The method was able to make predictions using gradient descent on multi-label classification tasks.

Shen *et al*. (2018) presented an attack prediction approach called *Tiresias xspace*. This technique uses recurrent neural networks (RNN) to predict the likelihood of future attacks on a host machine using previous observations. Nguyen *et al*. (2018) presented an approach that used deep learning to detect and isolate cyberattacks in mobile clouds, achieving an accuracy of 97.11%. Rhode *et al*. (2018) used recurrent neural networks (RNNs) to predict the state of an executable code as either malicious or benign. The model used a short snapshot of behavioural data and achieved a 94% accuracy within the first five seconds of execution and an accuracy of 96.01% during the first twenty seconds of execution on unseen test set. Nevertheless, the RNN performed poorly for detecting malware at a family level when not initially trained on it.

Diro and Chilamkurti (2018) used a deep learning approach to detect attacks in social Internet of Things (IoT). The distributed attack detection approach is based on IoT/Fog network that uses a master node for collaborative parameter sharing and optimisation. The approach accelerates data training near the source of the attack and achieved an accuracy of 98.27%. Furthermore, Loukas *et al*. (2018) proposed an attack detection system for vehicles using a combination of deep multilayer perceptron and recurrent neural network architecture. The approach used data captured in real-time for both cyber and physical processes. This data served as input to a neural network architecture in the form of time series data. Experiments were conducted based on denial of service, command injection and malware attacks with an accuracy of 86.9% achieved.

Also, a deep learning approach for feature learning and dimensionality reduction was proposed in Al-Qatf *et al*. (2018). The model was able to decreasetraining and testing time, and couldenhance the attack prediction accuracy of SVM. Unsupervised pre-training with sparse autoencoder was used to buildthe model, and the transformed feature space was fed into the SVM algorithm to detect attacks. A good detection accuracy was reported for the KDD99 and NSL-KDD datasets. In addition, Rezvy *et al*. (2019) useda deep autoencoded dense neural network algorithm to detect attacks on Fifth Generation (5G) and IoT networks. The proposed approachdemonstrateda 2-step detection approach with deep autoencoders used for unsupervised pre-training to reduce high dimensional data to low-dimensional representation. In the second stage, the approach performed supervised classification with a deep neural network to achieve good performance with an accuracy of 99.9%.

Vinayakumar *et al*. (2019) presented an approach called scale-hybrid-IDS-AlertNet. This approach can be applied to the monitoring of network traffic in real time in order to indicate the presence of anomalies representing attacks in network traffic. Scale-hybrid-IDS-AlertNetuseddistributed and parallel machine learning algorithms with a diversity of optimisation techniques for handling a huge number of network and host-level events.

A technique that pooled the effect of improved Genetic Algorithm (GA) and Deep Belief Network (DBN) to develop an adaptive model for detecting attacks on IoT was studied in Zhang *et al*. (2019). For the experimentation, NSL-KDD dataset was used to simulate and evaluate the modelto recognise attacks. An accuracy of 99.45% for DoS attacks was achieved. In the GA-DBN model, GA was used to select an optimal network structure through multiple iterations on the attack dataset. The DBN then deploys the optimal network structure for the classifying of attacks thus enhancing the classification accuracy.

Unsupervised pre-training

This section will provide an insight into the relevance of the preliminary stage of unsupervised pre-training in the proposed model. Since this approach is based on a deep feedforward network that leverages structured data with entity embedding, it is significant to solve the problem of spontaneous classification for the existing statistical variance on a large dataset. This is relevant for enhancing the extraction of useful information from a large amount of samples.

Most extant prediction approaches find it difficult to cope with this variance due to statistical noise resulting in intermittent failures at different timestamps, at which points attacks can easily infiltrate a network. What is required, therefore, is the ability to abstract the noise away in order to capture the useful information from the large attack dataset. This necessitates the use of an unsupervised pre-training process based on Principal Component Analysis (PCA) and Expectation Maximisation (EM) clustering. Prior to feeding the neural network with the attack dataset, it is important to remove redundant data and also convert the data to numeric feature vectors. Initially, PCA is used to perform a preliminary compression of the feature space to serve as input into a latent space representation as opined by Vasan and Surendiran (2016). PCA automaticallysearches for the principal components relevant for expressing the intrinsic information in the data. That is, it performs feature selection, in which case it transforms the data space into a feature space that retains the same dimension as the original data.

In dimensionality reduction with PCA, the number of features required for the effective representation of data can be reduced by eliminating linear combinations with small variances while retaining terms with large variances. This is achieved by computing the largest $\boldsymbol{k}$ eigenvalues of the correlation matrix $\boldsymbol{R}$ ((Jolliffe and Cadima, 2016; Goodfellow *et al*., 2016).

In reality, PCA performs a linear projection from the data space (say R^i to the feature space (R^k), resulting in an approximate representation of the input data vector. De la Hoz *et al*. (2015) claims that this encoding process produces a vector of principal components, which has a more reduced feature space than the original dataset. In the same sense, the original data vector can be reconstructed from the vector of principal components through decoding. Here, a linear projection from (R^k) to (R^i) is performed. That is, a mapping is performed from the feature space to the data space to reconstruct the original input vector.

In the next stage of the unsupervised pre-training procedure, the expectation maximisation (EM) algorithm is used. The features produced at the previous level of the pre-training with PCA are taken as inputs to the EM algorithm. The resulting clusters are used as initialisation to the deep supervised neural network. The EM finds maximum likelihood parameters of a statistical model where the equations cannot be solved directly. In the actual sense, these models include latent variables in addition to unknown parameters and known data observations (Kishor and Venkateswarlu, 2016).

The choice of the EM algorithm in this work is based on its flexibility with cluster covariance. To this effect, and due to the standard deviation parameters, the clusters can be of any

elliptical shape. The EM algorithm also uses probabilities and as such can have multiple clusters per data point. Assuming a data point, say $x_i$falls between two overlapping clusters, its class can be defined by stating that x_i belongs y-percent to class A and z-percent to class B. In this way, mixed membership is supported by the EM algorithm (Do and Batzoglou, 2008).

The goal of using pre-trained weights for the deep neural network rather than random initialised weights is to optimise the performance of the supervised deep learning predictor. The starting point in parameter space is a significant factor for a better optimisation and generalisation of the problem space. Using random inputs may likely propagate the error across the multi-layer structure of the network. Erhan *et al*. (2010) mentioned that the non-linearity of the layers of the deep neural network yields an error surface that is non-convex and difficult to optimise. With a possible resulting local minima, the unsupervised pre-training procedure is targeted at yielding better generalisation or test error that leads to a better performance of the predictor.

Our Approach

With the limitations of task specific algorithms and ongoing researches in deep learning, it is imperative to develop new models for predicting attacks as attack surfaces escalate. There is no single approach we can rely on since the vastness of the cyberspace portends new challenges at varying timestamps arising from the emergence of new paradigms and disruptive technologies. Subsequently, in this report, a combination of techniques is used to realise an efficient model for predicting cyberattacks.

First, the alerts collected from different sources are filtered to discriminate attack vectors from normal connection vectors. Then, the connection vectors undergo two learning processes. First, unsupervised learning is used to perform preliminary dimensionality reduction and clustering. At the second stage, supervised deep learning is used to train the model for making predictions on test data. The model uses a feed forward deep neural network (DNN) with n-hidden dense layers and a Softmax layer for classifying network attacks into one of the classes of DoS, Probe, R2L, U2R, and Normal connection vectors. The entre prediction process is modeled as a multi-label classification problem.

Architecture of the Proposed Approach

The architecture of the proposed approach is depicted in Fig. 1. As shown in the model, network traffic goes through a filter that discriminates potential alerts and directs them to an alert database while conceived benign traffic is directed towards the normalisation module. The potential attack vectors, which are indicated in the alert database are then sent to the normalisation module to undergo two basic processes. These include the replacement of missing values in the modeled dataset and Binarisation. At the completion of these processes, the captured traffic undergoes initial dimensionality reduction, in which case, PCA is used to scale the dataset to a set of ***p*** uncorrelated principal components (Jolliffe and Cadima, 2016).

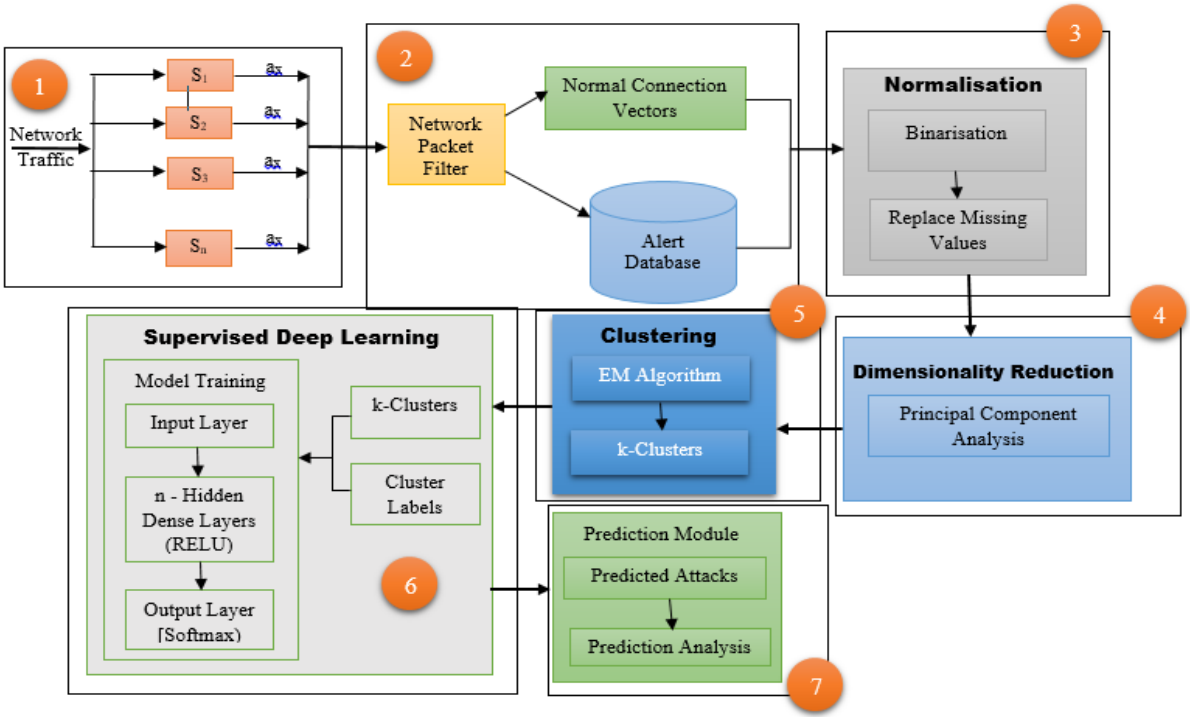

Fig. 1: Architecture of the ensemble technique

To enhance the training of the model and prediction accuracy, the reduced feature space is clustered using Expectation Maximisation (EM) algorithm to generate a set of ***k***-clusters. These ***k***-set of clusters represent hyper-alerts with labels generated automatically by the EM clustering algorithm. The cluster labels determine the class to which a certain connection vector belongs, which can be an attack or normal class.

For the training of the model, a supervised DNN is used. The model is trained per cluster using a number of ***n***-epochs, ***n*** is chosen to avoid overfitting. The output of the model is the predicted attack classes. Furthermore, the analysis of the prediction is based on performance metrics such as accuracy (ACC), false positive rate (FPR), precision rate (PR), recall rate (RR), F-measure (F) and entropy (E). The components of the model are described in subsequent sections.

The components of the model are described in subsequent sections.

I. **Network traffic capture:** The first component represents the capture of network traffic from different sources across the network perimeter. Each instance of

network traffic capture is a connection vector. We define a vector of features for each connection as:

$$v_n = \{f_1, f_2, \dots, f_m\} \quad (1)$$

Where f_m is the number of features in the connection vector

II. **Network traffic discrimination:** At this stage, an attack or normal connection vector is identified using a feature vector. This feature vector consists of 41 features, some of which include protocol type, service, flag, source and destination bytes. The feature vector depicting an alert is stored in an alert database, and other feature vectors are combined with the flagged alerts for normalisation.

III. **Normalization:** To achieve an error-free prediction, the captured alerts and non-alerts data are normalised. Missing values are replaced and categorical values are identified and converted to nominal values.

IV. **Dimensionality reduction:** At stage 4, dimensionality reduction is performed on the dataset to generate a set of ***p*** uncorrelated principal components from the correlated connection vectors. PCA is used for this purpose to reduce the feature space while still maintaining the variability in the dataset. Given the dataset, D, with ***n***-instances and ***p*** features or variables, PCA generates $min(n-1|, p)$ distinct principal components from which the target output can be reconstructed That is,

$$d = D(min(n-1), p) \quad (2)$$

V. **Clustering:** The compressed dataset, ***d***, is used to automatically generate ***k***-clusters with the EM algorithm. Clustering is relevant, in this context, to enhance the training of the model by automatically categorizing attack data. This can be very helpful in the early stages of an attack. The EM algorithm performs parameter estimation in probabilistic models even when the data is incomplete. In other words, the EM algorithm measures the distances between data points based on probability distributions. Kishor and Venkateswarlu (2016) and Pai *et al.* (2017) mention that these distributions are re-estimated at each step of the algorithm's 2-step iterative process.

EM achieves clustering by initialising the mean and variance as the parameters for ***k*** probability distributions. The algorithm then alternates between the 2-step iterative processes as follows:

a) **E Step:** the probabilities required in the M Step are computed using the current estimates of the distribution parameters

b) **M Step:** the distribution parameters with respect to maximum likelihood estimators are then recomputed using the probabilities from the E Step.

The shape of the cluster changes as these parameters are recomputed iteratively until the ***k***-clusters are generated. Therefore, if we represent the EM algorithm as θ, we have:

$$d_k = \theta(d)_k = \theta(x, y) \quad (3)$$

Where; d_k is the clustered dataset by applying the EM algorithm, θ on ***d***, ***k*** represents an automatically generated number of clusters on ***d***. Since the dataset is 2-dimensional with the instances as a matrix, ***x[n, m]***, and the class labels as a vector, y, fitting ***x*** and ***y*** into the EM algorithm will generate a function$\theta(x, y)$, to match the instances (data points) to the class labels prior to input to the DNN. The k^{th} cluster in d_k is represented as μ^k.

VI. **Supervised deep learning:** At the DNN, the model is trained using the constructed ***k***-clusters and cluster labels, generated with the EM algorithm. The DD is a Feed Forward (FF) DNN with 5 layers (1 input layer, 3 hidden dense layers $(h^i, 1 \le i \le 3)$, and 1 output layer). The FF DNN is trained with the ***k***-clusters fed into the input layer of 1000 units. In the hidden dense layers of 750, 500, and 250 units respectively, the model learns the data representation of the different attack types using the input (***k***-clusters). The DNN also performs secondary dimensionality reduction for feature abstraction at this stage. In the output, there are 5 units depicting the modeled attack types and normal connection.

Rectified linear units (ReLU) are used in the hidden layers of the model as the activation function. In the process of classifying an attack type, y_k, the model learns the function:

$$f(\mu) = max(\mu, 0) \quad (4)$$

The output of $f(\mu)$ is 0 for $k < 0$, otherwise the output is equal to the input, which is an approximation to the identity function. That is, the model will output y_k that is equivalent to μ^k. Some constraints are used on the DNN to enhance the learning of the identity function as discussed in Agarap (2018). These constraints include placing a limit on the number of hidden layers that represent a cascade of concepts for developing feature representations. This is essential to discover patterns in the data for predicting the modeled attack types and also avoid such challenges as overfitting. The DNN is shown in Fig. 2.

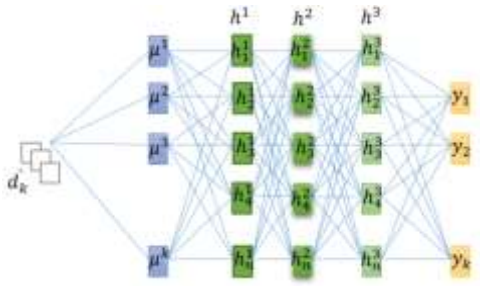

Fig. 2: The feed forward deep neural network of the model

VII. **Prediction module:** At the hidden layers, the network can learn a compressed representation of each cluster, μ^k. The Softmax function is used to classify this compressed representation at the output layer. With the Softmax function, the output is partitioned such that the total sum is 1, an equivalent of a categorical probability distribution (Agarap, 2018). In this sense, the final layer consists of one neuron for each of the attack classes. Each attack class returns a value between 0 and 1, which is inferred as a probability. This results in an output with a probability that sums to 1.

The probability of an attack, is computed by applying the Softmax function to each cluster class value, that is:

$$\acute{y} = Softmax(y_k) = \frac{e^{y_k}}{\sum_{k=1}^{n} e^{y_k}} \quad (5)$$

$\acute{y}$ is the predicted class. A standard categorical cross-entropy loss function is used at the output layer. The model is trained with an initial learning rate of 0.1 and optimized using the stochastic gradient descent (SGD) algorithm. Predictions are made using equation (5), and the range of $\acute{y}$, which is (0, 1) indicates the accuracy of predictions. To validate the model, a value of $0.8 \le \acute{y} \le 1$ indicates very high performance while $\acute{y}$<0.5 is indicative of a poor predictor. For $0.5 \le \acute{y} < 0.8$, an

average performance is reported. The ranges of values for validating the model are given in Table 1.

Table 1: Range of $\acute{y}$ for validating the model's performance

Range of $\acute{y}$	Model Performance
$0.8 \leq \acute{y} \leq 1$	High (acceptable)
$0.5 \leq \acute{y} < 0.8$	Average
$\acute{y}$<0.5	Poor

The predictions of the model are analysed using a confusion matrix. From the confusion matrix, the following performance metrics are computed (Milenkoski *et al*, 2015):

i) **Accuracy of prediction (ACC):** the rate of instances of attacks or normal connections predicted correctly. This is calculated as:

$$ACC = \frac{TP+TN}{TP+TN+FN+FP} \quad (6)$$

Where, TP is True Positive: correct positive prediction; TN is True Negative: correct negative prediction; FN is False Negative: incorrect negative prediction; FP is False Positive: incorrect positive prediction.

ii) **False positive rate (FPR)**: the rate of instances of attacks predicted as normal connections or vice versa denoted by:

$$FPR = \frac{FP}{TN+FP} \quad (7)$$

iii) **Precision rate (PR)**: the fraction of relevant instances in the dataset given as:

$$PR = \frac{TP}{TP+FP} \quad (8)$$

iv) **Recall rate (RR)**: the retrieved relevant instances over the total amount of relevant instances. RR calculated as shown in equation (7):

$$RR = \frac{TP}{TP+FN} \quad (9)$$

v) **F-Measure (F-Score or F1)**: a measure of the accuracy of the model computed as the weighted harmonic mean of the precision and recall of the model. F-measure is denoted by:

$$F1 = 2x\frac{PR.RR}{PR+RR} \quad (10)$$

vi) **Cross Entropy (E)**: a measure of the performance of a classification model whose output is a probability value between 0 and 1. That is,

$$E = -\sum_{i=1}^{n} y_i log(\acute{y}_i) \quad (11)$$

In this case, ***n*** is the number of classes, ***y*** is the true class value and $\acute{y}$ is the predicted class value. A good model will have E that is 0 or close to 0. The consideration of the value of E is used to assess the efficiency of the model, i.e. E <0.15 is used as the benchmark for determining good performance by the model.

The accuracy of prediction is interpreted by comparing each output from the Softmax layer with its corresponding true value. That is, the true values are one-hot-encoded such that a value of one (1) appears in the column corresponding to the correct attack class, otherwise a value of zero (0) is shown (Montavon *et al*., 2018).

Data preparation

The model is validated using the NSL-KDD dataset. This dataset has 41 features with a large number of connection vectors labelled as either normal or a specific attack type. The NSL-KDD dataset is an enhanced and reduced version of KDDCup'99 dataset. It contains 22 attack types in the training set and 37 attack types in the test set. The general classes of attacks in the dataset are probe, denial of service (dos), remote to local (r2l), and user to root (u2r) attacks (Dhanabal and Shantharajah, 2015). Twenty percent (20%) of the original NSL-KDD dataset (with 25, 192 connection vectors) is used for the training and testing of the model. A test split of 30% of the original dataset is used for the validation of the model. The number of connection vectors in the dataset is given in Table 2.

Table 2: Summary of the number of connection vectors used for the experiments

Connection Vector	Number of Instances	% of Total
Normal	13, 449	53.39
Denial of Service (DoS)	9,234	36.65
Probe	2,289	9.09
Root to Local (R2L)	209	0.83
User to Root (U2R)	11	0.04
Total	25, 192	100.00

Feature ranking

The model was trained using an automatically generated number of clusters (***k***-clusters) from the set of 41 features in the NSL-KDD dataset. The ***k***-clusters were formed from a dimensionally reduced dataset, ***d***, using PCA. PCA generated ***p***-principal components, representing a compressed feature space as mentioned in Vasan and Surendiran (2016).

Deep learning can perform optimally with a few features based on its representation learning structure. Consequently, PCA was vital to enhancing the selection of the features that contribute to the representation of the internal structure of the data. With PCA, the variance in the data was optimised to generate the representative subset of features for training the model.

The model is trained using 70% of the 25,192 instances used and was able to generalize to an unknown dataset while deriving an accurate estimate of model prediction performance.

Testbed of the experiment

The DNN is implemented using a TensorFlow backend in Python 3.6 on an Ubuntu 18.04 64-bit operating system with. Keras and ScikitLearn libraries (Abadi *et al*., 2016; Gulli and Pal, 2017; Hackeling, 2017; Ketkar, 2017). TensorFlow is a symbolic math library for machine learning applications such as neural networks. As discussed in Abadi *et al*. (2016) and Ramsundar and Zadeh (2018), TensorFlow computations are expressed as stateful dataflow graphs for high performance numerical computations across a variety of platforms (CPUs and GPUs). These stateful graphs allow neural networks to perform operations on multidimensional data arrays composed of scalars, vectors and matrices (Wongsuphasawat *et al*., 2018).

Keras, which is an open-source neural network library runs on the TensorFlow backend to enable a fast implementation of the deep neural network. At the same time, Scikit-learn, which is used for machine learning contains a collection of classification, regression and clustering algorithms. This software machine learning library also interoperates with the Python numerical and scientific libraries called NumPy and SciPy (Gulli and Pal, 2017; Hackeling, 2017). The system properties of the machine used for experimentation are shown in Table 3.

Table 3: System properties of the implementation machine

Host Operating System	Ubuntu 18.04
Processor	Intel ® Core ™ i3 6100U CPU @2.30 GHz 2.30GHz
RAM	4.00GB
System Type	64-bit Operating System, x-64 based processor

Experimental Results and Discussion

The model is trained over 100, 200, and 500 epochs respectively. For each training period, the performance of the model was recorded using such metrics as accuracy, recall rate, precision rate, F-measure and cross entropy. The results obtained by the model at the end of the train and test phases of the implementation are shown in Table 4. The summary of the model's performance is based on 5 class labels (normal, dos, probe, r2l, and u2r).

As shown in Table 4, the model shows improved performance as the number of epochs increased. This implies that the model is able to learn more abstracted features of the dataset during the training phase at each layer to make better generalisations for predicting the modeled attack types while minimising the cross entropy loss and the false positive rate.

Using Figs. 8, 9 and 10, the accuracy and cross entropy loss of the model are depicted to provide visualisations of the evaluation of the model. These visualisations were produced by plotting the accuracy and cross entropy loss of the model against the number of epochs during the training and testing phases of the experimentation. For all cases, the model shows an improvement in the training phase over the number of iterations (epochs) used.

Table 4: Results of model's performance over 100, 200, and 500 epochs

Epochs Metrics	100	200	500
ACC	99.83730217423458	99.88170930060623	99.8963423663557
RR	99.87586891757697	99.95033523714925	99.92537313432835
FPR	0.0021953896816684962	0.0021929824561403508	0.0014635931211123307
PR	99.85107967237528	99.85115355991068	99.90052225814475
F-Measure	99.86347275660916	99.90071978158352	99.91294615097624
Cross Entropy	0.0014880938621971204	0.0014873560877253463	0.000994282463345708

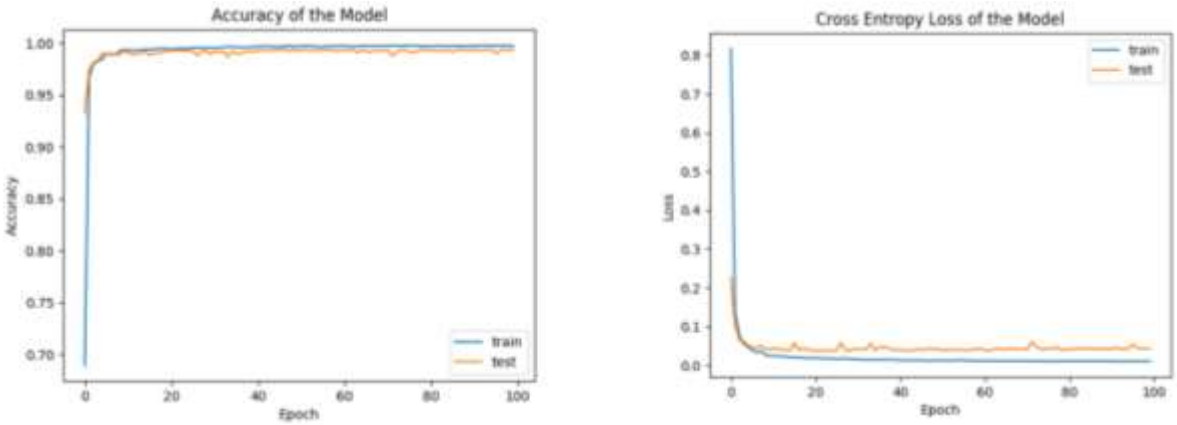

Fig. 8: Accuracy and cross entropy loss of the model for 100 epochs

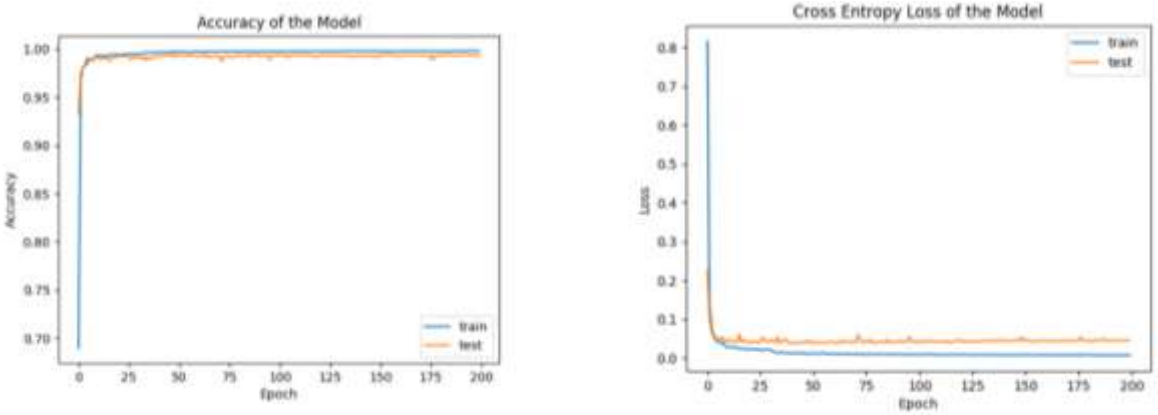

Fig. 9: Accuracy and cross entropy loss of the model for 200 epochs

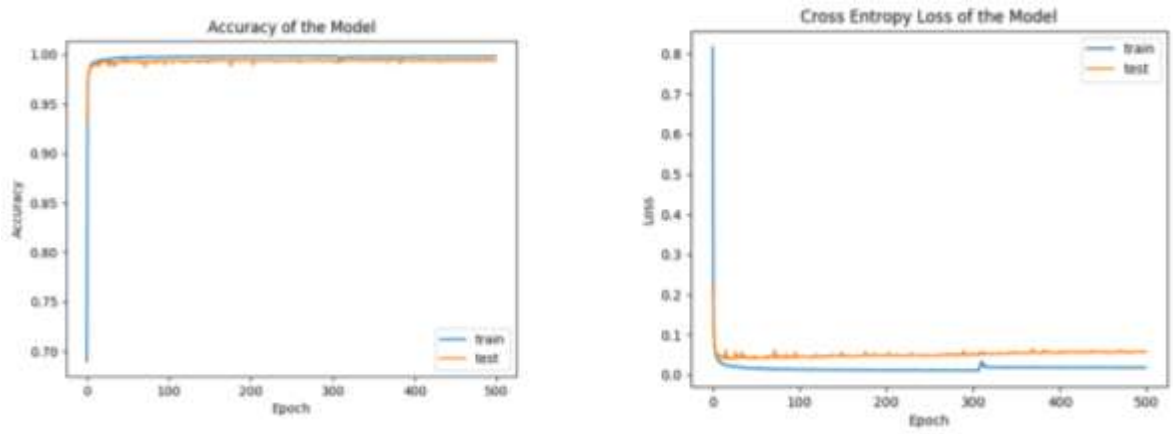

Fig. 10: Accuracy and cross entropy loss of the model for 500 epochs

As shown in Fig. 8, a prediction accuracy of 99.837% was achieved by the model over 100 epochs with an entropy loss of 0.00148809. It can be seen that as the accuracy goes to 1 (or 100%), the entropy loss tends to 0 indicating a very good model. This accuracy improves to 99.882% over 200 epochs while a decrease in the entropy loss (0.00148735) is also recorded as illustrated in Fig. 9. With the model showing the tendency to improve through the results already obtained, further experiments were conducted over 500 epochs. The results obtained show an improvement of 0.0146%, that is, at 500 epochs, a prediction accuracy of 99.896% was achieved, which is higher than previous results. Similarly, a cross entropy loss of 0.00099428 was obtained. This is represented in Fig. 10.

In each layer of the DNN, an abstracted representation of the data from a preceding layer is used as its current input, which it learns from. This learning process allows the model to make predictions using test data split from the given dataset. The test plots also illustrate good performance by the model, that is, the model gneralises to test data to make accurate predictions.

From Table 4, the false positive rates (FPR) for the three levels of iterations (i.e. 100, 200 and 500) were extremely low. An FPR value is low when it goes towards zero (0). A high FPR value implies a poor predictor while a low FPR value indicates a very good predictor as achieved by the model in this research. The lowest FPR value of 0.001463593 was achieved over 500 epochs, indicating a point at which the model is stable. Conversely, the precision and recall rates as well as the F-Measure values from the experiments were above 99%, thus validating the efficiency of the ensemble model.

In this approach, a Softmax function is used at the output layer with classification probability $\acute{y}$delivered by equation (9). The range of $\acute{y}$is (0, 1), equivalently (0, 100), thus the model demonstrates more than 99% prediction accuracy as depicted by the test plots of Figs. 8, 9 and 10. Similarly, the cross entropy loss of the model indicates a good classifier. FPR values, which are used as prediction errors of the model were minimal, implying that only a few instances of the attack data were misclassified or predicted.

Comparison of Results

The results of experimentation of the proposed model will be benchmarked against extant state-of-the-art approaches in the field of deep learning. There is recent research focus on deep learning models for cyberattack and malware detection, classification and prediction. While every system comes with inherent limitations as no system can be 100% efficient, there is the need to appraise minor significant improvements in the implementation of a given system to ascertain its relevance in the context of use.

Consequently, the performance of the proposed model is benchmarked against the works of Tobiyama *et al*. (2016), Rhode *et al*. (2018), Nguyen *et al*. (2018), Diro and Chilamkurti (2018). The results of comparison are given in Table 5.

Table 5: Performance comparison of the proposed model with extant State-of-the-art Approaches

Approach	Accuracy	False Positive Rate
Proposed Approach	99.8%	0.00146
Tobiyama *et al*. (2016)	96%	
Rhode *et al*. (2018)	96.01%	3.17
Nguyen *et al*. (2018)	97.11%	2.89
Diro & Chilamkurti (2018)	98.27%	2.57

The comparison in Table 5 shows that the proposed model can perform well in the prediction of cyber-attacks. A very high accuracy of 99.8% and FPR of 0.00146 shows that the model outperforms similar models. The ensemble prediction approach is therefore fit for purpose in the prediction of cyberattacks.

Conclusion

In this report, an ensemble technique for predicting cyberattacks is introduced. To predict attacks, it is significant to use representation learning to tune the parameters of a model rather than depending on task specific algorithm. Deep learning performs representation learning by extracting and abstracting features in order to perform non-linear transformation of the input data to deliver compressed feature space. The model learns the features to optimally choose at each layer for enhance classification of the multi-label problem. The participation of PCA and the EM algorithm at the initial stage improves the learning process of the deep neural network through cluster-based training. The model is evaluated using NSL-KDD dataset. With NSL-KDD dataset having a large set of connection vectors, the model is tuned to learn different attack types to make accurate predictions. The results obtained show a 99.8% prediction accuracy for the modeled attack types.

Conflict of Interest

Authors declare that there is no conflict of interest related to this study.

References

Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, ... & Kudlur M 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on*

Operating Systems Design and Implementation ({OSDI} 16) (pp. 265-283).

Agarap AF 2018. Deep Learning using Rectified Linear Units (ReLU). *arXiv preprint arXiv:1803.08375*.

Al-Qatf M, Lasheng Y, Al-Habib M & Al-Sabahi K 2018. Deep learning approach combining sparse autoencoder with SVM for network intrusion detection. *IEEE Access*, 6: 52843-52856.

Asaju LAB, Shola PB, Franklin N & Abiola HM 2017. Intrusion detection system on a computer network using an ensemble of randomizable filtered classifier, K-nearest neighbor algorithm. *FUW Trends in Sci. and Techn. J.*, 2(1B): 550 – 553.

Belanger D & McCallum A 2016. Structured prediction energy networks. *In International Conference on Machine Learning*, pp. 983-992.

Cho K 2014. Foundations and advances in deep learning taxonomy, and future directions. *Computer Communications*, 107: 30-48.

De la Hoz E, De La Hoz E, Ortiz A, Ortega J & Prieto B 2015. PCA filtering and probabilistic SOM for network intrusion detection. *Neurocomputing*, 164: 71-81.

Deng L & Yu D 2014. Deep learning: Methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4): 197-387.

Dhanabal L & Shantharajah SP 2015. A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. *Int. J. Advanced Res. Comp. and Commun. Engr.*, 4(6): 446-452.

Diro AA & Chilamkurti N 2018. Distributed attack detection scheme using deep learning approach for internet of things. *Future Generation Computer Systems*, 82: 761-768.

Do CB & Batzoglou S 2008. What is the expectation maximization algorithm? *Nature Biotechnology*, 26(8), 897.

Erhan D, Bengio Y, Courville A, Manzagol PA, Vincent P & Bengio S 2010. Why does unsupervised pre-training help deep learning? *J. Machine Learning Res.*, 11: 625-660.

Goodfellow I, Bengio Y & Courville A 2016. *Deep Learning*. MIT Press.

Gulli A & Pal S 2017. *Deep Learning with Keras*. Packt Publishing Ltd.

Hackeling G 2017. *Mastering Machine Learning with Scikit-Learn*. Packt Publishing Ltd.

Jolliffe IT & Cadima J 2016. Principal component analysis: A review and recent developments. *Phil. Trans. R. Soc. A*, 374(2065): 20150202.

Ketkar N 2017. *Deep Learning with Python*. Apress, pp. 159-194.

Kishor DR & Venkateswarlu NB 2016. A novel hybridization of expectation-maximization and K-means algorithms for better clustering performance. *Int. J. Ambient Comp. and Intelligence (IJACI)*, 7(2): 47-74.

Längkvist M, Karlsson L & Loutfi A 2014. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42: 11-24.

LeCun Y, Bengio Y & Hinton G 2015. Deep learning. *Nature*, 521(7553): 436.

Loukas G, Vuong T, Heartfield R, Sakellari G, Yoon Y & Gan D 2018. Cloud-based cyber-physical intrusion detection for vehicles using deep learning. *IEEE Access*, 6: 3491-3508.

Milenkoski A, Vieira M, Kounev S, Avritzer A & Payne BD 2015. Evaluating computer intrusion detection systems: A survey of common practices. *ACM Computing Surveys (CSUR)*, 48(1): 12.

Montavon G, Samek W & Müller KR 2018. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73: 1-15.

Nguyen KK, Hoang DT, Niyato D, Wang P, Nguyen D & Dutkiewicz E 2018. Cyberattack detection in mobile cloud computing: A deep learning approach. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1-6.

Pai S, Di Troia F, Visaggio CA, Austin TH & Stamp M 2017. Clustering for malware classification. *J. Comp. Virology and Hacking Techniques*, 13(2): 95-107.

Paine TL, Khorrami P, Han W & Huang TS 2014. An analysis of unsupervised pre-training in light of recent advances. *arXiv preprint arXiv:1412.6597*.

Ramsundar B & Zadeh RB 2018. *TensorFlow for Deep Learning: from Linear Regression to Reinforcement Learning*. "O'Reilly Media, Inc.".

Rezvy S, Luo Y, Petridis M, Lasebae A & Zebin T 2019. An efficient deep learning model for intrusion classification and prediction in 5G and IoT networks. *In 2019 53rd Annual Conference on Information Sciences and Systems (CISS)*, pp. 1-6.

Rhode M, Burnap P & Jones K 2018. Early-stage malware prediction using recurrent neural networks. *Computers & Security*, 77: 578-594.

Schmidhuber J 2015. Deep learning in neural networks: An overview. *Neural Networks*, 61: 85-117.

Shen Y, Mariconti E, Vervier PA & Stringhini G 2018. Tiresias. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security - CCS '18. doi:10.1145/3243734.3243811

Tobiyama S, Yamaguchi Y, Shimada H, Ikuse T & Yagi T 2016. Malware detection with deep neural network using process behavior. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, 2: 577-582.

Vasan KK & Surendiran B 2016. Dimensionality reduction using principal component analysis for network intrusion detection. *Perspectives in Science*, 8: 510-512.

Vinayakumar R, Alazab M, Soman KP, Poornachandran P, Al-Nemrat A & Venkatraman S 2019. Deep learning approach for intelligent intrusion detection system. *IEEE Access*, 7: 41525-41550.

Wei J & Mendis GJ 2016. A deep learning-based cyber-physical strategy to mitigate false data injection attack in smart grids. In *2016 Joint Workshop on Cyber-Physical Security and Resilience in Smart Grids (CPSR-SG)*, pp: 1-6.

Wongsuphasawat K, Smilkov D, Wexler J, Wilson J, Mane D, Fritz D, ... & Wattenberg M 2018. Visualizing dataflow graphs of deep learning models in TensorFlow. *IEEE Transactions on Visualization and Computer Graphics*, 24(1): 1-12.

Zhang Y, Li P & Wang X 2019. Intrusion detection for IoT based on improved genetic algorithm and deep belief network. *IEEE Access*, 7: 31711-31722.