Ibor et al. Cybersecurity (2020) 3:14
https://doi.org/10.1186/s42400-020-00053-7

Cybersecurity

RESEARCH Open Access

Conceptualisation of Cyberattack prediction with deep learning

Ayei E. Ibor[1*], Florence A. Oladeji[2], Olusoji B. Okunoye[2] and Obeten O. Ekabua[1]

Abstract

The state of the cyberspace portends uncertainty for the future Internet and its accelerated number of users. New paradigms add more concerns with big data collected through device sensors divulging large amounts of information, which can be used for targeted attacks. Though a plethora of extant approaches, models and algorithms have provided the basis for cyberattack predictions, there is the need to consider new models and algorithms, which are based on data representations other than task-specific techniques. Deep learning, which is underpinned by representation learning, has found widespread relevance in computer vision, speech recognition, natural language processing, audio recognition, and drug design. However, its non-linear information processing architecture can be adapted towards learning the different data representations of network traffic to classify benign and malicious network packets. In this paper, we model cyberattack prediction as a classification problem. Furthermore, the deep learning architecture was co-opted into a new model using rectified linear units (ReLU) as the activation function in the hidden layers of a deep feed forward neural network. Our approach achieves a greedy layer-by-layer learning process that best represents the features useful for predicting cyberattacks in a dataset of benign and malign traffic. The underlying algorithm of the model also performs feature selection, dimensionality reduction, and clustering at the initial stage, to generate a set of input vectors called hyper-features. The model is evaluated using CICIDS2017 and UNSW_NB15 datasets on a Python environment test bed. Results obtained from experimentation show that our model demonstrates superior performance over similar models.

Keywords: Cyberattacks, Prediction, Deep learning, Python, Dimensionality reduction

Introduction

The expansion in the attack landscape has affected a huge number of resources in the cyberspace. According to Sharafaldin et al. (2018a) and Sharafaldin et al. (2018b), attacks involving Botnets, Bruteforce, SQL Injection, Denial of Service (DoS), Infiltration, Heartbleed and Distributed Denial of Service (DDoS) are having tremendous adverse effect on the security of network topologies. Other evolving attacks include analysis, backdoor, exploits, fuzzers, generic, reconnaissance, shellcode, and forms (Moustafa and Slay 2016; Janarthanan and Zargari 2017). Similarly, Tobiyama et al. (2016) and Pai et al. (2017) agree that malign users are developing new techniques that are able to evade network defenses while compromising the internal structure of networks. The accessibility to big data also adds more concerns to the security of data and other digital assets. Though recent researches have tilted towards the modeling of cyberattack prediction, it has become increasingly difficult to identify a single approach that solves the problem of cyberattacks in recent times.

Most approaches in the literature rely on task specific algorithms, thus requiring the need for an approach that relies more on representation learning. That is, an approach that can learn different attack classes from raw data instead of depending on pre-programmed tasks. Dong and Wang (2016), Erfani

* Correspondence: ayei.ibor@gmail.com
[1]Department of Computer Science, University of Calabar, Calabar, Nigeria
Full list of author information is available at the end of the article

et al. (2016), Gulli and Pal (2017), Shone et al. (2018) and Marcus (2018) argue that representation learning can be helpful for extracting the intrinsic features of a dataset in order to generalise on the test cases. In this sense, our model extracts the intrinsic features of network traffic to generate a cascade of concepts for learning the representation of different attack scenarios.

Furthermore, we optimised the accuracy of the model by combining unsupervised and supervised learning to predict 16 attack types in two different datasets as shown in Tables 1 and 2. We evaluated our model for accuracy, false positive rate, precision rate, recall rate, F-measure and entropy in a python environment test bed. Results of experimentation show high prediction accuracy and very low false positive rate for all attack types. Furthermore, we benchmarked our model with similar models to clearly show that it is superior for the prediction of cyberattacks.

Review of related literature

This section introduces the most recent researches in cyberattack detection and prediction with deep learning models in order to establish the relevance of the proposed approach. The extant literature discussed here will also highlight researches that benchmarked public datasets such as KDD99, NSL-KDD and most recently, CICIDS2017 datasets. The modeling of cyberattack detection and prediction systems is fast tilting towards deep learning models. This is based on the fact that these models tend to learn the representations of data instead of the traditional Machine Learning (ML) algorithms, which assume that data is static (Folino and Sabatino 2016; Goodfellow et al. 2016).

For the purpose of clarity, a neural network (NN) is a mathematical model of the information processing and network structure of the human brain. It is a connectionist system consisting of many neurons in layers for communicating signals. A Deep Neural Network (DNN) is a neural network with several hidden layers (Cho, 2014). A DNN typically learns data representations rather than perform task specific functions. In learning data representations, a 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).

Shen et al. (2018) proposed an attack prediction approach called Tiresias xspace. This approach was based on a Recurrent Neural network (RNN) to

Table 1 Evolving Attacks in the CICIDS2017 Dataset

Attack Class	Description	Number of Instances	
		Train	Test
Benign	Normal network traffic	4003	1000
Brute force	This attack is used for password cracking as well as the discovery of hidden pages and content in a web application	5000	1708
Heartbleed	The heartbleed attack emanates from a bug in the OpenSSL cryptography library, which is an implementation of the Transport Layer Security (TLS) protocol	6	5
Botnet	This attack uses a number of devices connected over the Internet to circumvent and exploit vulnerable machines	1500	466
DoS	The Denial of Service (DoS) attack temporarily or indefinitely disrupts services on a host machine connected to the Internet. These services then become unavailable to the intended users for the period of the attack	8000	3936
DDoS	Usually results from a botnet of compromised machines flooding the bandwidth or resources of a victim machine	95,000	32,538
Web	Includes SQL injection, Cross-Site Scripting (XSS) and Brute force over HTTP:	1600	580
	SQL injection is a code injection technique that is used to attack data-driven applications. An attacker can create a string of SQL commands in order to force the database to divulge its contents.		
	XSS attack allows attackers to inject client-side scripts into web pages, which are viewed by other users.		
	Brute force over HTTP enables an attacker to try a list of passwords to find the administrator's password.		
Infiltration	This is an attack that exploits the vulnerability of a software in order to execute a backdoor on the victim's machine. This can lead to attacks such as IP Sweep, port scan and service enumerations.	700	281

Table 2 Evolving Attacks in the UNSW_NB15 dataset

Attack Class	Description	Number of Instances	
		Train	Test
Normal	Benign traffic	56,000	37,000
Analysis	Can be a port scan or spam	2000	677
Backdoor	Bypassing a secured authentication to have access to a machine	1746	583
DoS	See Table 1	12,264	4089
Exploits	Exploitation of a vulnerability in a piece of software	33,393	11,132
Fuzzers	Feeding a network with randomly generated data to cause it to malfunction	18,184	6062
Generic	An attack on block ciphers	40,000	18,871
Reconnaissance	An attack that extracts information about a user or network	10,491	3496
Shellcode	Exploitation of a software vulnerability using a small piece of code as payload	1133	378
Worms	An attack that can replicate itself across multiple connected systems or networks	130	44

predict the possibility of imminent attacks on a host machine using preceding observations. In Nguyen et al. (2018), an approach that used deep learning to detect and isolate cyberattacks in mobile clouds was studied. The approach achieved an accuracy of 97.11% by applying the greedy layer-wise learning algorithm using Restricted Boltzmann Machine (RBM) for pre-training to perform non-linear transformation on its input vectors. The model is then fine-tuned using labeled data to achieve trained weights suitable for detecting attacks.

Similarly, Rhode et al. (2018) predicted the state of an executable code as either malicious or benign with Recurrent Neural Networks (RNNs). The model depended on a short snapshot of behavioural data to obtain a 94% accuracy within the first 5 s of execution and an accuracy of 96.01% during the first 20 s of execution on unseen test set. In Aksu and Aydin (2018), Deep Learning with Support Vector Machine (SVM) algorithm is used to introduce an Intrusion Detection System (IDS) that could detect port scan attempts on a host machine. The approach was evaluated using the CICIDS2017 dataset and reported an accuracy rate of 97.80% for the deep learning model and 69.79% for SVM.

In the same sense, Al-Qatf et al. (2018) proposed a deep learning approach for feature learning and dimensionality reduction. The model could reduce training and testing time and also enhanced the attack prediction accuracy of SVM. Sparse autoencoder was used to build the model for unsupervised pre-training and the transformed feature space was fed into the SVM algorithm to detect attacks. The model reported good detection accuracy for the KDD99 and NSL-KDD datasets. Rezvy et al. (2019) applied a deep autoencoded dense neural network algorithm to detect attacks on Fifth Generation (5G) and IoT networks. The paper presented a 2-step detection approach with deep autoencoders used for unsupervised pre-training to reduce high dimensional data to low-dimensional representation. The next stage performs supervised classification with a deep neural network to achieve good performance with an accuracy of 99.9%. However, this approach is not applied to larger attack types, and it is difficult to ascertain its performance when exposed to current evolving attacks.

An approach called scale-hybrid-IDS-AlertNet was proposed by Vinayakumar et al. (2019). The approach can be used to monitor network traffic in real time in order to indicate the presence of anomalies representing attacks in network traffic. Scale-hybrid-IDS-AlertNet leveraged distributed and parallel machine learning algorithms with a diversity of optimisation techniques for handling a huge number of network and host-level events. Kasongo and Sun (2019) presented an IDS for detecting attacks on wireless networks. The popularity of wireless networks and ease of use has come with many security issues similar to those that affect conventional wired networks. To this effect, the paper discussed the application of a feed forward deep neural network for achieving an effective IDS using NSL-KDD dataset for evaluation.

Zhang et al. (2019) presented a technique that combined the effect of improved Genetic Algorithm (GA) and Deep Belief Network (DBN) to develop an adaptive model for detecting attacks on IoT. The model was simulated and evaluated using the NSL-KDD dataset to recognise attacks and reported the highest accuracy of 99.45% for DoS attacks. 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.

Materials and method

Datasets

The CICIDS2017 dataset is a time-based dataset generated over a 5 day period. It has 80 features with 13 attack types and 1 benign (or normal) traffic (Sharafaldin et al. 2018a; Sharafaldin et al. 2018b; and Sharafaldin et al. 2018c).. These 13 attack types were classified into 7 broad attack types by Panigrahi and Borah (2018). The 7 broad attack classes are discussed in Table 1. The CICIDS2017 dataset has attack diversity, complete network configuration, complete traffic, labelled dataset, heterogeneity, and feature set. It has been used as the benchmark dataset for attack prediction and detection systems (Chadza et al., 2019; Faker and Dogdu, 2019; Vinayakumar et al. 2019; Gharib et al. 2016; Yin et al. 2017; Kasongo and Sun 2019), and several other works.

Similarly, the UNSW_NB15 dataset is a time-based dataset generated over a 16-h period for the training set, and 15-h period for the test set. It has 9 attack types and 49 features (Moustafa and Slay 2015), and also a benchmark dataset for evaluating intrusion prediction and detection systems (Moustafa and Slay 2016; Janarthanan and Zargari 2017). An overview of the evolving attacks in the CICIDS2017 and UNSW_NB datasets is given in Tables 1 and 2.

The model is trained with the training set, validated and tested with the test set for all experiments.

Methodology

The attack data undergoes two learning processes. First, unsupervised learning is used to perform feature engineering and clustering. Unsupervised pre-training is significant for solving the problem of spontaneous classification in order to improve the process of extracting valuable information, which will serve as input to the DNN.

For the second stage, supervised deep learning is used to train the model for making predictions on test data. The model performs cascaded learning based on a deep feed forward neural network with h-hidden dense layers and a Softmax layer for classifying network attacks into one of the classes listed in Tables 1 and 2. The entre prediction process is modeled as a multi-label classification problem.

The proposed model

The architecture of the proposed approach is depicted in Fig. 1.

The components of the model in Fig. 1 include:

i. Network Traffic Capture

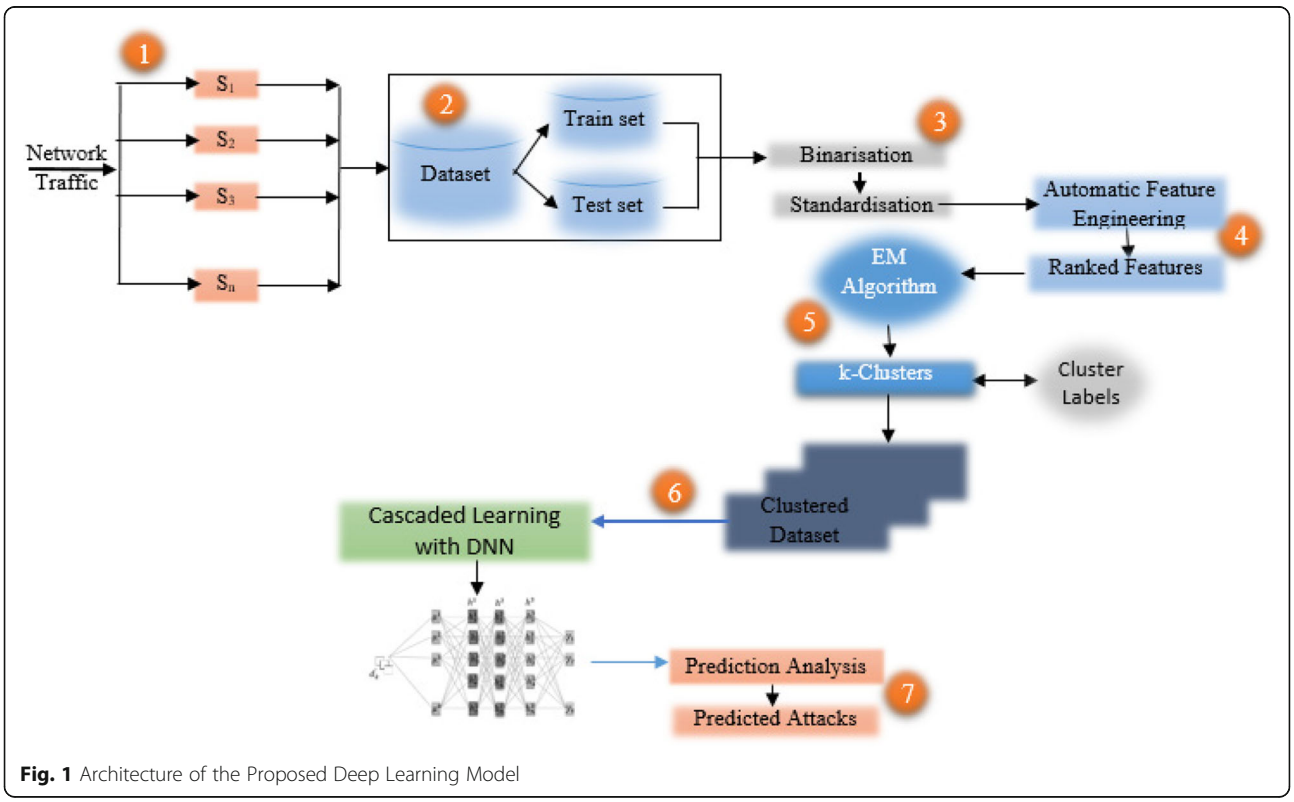

Fig. 1 Architecture of the Proposed Deep Learning Model

The first component represents the capture of network traffic from different sources across the network perimeter. Each source, S_i; $1 \le i \le n$, generates network traffic (malign or benign), which is simulated using the CICIDS2017 and UNSW_NB15 datasets.

ii. Dataset

The dataset, which represents the captured network traffic is further split into the train and test sets for evaluating the performance of our model. In the dataset, each row represents an input vector defined as x_i, $1 \le i \le n$ while each input vector consists of m number of features denoted by f_m. These features can include the destination port, flow duration, total forward packets, total backward packets, protocol type, service state, and so on. Therefore, we defined an input vector in terms of its features as given in eq. 1.

$$x_i = |f_1, f_2, ..., f_m| \tag{1}$$

iii. Normalisation

To achieve an error-free prediction, the captured network traffic is normalised. The normalisation process comes in 2 forms. First, the input vectors are used as rows while the features are used as columns to create an $n \times m$ matrix. This matrix is processed with all categorical values converted to nominal values using label encoders. This is followed by the multi-label Binarisation of all class labels y_k.

Secondly, the dataset is scanned for missing values in order to standardise the range of continuous initial variables or features. In this way, each variable or feature will contribute equally to the analysis of the modeled dataset. Patro and Sahu (2015) assert that standardisation or Z-score normalisation also involves transforming the dataset to comparable scales in order to achieve unbiased results.

Mathematically, a balanced dataset of inputs vectors is created by subtracting the mean and dividing by the standard deviation of each variable or feature in the dataset as given in eq. 2.

$$Z = \frac{x - \bar{x}}{\sigma} \tag{2}$$

Where x is the original samples in the dataset, $\bar{x}$ is the mean, and σ is the standard deviation of the samples. Eq. 2 is relevant for enhancing the convergence speed of the optimisation algorithm. We used an $n \times m$ matrix to represent the normalised dataset as shown in eq. 3.

$$D = \begin{bmatrix} x_{11} & x_{12}... & x_{1m} \\ x_{21} & x_{22}... & x_{2m} \\ x_{n1} & x_{n2}... & x_{nm} \end{bmatrix} \tag{3}$$

iv. Feature Engineering

At stage 4, feature engineering is performed on the dataset using Principal Component Analysis (PCA). This generates a set of p uncorrelated principal components from the correlated feature set. PCA has been used in recent studies for feature engineering such as the works of Ibrahimi and Ouaddane (2017), Moustafa et al. (2017), and Wang et al. (2017). In a dataset, it is likely that some of the features or variables in the dataset will be highly correlated in such a way that they contain redundant information. Thus, it is a good practice, when modeling predictive problems, to remove linear correlations among features in a dataset. In this sense, PCA is used to reduce the feature space while still maintaining the variability in the dataset (Lakhina et al. 2010).

Given the dataset, D, with *n-instances* and *m-features or variables*, PCA generates $min(n-1|, m)$ distinct principal components, which can be used to reconstruct the target output. In this way, the large dataset of connection vectors is easily represented by projecting it on more than one dimensional vector (De la Hoz et al. 2015). The transformed low-dimensional representation is based on the preservation of the variance of the dataset, and the ranking of the principal components. That is, the first principal component contains the largest possible variance, and this threshold decreases with each succeeding principal component as given in eq. (4).

$$d = D(\ min(n-1), m) \tag{4}$$

v. Clustering

The Expectation Maximisation algorithm (EM) is used to generate k number of clusters from the dimensionally reduced dataset. With clustering, the training of the model is improved by automatically categorising attack data. This can be useful in the early steps of an attack. EM performs 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. ***Expectation Step (E-Step)***: the probabilities required in the *M-Step* are computed using the current estimates of the distribution parameters

b. ***Maximisation Step (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, representing the EM algorithm as θ, we have eq. (5).

$$d_k = \theta(d_k) = \theta(x, y) \tag{5}$$

Where, d_k is the clustered dataset by applying the EM algorithm θ on d, k represents the generated number of clusters on d. Since the dataset is 2-dimensional with the instances as a matrix, 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, which is particularly significant for supervised learning. The k^{th} cluster in d_k is represented as μ^k.

Given the statistical model that generates a set x, of observed data, a set of unobserved latent data or missing values y, and a vector of unknown parameters α. Assuming that there is a likelihood function defined as $L(\theta, x, y) = p(x, y|\ \theta)$,the maximum likelihood estimate of the unknown parameters is obtained by maximizing the marginal likelihood of the observed data.

The clusters are then created by using y as a latent variable indicating membership in one of a set of groups as follows:

i) The observed data points x may be discrete or continuous. Associated with each data point may be a vector of observations.

ii) The missing values (or latent variables) y are discrete, drawn from a fixed number of values, and with one latent variable per observed unit.

iii) The parameters are continuous, and are of two kinds: Parameters that are associated with all data points, and those associated with a specific value of a latent variable (i.e., associated with all data points, which corresponding latent variable has that value).

This approach is also used in Dubois et al. (2011) for analysing bioequivalence crossover trials. The clusters (μ^k.) generated based on the membership of the latent variable are then fed into the DNN for supervised learning and classification.

vi. Cascaded Learning with Supervised DNN

Cascaded learning is performed at each layer of the DNN. Each layer passes its information to the next layer through the DNN without feedback connections. The model is trained using the constructed *k*-clusters and cluster labels, generated with the EM algorithm. 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) is used. The DNN is shown in Fig. 2.

vii. Prediction Module

The DNN learns a compressed representation of each cluster μ^k in the hidden layers. At the output layer, the Softmax function is used to classify this compressed representation. In reality, the Softmax function partitions the output such that the total sum is 1, which is equivalent to a categorical probability distribution (Agarap 2018). Thus, the final layer

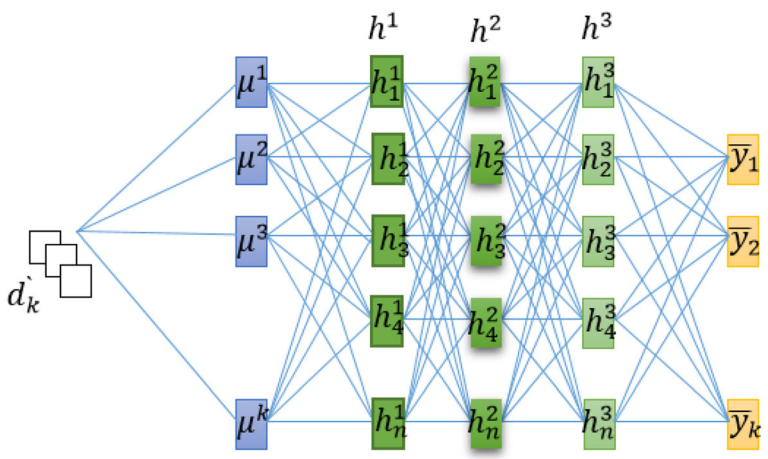

Fig. 2 The Deep Feed Forward Neural Network of the Model

comprises a single neuron for each of the attack classes. Each attack class yields a value between 0 and 1, which is inferred as a probability. The sum of the probability of the output is 1.

To compute the probability of an attack, we applied the Softmax function to each cluster class value as shown in eq. (6).

$$\bar{y} = Softmax(y_k) = \frac{e^{y_k}}{\sum_{k=1}^{n} e^{y_k}} \tag{6}$$

with $\bar{y}$ as the predicted class. We made predictions using equation (6), and the range of $\bar{y}$ (0, 1) indicates the accuracy of predictions. Next, we analysed the predictions made by the model with the help of a confusion matrix, and then computed the following evaluation metrics based on the work of Milenkoski et al. (2015):

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

$$ACC = \frac{TP + TN}{TP + TN + FN + FP} \tag{7}$$

Where,

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

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

$$FPR = \frac{FP}{TN + FP} \tag{8}$$

c) **Precision Rate (PR)**: the fraction of relevant instances in the dataset given as:

$$PR = \frac{TP}{TP + FP} \tag{9}$$

d) **Recall Rate (RR)**: the retrieved relevant instances over the total amount of relevant instances. RR calculated as shown in equation (10):

$$RR = \frac{TP}{TP + FN} \tag{10}$$

e) **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:

$$F1 = 2x\frac{PR.RR}{PR + RR} \tag{11}$$

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

$$E = -\sum_{i=1}^{n} y_i \log(\bar{y}_i) \tag{12}$$

In equation (12), n is the number of classes, y is the true class value and $\bar{y}$ 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.

Underlying algorithm of the model

The model's underlying algorithm is given in Algorithm 1.

Algorithm 1:

1. *Initialise all parameters and variables*
2. *Initialise number of input vectors as:* $x_i = |f_1, f_2, \ldots, f_m| \; ; 1 \le i \le n$
3. *Split the dataset D, to obtain the train and test sets with x instances and y labels: D = D(x, y)*
4. *Generate principal components from D:* $d = D(min(n-1), m)$
5. *Generate k clusters from d:* $d_k = \theta(d_k) = \theta(x, y)$
6. *For k = 1 to numberofClusters*
   Assign clusters labels: $y_k = \theta(\mu^k)$
   end For
7. *j = 0*
8. *For i = 1 to 5*
   if (i==1) return inputLayer units
   if (i==5) return outputLayer units
   else
   j = j+ 1
   return h^i *as hidden layers units*
   end For
9. *i = 1*
10. *Repeat*
    if(i==1) units = 750 for h^1
    if(i==2) units = 500 for h^2
    if(i==3) units = 250 for h^3
    i = i + 1
    Until (i>j)
11. *For n = 1 to epochs*
    Train model on train set
    Validate and test model with test set
    Predict attacks using $\bar{y} = Softmax(y_k) = \frac{e^{y_k}}{\sum_{k=1}^{n} e^{y_k}}$
    end For
12. *Return* $\bar{y}$ *as the predicted attack class*

Feature ranking

The model was trained using the EM generated number of clusters (k-clusters) based on the features in the dataset. The k-clusters were formed from a feature space with a reduced dataset, d, using PCA. PCA generated p-principal components, representing a compressed feature space as mentioned in Vasan and Surendiran (2016). With PCA, the variance in the data was optimised to generate the representative subset of features for training the model.

After training the model with the train set, it was able to generalise to an out-of-sample (test) data while deriving an accurate estimate of model prediction performance.

Testbed of the experiments

We implemented the DNN 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). The system properties of the machine used for experimentation are shown in Table 3.

Experimental results and discussion

The experimental results for this implementation are discussed in this section. The tuning of the hyperparameters for the deep neural network and the predictions made at the completion of the execution of the Python code used for the implementation are presented. Furthermore, the performance of the model is benchmarked against state-of-the-art approaches and findings show that the proposed model outperforms other models in terms of accuracy and false positive rate.

Configuration and tuning of Hyperparameters

The DNN performed computations on the transformed feature space, which is basically comprised of numeric values. In each layer, the feature space is compressed and abstract features for the optimal representation of the original dataset are learned, transformed and passed on to subsequent layers in a cascaded learning technique. To achieve the required accuracy, the DNN must have an adequate number of layers. Similarly, each layer must have an adequate number of neurons in order to be able to represent the different output classes during predictions.

In a DNN, hyperparameter tuning helps the model to generalise on the training data while finding the distinctions between the output classes (Rhode et al. 2018). It is noteworthy to mention that there are several ways the hyperparameters of the DNN can be configured and tuned. During the experiments, different configurations were chosen and tested. In tuning the model, it was found that a depth of 3 hidden layers produced the best results.

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

Additionally, the number of neurons per layer and the number of epochs for training the model were randomly chosen. Through this random search process, the hyperparameter values for the optimal performance of the model were chosen. Subsequently, the hyperparameters that generated the best performance of the model are summarised in Table 4.

Training and testing of the model

The model was trained and tested with 500 iterations or epochs. The visualisation of the pre-trained samples for the dataset is illustrated in Fig. 3.

The training of the model was based on three sequence of processes as mentioned in Kasongo and Sun (2019). These processes include:

i. Forward propagation, in which case, each layer passes information to the next layer
ii. Back propagation of the error computed during the cascaded learning process
iii. The update of the weights and biases across the DNN.

The update of all the weights and biases was based on a backpropagation algorithm, which is optimised with stochastic gradient descent (SGD) and an Adam updater. A standard categorical cross-entropy loss function is used at the output layer. This loss function measures the model's classification performance whose output is a probability value within the range of 0 and 1. When the predicted probability diverges from the actual value, the value of cross-entropy loss increases, and then tends towards 0 as the predicted probability converges to the actual value. A cross-entropy loss of 0 implies a perfect model.

The model is trained with an initial learning rate of 0.1. A lower learning rate of 0.01 was subsequently introduced to test the model's predictions over the training and test data. However, the optimisation took a longer time due to the tiny steps towards the minimum of the loss function. It is important to note that choosing the appropriate learning rate is significant for achieving optimal

Table 4 Hyperparameters for Optimal Model Performance

Hyperparameter	Experimented Values	Selected Configuration
Number of hidden dense layers	1 to 10	3
Neurons per layer	100 to 1000	750, 500, 250
Number of epochs	10 to 500	500
Batch Size	32, 64, 128	64
Learning Rate (lr)	0.0001, 0.01, 0.1, 0.2, 0.3	0.01, 0.1

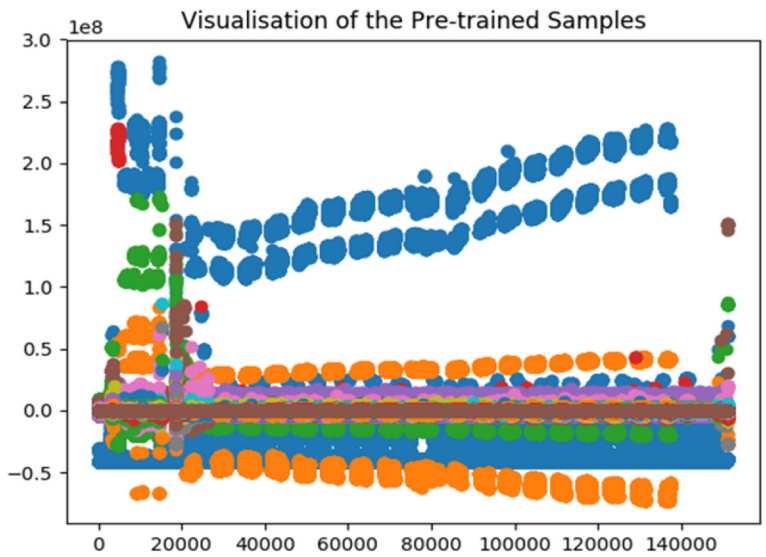

Fig. 3 Visualisation of the Pre-trained Samples

predictions. This is because a high learning rate may result in the training not converging, or even diverging. Consequently, changes in weights can be large enough to allow the optimizer overshoot the minimum and make the loss worse.

At the end of the training phase, the performance of the model was evaluated and tested using out-of-sample (or test) data. The results obtained are presented in the next section.

Discussion of results

For each training, validation and testing phase, the performance of the model was recorded using such metrics as Accuracy, Recall Rate, Precision Rate, F-measure and Cross Entropy for the modeled datasets and selected learning rate. The results obtained are shown in Table 5 for the CICIDS2017 dataset and in Table 6 for the UNSW_NB15 dataset. From Tables 5 and 6, the model demonstrated good stability for the 16 attack and two benign (or normal) classes in both datasets, thus showing a significant improvement over any existing model.

All attack classes in the CICIDS2017 dataset were predicted with an accuracy of more than 0.99 or 99% with values of E close to 0.

Similarly, the 9 attack classes in the UNSW_NB15 dataset were predicted with the highest accuracy of 99.92% for Worms, and the lowest accuracy of 81.68 for Analysis. The values of E were also very low, an indication of a good predictive model. The Accuracy and Cross Entropy Loss curves for the CICIDS2017 and UNSW_NB15 datasets are depicted in Figs. 4 and 5.

As shown in Figs. 4 and 5, there is no significant deviation between the train and test curves showing that the model is able to learn a representation of the attack clusters from the raw attack data. 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 both datasets, the model predicted the attacks and benign (or normal) traffic in the datasets accurately. This shows that our model has a strong modeling ability, and

Table 5 Performance Metrics of the Model for CICIDS2017 Dataset using lr = 0.1

Metrics	Benign	DDoS	DoS	Web attack	Bruteforce	Heartbleed	Botnet	Infiltration
ACC	99.98	99.82	99.93	99.98	99.93	100.0	99.95	99.93
PR	99.99	99.99	99.94	99.98	99.95	100.0	99.99	99.94
RR	99.99	99.97	99.99	100.0	99.99	100.0	99.96	99.93
F1	99.99	99.98	99.97	99.99	99.97	100.0	99.97	99.96
E	0.000064	0.000064	0.000598	0.000234	0.000534	0.0	0.000149	0.000064

Table 6 Performance Metrics of the Model for UNSW_NB15 Dataset using lr = 0.1

Metrics	Normal	Analysis	Back Door	DoS	Exploits	Fuzzers	Generic	Recon	Shell code	Worms
ACC	89.46	81.68	98.71	91.41	85.10	90.76	98.78	96.94	99.29	99.92
PR	99.18	82.02	99.97	99.79	96.41	93.74	99.82	99.21	99.87	100.0
RR	93.69	99.79	98.74	91.60	89.91	97.69	99.32	97.87	99.42	99.93
F1	96.36	90.03	99.35	95.52	93.05	95.67	99.57	98.54	99.64	99.96
E	0.00818	0.16261	0.00025	0.00206	0.03526	0.06058	0.00181	0.00788	0.00130	0.00005

yields very high accuracy for predicting multi-class attacks.

In Table 7, the overall performance of the model is shown. The model showed improved performance for all the 16 attack and two benign classes used. Our model was able to learn more abstract features of the dataset during the training phase at each layer to make better generalisations for predicting the modeled attack and benign traffic while minimising the cross entropy loss, and false positive rate.

Furthermore, a Precision-Recall analysis is performed by plotting the Precision Rate (PR) on the x-axis against the Recall Rate (RR) on the y-axis, to ascertain the stability of the model in making predictions. The PR-RR Analysis is represented in Fig. 6. This plot shows very high stability, and affirm the suitability of the proposed model for multi-class predictions.

Comparison of results

In our model, we used unsupervised and supervised learning techniques to achieve very high accuracy in the prediction of cyberattacks. From the results obtained during experimentation, our model demonstrated more than 99% prediction accuracy for 9 of the 16 attack classes, and more than 90% prediction accuracy for 14 of the 16 attack classes in both datasets. The benign classes were also predicted with very high accuracy, thus the negligible FPR achieved. This is clearly indicated by the test plots of Figs. 4 and 5. Similarly, the cross entropy loss of the model indicates that our model is a good classifier. The FPR, which is used as the prediction error of the model was minimal, implying that only a few instances of the benign and attack data were misclassified or predicted incorrectly. We benchmarked the results of experimentation of our model against extant state-of-the-art techniques in deep learning as shown in Table 8. This comparison shows that our model outperforms extant approaches as illustrated in Fig. 7.

As shown in Table 8, our model achieved an overall accuracy of 99.99%, and FPR of 0.00001. The approach of Rezvy et al. (2019) also demonstrated good performance with an accuracy of 99.9%. However, this approach is applied to only three attack types in one dataset, which is not substantial for measuring performance on most evolving attack types due to the complexity in analysing and predicting them. Consequently, our model demonstrated significant improvement over existing models for cyberattack prediction.

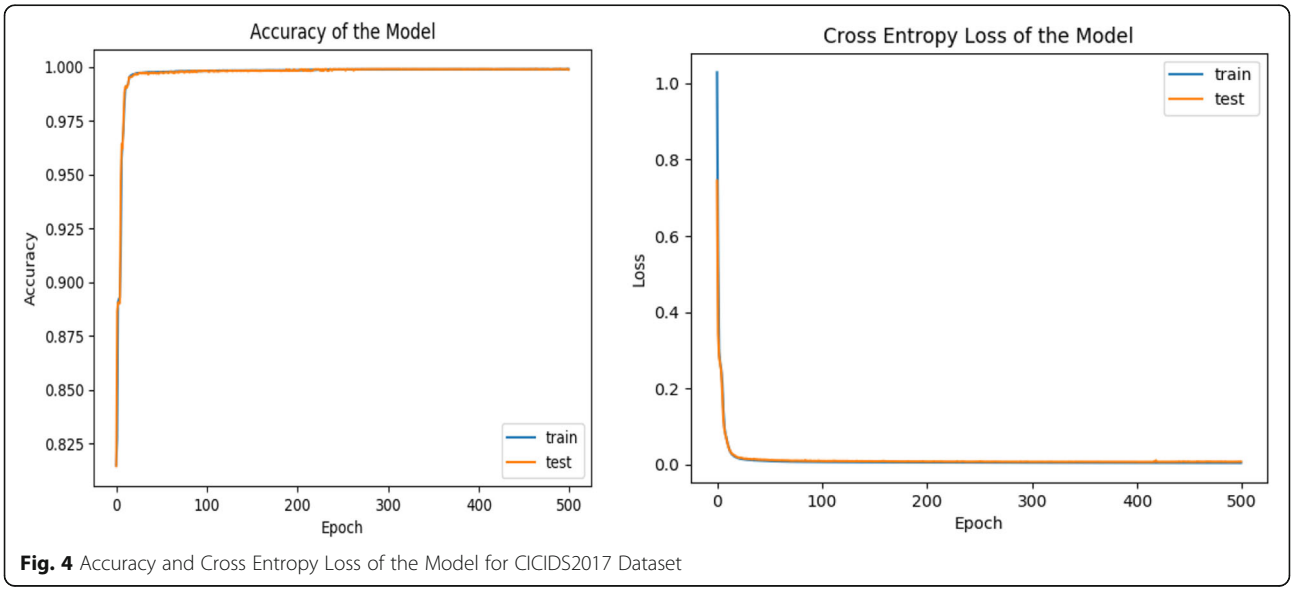

Fig. 4 Accuracy and Cross Entropy Loss of the Model for CICIDS2017 Dataset

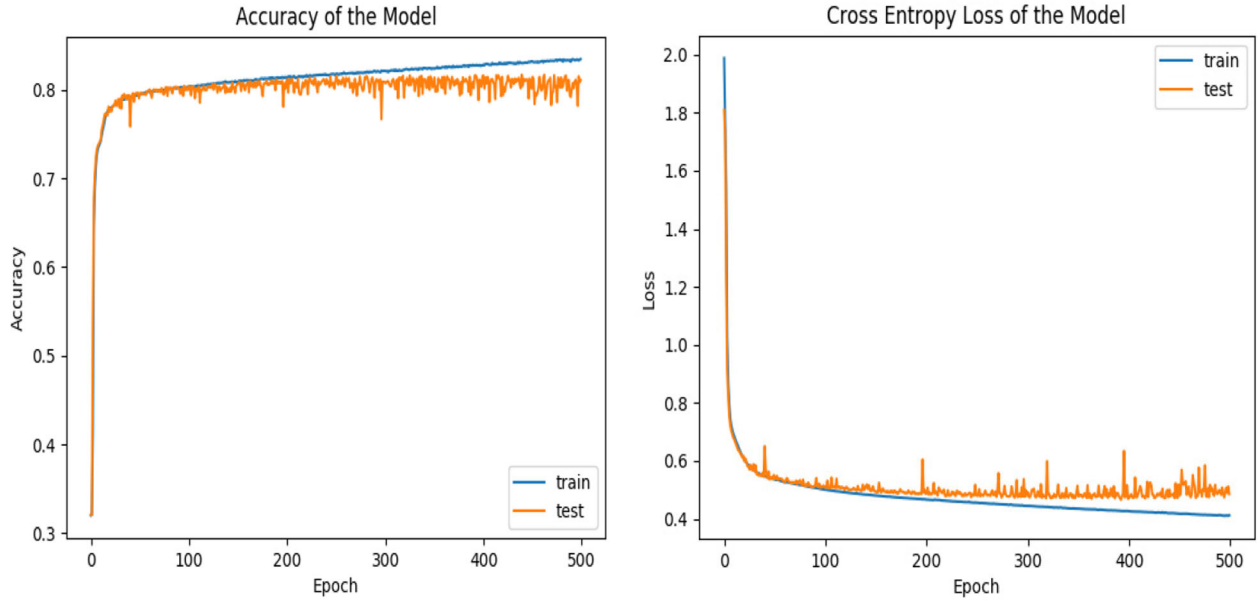

Fig. 5 Accuracy and Cross Entropy Loss of the Model for the UNSW_NB15 Dataset

Conclusion

We introduced a new approach to cyberattack prediction never used in any previous work. We showed that by clustering the attack data using an unsupervised learning approach prior to classification and prediction of attacks, we can achieve very high prediction accuracy suitable for the current cyberspace. Though there are numerous approaches in the literature for the same problem, our approach demonstrated that it is possible to use one trained model and topology to effectively predict multiple attacks, especially at the early stages of the attack. The combination of techniques used in this work is novel, and can be very useful for current embedded systems, which do not require very complicated design. Furthermore, we evaluated the model using CICIDS2017 and UNSW_NB15 datasets as the benchmarked datasets. These datasets have large sets of connection vectors of evolving attacks, which enabled us to tune the model to learn different attack types to make accurate predictions. Finally, we obtained a prediction accuracy of 99.99% for most of the modeled attack types, thus outperforming extant approaches for the same problem domain.

Table 7 Overall Performance of the Model for lr = 0.1

Epochs	500
Metrics	
ACC	99.99761
FPR	0.00003
PR	99.97223
RR	100
F!	99.98611
E	0.00028

Table 8 Performance Comparison of the Proposed Model with Extant State-of-the-Art Approaches

Approach	Accuracy	False Positive Rate
Proposed Model	99.99%	0.00001
Rezvy et al. (2019)	99.9%	0.1
Vinayakumar et al. (2019)	93.5%	6.45
Kasongo and Sun (2019)	99.54%	0.43
Zhang et al. (2019)	99.45%	0.54

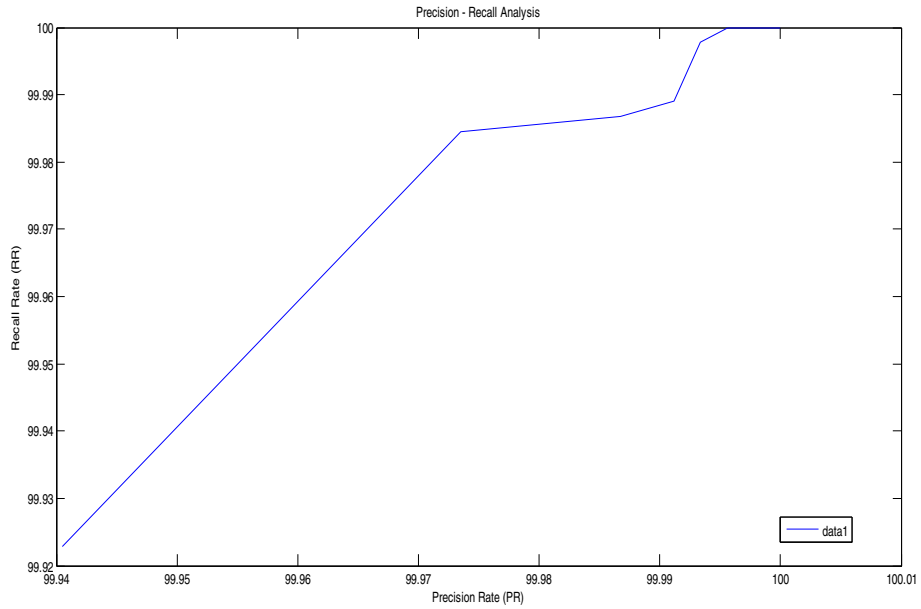

Fig. 6 Precision – Recall Analysis of the Proposed Model

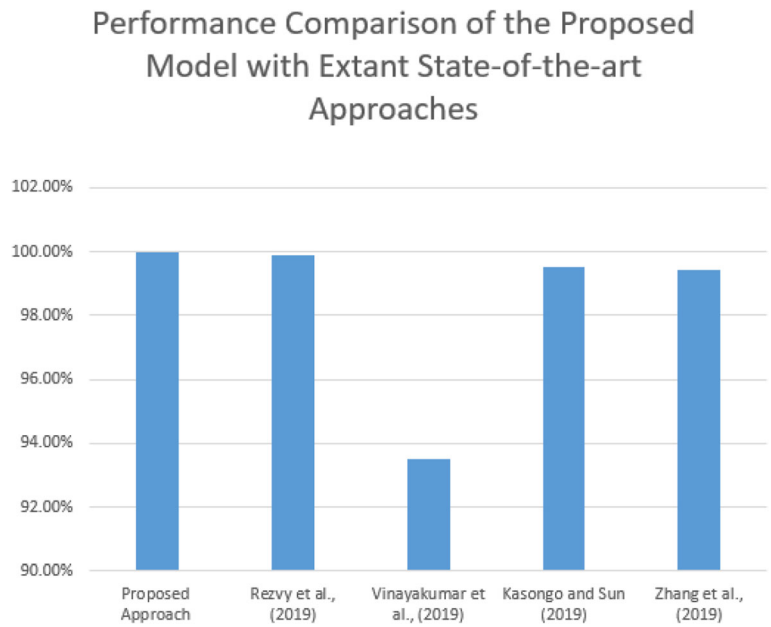

Fig. 7 Visualising the Accuracy of the Proposed Model against other Approaches

Acknowledgements
Not applicable.

Authors' contributions
This work is the original idea of the first author. The work was supervised by the second and third authors, while the fourth author served as an adviser. All experimentations and reporting were performed by the first author, and validated for correctness and relevance by the second and third authors. The authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
CICIDS2017 Dataset is available at https://www.unb.ca/cic/datasets/ids-2017.html while the UNSW_NB15 dataset is available at https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

Competing interests
The authors declare that there are no competing interests in this research work.

Author details
[1]Department of Computer Science, University of Calabar, Calabar, Nigeria. [2]Department of Computer Sciences, University of Lagos, Lagos, Nigeria.

Received: 19 October 2019 Accepted: 12 April 2020
Published online: 17 June 2020

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... and 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

Aksu D, and Aydin MA. (2018). Detecting port scan attempts with comparative analysis of deep learning and support vector machine algorithms. In 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT). IEEE, Ankara, pp. 77–80

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

Chadza T, Kyriakopoulos KG, Lambotharan S. (2019). Contemporary Sequential Network Attacks Prediction using Hidden Markov Model. In 2019 17th International Conference on Privacy, Security and Trust (PST). Fredericton: IEEE (pp. 1-3).

Cho K (2014) Foundations and advances in deep learning.Taxonomy, and future directions. Comput Commun 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

Dong B, Wang X. (2016). Comparison deep learning method to traditional methods using for network intrusion detection. In 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN). IEEE, Beijing, pp. 581-585.

Dubois A, Lavielle M, Gsteiger S, Pigeolet E, Mentré F (2011) Model-based analyses of bioequivalence crossover trials using the stochastic approximation expectation maximisation algorithm. Stat Med 30(21):2582–2600

Erfani SM, Rajasegarar S, Karunasekera S, Leckie C (2016) High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. Pattern Recogn 58:121–134

Faker O, Dogdu E. (2019). Intrusion detection using big data and deep learning techniques. In Proceedings of the 2019 ACM Southeast Conference - ACMSE. Kennesaw: 2019:86-93.

Folino G, Sabatino P (2016) Ensemble based collaborative and distributed intrusion detection systems: a survey. J Netw Comput Appl 66:1–16

Gharib A, Sharafaldin I, Lashkari AH, Ghorbani AA (2016) An evaluation framework for intrusion detection dataset. In 2016 international conference on information science and security (ICISS). IEEE, Pattaya, pp 1–6.

Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MT Press, Cambridge.

Gulli A, Pal S (2017) Deep learning with Keras. Packt Publishing Ltd, Birmingham.

Hackeling G (2017). Mastering machine learning with scikit-learn. Packt Publishing Ltd, Birmingham.

Ibrahimi K, Ouaddane M (2017) Management of intrusion detection systems based-KDD99: analysis with LDA and PCA. In 2017 international conference on wireless networks and Mobile communications (WINCOM). Rabat, IEEE, pp 1–6.

Janarthanan T, Zargari S (2017). Feature selection in UNSW-NB15 and KDDCUP'99 datasets. In 2017 IEEE 26th international symposium on industrial electronics (ISIE). IEEE, Edinburgh, pp 1881–1886.

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

Lakhina S, Joseph S, Verma B (2010) Feature reduction using principal component analysis for effective anomaly–based intrusion detection on NSL-KDD

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

Marcus, G. (2018). Deep learning: a critical appraisal. arXiv preprint arXiv:1801.00631

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

Moustafa N, Slay J (2015). 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, Canberra, pp 1–6.

Moustafa N, Slay J (2016) The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Inf Secur J 25(1–3):18–31

Moustafa N, Slay J, Creech G (2017). Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks. IEEE Transactions on Big Data. 5(4):481-94.

Nguyen KK, Hoang DT, Niyato D, Wang P, Nguyen D, and Dutkiewicz E. (2018). Cyberattack detection in mobile cloud computing: a deep learning approach. In 2018 IEEE wireless communications and networking conference (WCNC). IEEE, Barcelona, pp. 1–6.

Pai S, Di Troia F, Visaggio CA, Austin TH, Stamp M (2017) Clustering for malware classification. J Comput Virol Hacking Tech 13(2):95–107

Panigrahi R, Borah S (2018) A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems. Int J Eng Technol 7(3.24):479–482

Patro, S., and Sahu, K. K. (2015). Normalization: a preprocessing stage. arXiv preprint arXiv:1503.06462

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). IEEE, Baltimore, pp 1–6.

Rhode M, Burnap P, Jones K (2018) Early-stage malware prediction using recurrent neural networks. Comput Secur 77:578–594

Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85–117

Sharafaldin I, Gharib A, Lashkari AH, Ghorbani AA (2018a) Towards a reliable intrusion detection benchmark dataset. Softw Netw 2018(1):177–200

Sharafaldin I, Lashkari AH, and Ghorbani AA. (2018b). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018). Funchal: pp. 108-116.

Sharafaldin I, Lashkari AH, Ghorbani AA (2018c) A detailed analysis of the CICIDS2017 data set. In international conference on information systems security and privacy. Springer, Cham, pp 172–188

Shen Y, Mariconti E, Vervier PA, Stringhini G (2018) Tiresias. Proceedings of the 2018 ACM SIGSAC conference on computer and communications security - CCS '18. https://doi.org/10.1145/3243734.3243811

Shone N, Ngoc TN, Phai VD, Shi Q (2018) A deep learning approach to network intrusion detection. IEEE Trans Emerg Topics Comput Intell 2(1):41–50

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). IEEE 2:577–582

Vasan KK, Surendiran B (2016) Dimensionality reduction using principal component analysis for network intrusion detection. Perspect Sci 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
Wang H, Gu J, Wang S (2017) An effective intrusion detection framework based on SVM with feature augmentation. Knowl-Based Syst 136:130–139
Yin C, Zhu Y, Fei J, He X (2017) A deep learning approach for intrusion detection using recurrent neural networks. Ieee Access 5:21954–21961
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

Publisher's Note