International Journal of Electrical and Computer Engineering (IJECE)
Vol. 11, No. 2, April 2021, pp. 1498~1509
ISSN: 2088-8708, DOI: 10.11591/ijece.v11i2.pp1498-1509

Forging a deep learning neural network intrusion detection framework to curb the distributed denial of service attack

Arnold Adimabua Ojugo[1], Rume Elizabeth Yoro[2]
[1]Department of Computer Science, Federal University of Petroleum Resources Effurun, Warri, Nigeria
[2]Department of Computer Science, Delta State Polytechnic Ogwashi-Uku, Nigeria

Article Info

Article history:

Received Apr 22, 2020
Revised May 28, 2020
Accepted Jul 10, 2020

Keywords:

Data security
DDoS
Deep neural network
Intrusion detection system
Machine learning
Spam

ABSTRACT

Today's popularity of the internet has since proven an effective and efficient means of information sharing. However, this has consequently advanced the proliferation of adversaries who aim at unauthorized access to information being shared over the internet medium. These are achieved via various means one of which is the distributed denial of service attacks-which has become a major threat to the electronic society. These are carefully crafted attacks of large magnitude that possess the capability to wreak havoc at very high levels and national infrastructures. This study posits intelligent systems via the use of machine learning frameworks to detect such. We employ the deep learning approach to distinguish between benign exchange of data and malicious attacks from data traffic. Results shows consequent success in the employment of deep learning neural network to effectively differentiate between acceptable and non-acceptable data packets (intrusion) on a network data traffic.

Corresponding Author:

Arnold Adimabua Ojugo
Department of Computer Science
Federal University of Petroleum Resources Effurun
P.M.B 1221, Effurun, Warri, Delta State, Nigeria
Email: ojugo.arnold@fupre.edu.ng

1. INTRODUCTION

The rapid advancement in technology over the years- has been geared towards effectively improving the way and how we live in a bid to meets specific targeted human needs. Technology seeks to better advance our living existence unto higher plains cum levels of sophistication with ease. Tremendous adoption and integration of the Internet has significantly advanced the use of data sharing programs that seeks to effectively disseminate data froom one user to another [1]. This adoption and integration has been attributed to its usage ease, ubiquity of its nature, low-cost of transaction and trust in communication channel-all of which continues to advance its popularity, adoption ease and usage. This growth has equally attracted spams [2, 3] an organized business aimed at making money via use of messages without the consent of users. Their services are unsolicited adverts, phishing and malware distribution called spams. Spams are unsolicited/unwanted messages sent to users. With spams on the rise, it has proven a great concern to security experts [4-7].

Such compromises designed to evade security, obscure data privacy and weaken network infrastructure have become a great concern with negative impacts on the adoption of technology. This includes (not limited to) attacks on data, stealing of private data, intrusion, service denial and outage [8]. Reports continues to stress of intrusion to networks that effectively attacks any given target at any given time [9-11]. The exponential rate of attacks is as broad as the range of constructive technology it self-leading to

denial of service attacks amongst others. This calls for urgent needs to terminate as close and as fast as possible to its source, the attacks. Many of these attacks on network resources are coordinated and targeted at a client system and launched against the server via a number of compromised systems [12-15]. DDoS threatens today's network as they are carefully crafted to target a large number of users as well as wreaks havoc at various security levels. The ease with which they are perpetrated has also become a great concern, even with a plethora of available tools. Thus, most studies use of machine learning methods to effectively differentiate between good and malicious data-packets that attempts to perform intrusion [16-19].

Distributed denial of service (DDoS) is usually a carefully crafted attack, initiated to target network resource(s). Targeted directly at compromised clients on a network [12, 20, 21], an adversary gains access of such compromised client, seizing up resources such CPU time, bandwidth, memory etc- to exploit a network's weakness. An adversary achieves this, by inserting malware that sought to overwhelm the network with requests [20-23]. Since the DDoS is carefully crafted and well-coordinated, the magnitude depends on the botnet size-which corresponds also to the severity of the attack [19-21]. A DDoS seeks to exhaust targeted resources, deny 'uncompromised' clients' access to services and ensure a compromised network on a larger-scale. DDoS attacks are easy to fix by manually disconnecting affected clients from the network, when detected. Detection schemes seek to stop a detected attack as close and as fast as possible to its source [15, 20, 21, 24-27].

There are two common forms of a DDoS attack: (i) An adversary seeks by exploit design, to flood a network with client requests that exhausts or seize up CPU-time, power, bandwith etc- making it difficult for other clients to access these resources (i.e. flooding); and (ii) Adversary sends large volume of malicious packets to a server (i.e. protocol attack). A DDoS attack can evade detection if the adversary spoofs the source address to mask packets and make it difficult to differentiate genuine from malicious data [28, 29]. Detection schemes are usually grouped based on their locality of deployment as [14, 20, 21, 30]:

- When a client system (known as source device) has security mechanism that helps it identify a malicious message in an outgoing packet and filters it. Such detection is said, to have been launched at the source of the attack-oreventing network clients generating a DDoS attack. This detection tries to stop a DDoS as close and as fast as possible to the source of the attack (a best practice) and minimizes havoc on the network as well as on other uncompromised legitimate packets cum traffic [20, 21].
- When a compromised system detects incoming malicious packet, it can clearly distinguishing genuine 'uncompromised' packets from 'compromised' attack packets from either the misuse of intrusion, or via an anomaly-intrusion detection scheme. This is called a victim-end detection. And an attack packet that reaches a victim may denied/degraded services and bandwidth saturation [20, 21].
- When a network router can independently attempt to identify a malicious packet by rate-limit on data-trying to balance between the accuracy of the detection and the consumption bandwidth of an attack. Such detection trace-back becomes easy-because the packet traffic are then aggregated by placing a rate-limit on all traffic data since both genuine and attack packets arrive at the router [20, 21]. This is usually called the core-end detection.

Knowledge-driven methods of detection-resources are a stream of events, checked on the backdrop of predefined attack rules and patterns. A general view of known attacks are formulated, so a system easily identifies occurrence of an attacks using either of signature/anamolay analysis, self-organizing maps, and state transition analysis. Gil and Poletto [31] used a multi-level tree for online packet statistics to monitor traffic feats on devices, and to detect/eliminate DDoS. It aggregates and rates packets statistics at various levels, expand/contract packets to successfully detect ongoing attack via a disproportional difference between the rates in/out a network. It is setup at locations that allows the equipped device to either detect, or fail to monitor bandwidth attack. Shrivaraj [32] Attackers can evade such detection mode by randomizing the source address IP for such malicious data and/or packets.

Thomas *et al*. [33] used NetBouncer to distinguish 'uncompromised' from 'compromised' clients. The client are updated on a list-so that only clients on this list are allowed access to network resources. If a client not in the list send a packets, the NetBouncer test for legitimacy if such a client is compromised. If the client passes the tests, it is added to the legitimacy list and such client is granted access to resources until the window for legitimacy expires. With the list's expiration, clients are revalidated.

Evolutionary driven and machine learning (ML) frameworks-learning algorithms effectively classify attacks with heuristics that can tolerate uncertainty, noise, ambuities, and imprecision while yielding an optimal solution. It models traffic data as a set of test for statistical inference, which seeks to determine if a new instance belongs to the class. Instances that do not conform to the trained model is classified as an anomaly. Nguyen [34] used a proactive detection to classify network status, which break an attack into phases-so that it can be investigated based on selected feats of interest. It then uses k-nearest neighbor (KNN) to group data feats of the network status into each phase of the DDoS attack.

Wu *et al*. [35] used a decision trees that detects attack using 15 parameters to monitor packets and flag rates in and out a system, to describe a traffic flow pattern. It detects abnormal traffic flow via a match scheme that identifies traffic flow similar to an attack flow as well as to trace back the origin of an attack based on this similarity. Lee *et al*. [36] used cluster analysis on DARPA 2000 dataset. Their results showed that each attack was partitioned, and their method can effectively detect precursors of a DDoS, and the attack itself. Ojugo *et al*. [37] used a signature-based memetic algorithm to detect attack as a classification problem. It uses seven parameters to monitor packet rate and traffic pattern. It uses a match method to identify traffic flow(s) into classes and trace them back to an attack's origin via the similarity. Karinmazad and Faraahi [38] used anomaly-based detection with packet feats, analyzed via a radial basis function network that was applied to an edge-routers on a victim networks. It uses seven-feats to train a RBF-net and classifies data into normal and attack classes. If model recognizes an incoming traffic as attack, its source packets are sent to a filter/attack-alarm routine for further actions. Else, it is sent to its destination.

Moore [26] proposed a detection scheme where each router detects traffic anomalies using profiles of a normal traffic constructed via stream sampling algorithms. Their results indicates: (i) we can profile a normal traffic reasonably accurately; (ii) identify anomalies with low false-positive and false-negative rates; and (iii) be cost effective with memory consumption per packet computation. Also, the routers exchanges data with each other to increase confidence in their detection. Results show that each router profiles capture key characteristics of the traffic effectively and identify anomalies with low false positive and false negative rate. Ojugo *et al*. [15] extended Ojugo *et al*. [37] via a genetic algorithm signature rule-based model, with 10-feats to monitor in/out packet rates. Jalili *et al*. [39] advanced this position using SPUNNID-an unsupervised neural net to extract traffic feats, analyse and classify traffic patterns as either a normal or DDoS attack.

Chen and Delis [40] used a distributed change point (DCP) detection that adopts change aggregation trees (CATs). This non-parametric model describes distribution of pre/post change in traffic. When a DDoS flood-attack is launched, the cumulative deviation is noticeably higher than random fluctuations. The CAT is designed so a router detects abrupt changes in traffic. A domain server uses the traffic change patterns detected at attack-transit routers to construct CATs. It works in inline-mode to inspect and manipulate ongoing traffic in real time. It continuously monitors both attacks and legitimate traffic by inspecting packets and correlating events among different sessions. It proactively terminates a session when it detects an attack.

Intrusion schemes have been devised to minimize the havoc by intrusion activities [41, 42], and for networks, some behavior exists with an external event. The architecture of intrusion detection systems (IDS) seeks to unmask malicious processes either via the signature of such attack, or via an anomaly on the network traffic. An IDS goal is to secure network resources and grant a user sytem confidentiality, data integrity, and resource availability [15, 16]. An IDS can retrieve data from a network section for analysis to iunveil intrusion affected component(s) using a various techniques. These techniques are characterized to depend on 3-main aspects [15, 25, 43, 44] in Figure 1:

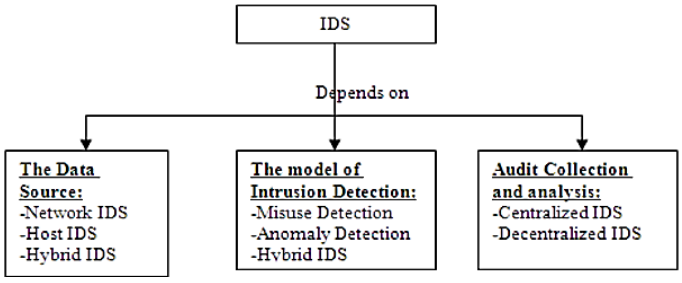

Figure 1. Structural architecture and classification of an IDS

- Data source-a network IDS examines the traffic; while, a host IDS examines network components such as the operating system. Conversely, a hybrid IDS supports both sources of data.
- Intrusion model deals with misuse detection. Signature detection aims to verify the signature on data traffic; While, anomaly detection seeks to verify the system behavior. Hybrid IDS monitors both detection modes.
- Audit collection and analysis is implemented using 2-methods namely: (i) centralized-controlled resource IDS; and (ii) decentralized IDS is controlled from a local control node with hierarchical reporting to one or more central location(s).

Motivation for the study:

- DDoS attacks are rising with ease at an alarming growth rate of malicious tasks guised to exploit users. They have caused user's loss in trust-level of tech adoption and finances. Using particular technique to resolve DDoS with IDS [2], and statistical models [44] have proven successful on malicious traffics. However, combating DDoS is an 'inconclusive' and continuous task as many of models' performance are hampered by selection of parameters that often results in model over-fitting and over-training [*].
- Many model(s) employ hill-climbing methods; and thus, often gets trapped at local maxima.
- DDoS attack prevents legitimate users from accessing resources. It consumes all available resources, overwhelms the network with requests overload, blocks uncompromised users' access to provisioned services with a view to compromise the entire network until countermeasures are employed. Thus, the urgent need to identify their source, manage and prevent them. This is effectively achieved via statistical means and guides a user to efficiently differentiate between legitimate and malicious acts.
- Formulating an effective detection scheme has its setback(s)-as malicious traffics are poised by their design architecture to evade filters, whose performance are hindered by the limited size of characters, non-availability of malicious traffic data etc-creating impediments in selecting parameters for training. And, ultimately, leading to both poor learning and classification of the learning algorithm.

To overcome these shortfalls in detecting malicious traffics, we adopt a deep neural network to reduce noise via pre-processing of traffic packets and fine-tuning messages sent as requests sent to/from a server, to enhance adequate classification.

2. RESEARCH METHOD

2.1. Deep neural networks (DNN)

DNN uses deep learning to adapt useful selected feats of interest and parameters, carefully constructing a multi-layer network from vast amount of data. Its deep architecture at its input, hidden and output layers-helps to improve its prediction accuracy. Its hidden layer transforms non-linearly from a previous layer to the next [21, 45]. Proposed by Hinto *et al*. [46], a DNN is trained via two phases: pre-trained, and fine-tuned processes [21, 47].

The auto-encoder is an unsupervised multi-layered neural network consisting both an encoder and a decoder network. Its encoder seeks to transform inputs data-points from a high unto a low-dimension via an encoding function f_{encoder} as in (1) where x^m is a data point, and h^m is the encoding vector obtained. Conversely, its decoder network seeks to reconstruct the function using f_{decoder} as in (2) with x^m as decoding vector from h^m. Thus, reverts the operations of the encoder [48]. Ojugo and Eboka [21] in Gilrot and Bengio [47] details specific algorithms for encoding and decoding functions respectively.

$$h^m = f_{encoder}(X^m) \tag{1}$$

$$X^m = f_{decoder}(h^m) \tag{2}$$

At the pre-training phase, *N* autoencoders can be stacked on to an N-hidden-layer so that with input accepted, the input layer and first hidden layer acts an encoder of the first auto-encoder. They are trained as thus, to minimize the reconstruction error. Training parameter(s) of the encoder are used to initialize first hidden layer before proceeding to second hidden layer. There, the first and second hidden layers are selected as encoder(s) and as in the earlier stage, the second hidden layer is initialized by the second trained auto-encoder. This process continues till the Nth auto-encoder is trained and initializes the final hidden layer. With all hidden layers stacked in the auto-encoder at each training *N*-times, they are thus regarded as pre-trained. This feat has proven to be significantly better than random initialization. It also achieves better generalization [20, 21, 46, 49].

Fine-tuning is a supervised phase that seeks to optimize a DNN's performance by retraining the network labeled training data. It computes the errors as a difference in real versus predicted values via back-propagated stochastic gradient descent (SGD), which randomly selects data, and iteratively updates gradient direction with the weight parameters. A merit of the SGD is that it converges faster and does not require the entire dataset. This makes it suitable for complex neural networks as given in (3) with E as loss function, *y* is label and *t* is output of the network [20, 21]:

$$E = \frac{1}{2}\sum_{j=1} M(y_i - t_i)^2 \tag{3}$$

The gradient of the weight *w* is obtained as a derivative of the error equation-so that an updated SGD is given by (4) with ŋ is step-size, *h* is number of hidden layers [20, 21]:

$$W_{ij}^{new} = W_{ij}^{old} - ŋ.\left(y_j - t_j\right).y_j\left(1-y_i\right).h_i \quad (4)$$

This process is optimized by the weights and threshold based on correctly labelled data. Thus, a DNN can learn accurately at its final output and direct thus, task all network parameters to perform correct classifications [20, 21].

2.2. The deep learning framework/algorithm

Deep learning solves tasks by: (a) dividing training data into clusters, computing center points from each cluster point, (b) each cluster is trained and scaled so that each DNN learns the various attributes of each subset, (c) the test data applies the previous cluster centers in its first step to detect outlier(s) by the pre-trained DNNs, and (d) output of each DNN is aggregated for the final result data/outliers [7, 20, 21]. Proposed solution is divided into 3-steps [10, 20, 21, 50]:

- Step 1 divides data into train and test clusters or partitions. DNN stores computed cluster centers, used as initialization center(s) to generate test datasets. Dataset attributes are formatted as data-points for selected parameters, and the data-points in the training dataset are aligned into groups of same class. To improve the performance of the DNN, model revises cluster numbers (to between 2 to 6) and sigma values (i.e. 0.1 to 1.0). The minimum distance from a data point to each cluster center is measured, and a data-point's nearness to a cluster, assigns it to that cluster-class. Training sets generated by clusters are taken up as input to DNNs. For training, the number of DNNs should equal the number of clusters. DNN architecture consists of five layers: an input, two hidden, a softmax and an output layer respectively. The hidden layers learn feats from each training subset, and the top layer is a five-dimensional output vector. Each training subset generated from the kth cluster center is regarded as input data to feed into kth DNN respectively. Trained sub-DNN models are marked sub-DNN 1 to k [20, 21, 50].
- Step 2 uses test dataset to generate k-datasets with the previous cluster center obtained from clusters in Step 1. The test sub-dataset are denoted as test 1 through test k [20, 21, 50].
- Step 3: The k-test data subsets are fed into k sub-DNNs, which were completed by the k training data subsets in Step 1. Output of each sub-DNN is integrated as final output and employed to analyse positive detection rates. Then, confusion matrix is used to analyse mining performance of generated rules [20, 21, 50].

Proposed DNN classifies data via back-propagation learning that maps input signals to low-dimensional space that seeks to discover patterns in the datasets. Algorithm is thus [20, 21, 50-56]:

Input: Dataset, cluster number, number of hidden-layer nodes HLN, number of hidden layers HL.
Output: Final prediction results

- Divide raw dataset into two components: training and a testing dataset.
  /*get the largest matrix eigenvectors and training data subsets*/
- Obtain cluster center and cluster results. Here, the clustering results are regarded as training data subsets.
  /*Train each DNN with each training data subset*/
- Learning rate, de-noising and sparsity parameters are set and the weight and bias are randomly initialized.
- HLN is set 40-nodes for first and 20-nodes for second hidden layer.
- Compute sparsity cost function
- Parameter weights and bias are updated
- Train k sub-DNNs corresponding to the training data subsets.
- Fine-tune the sub-DNNs by using backpropagation to train them.
- Final structure of trained sub-DNNs is obtained and labelled with each training data subset.
- Divide test dataset into subsets with SC. Cluster center parameters from the training data clusters are used.
- Test data subsets is used to test corresponding sub-DNNs, based on each cluster center between the testing and training data subsets
  /*aggregate each prediction result*/
- Results are generated by each sub-DNN, are integrated and the final outputs are obtained.
- return classification result=final output

2.3. Model optimization

A major issue in ML implementation is fine tuning features that live outside the model. These feats often influence the model's behavior-rippling effects acros hidden elements called hyper-parameters. Hyper-parameters are critical settings that can be tuned to control a model's behavior. They are parameters which are specific to the type of learning model we wish to optimize [7]. If a model seeks to learn these settings directly from a training dataset-there is the likelihood for the model to try to maximize these parameters-which will lead to over-fitting. And thus, will result in poor generalization [57, 58]. Major criteria of hyper-parameters are [59-61]:

- Learning rate is a hyper-parameter that controls how much and what weights needs to be adjusted on our network in lieu of gradient loss. The lower the value, the slower we travel on downward slope. Learning rate connotes how quickly a net abandons old beliefs for new ones. It can either be unsupervised and/or supervised learning. Also, with small/large learning rate, the net quickly differentiates between important feats and otherwise. Higher learning rate means the network can change and adapts flexibly, more easily. The model must be able to adequately adjust its learning rate to avoid over-fitting and overtraining.
- Batch size is the number of training utilized in one iteration. We can adopt one-of-three options: (i) batch mode where the iteration and epoch values are equal; (ii) mini-batch uses a batch size greater than one; (iii) stochastic in which the gradient and the neural network parameters are updated after each sample.
- Epoch measures the number of times all of the training vectors are used once to update the weights. It is a single step in training a network. Thus, if a network is trained on every training dataset samples in just one-pass, then an epoch is exhausted. A training may consist of more than one epochs. In batch training, all samples filter through the learning model simultaneously in one epoch with weights updated. Conversely, in sequential training, all weights are updated after each training.

3. RESULTS AND DISCUSSIONS

3.1. Data sampling

A major challenge is to get a dataset properly formatted for the task at hand. Dataset used for training (to fit the model) must be same for evaluating the model. Here, we adopt the Hochschule Coburg IDS datasets (CIDDS-2017)-a set of labeled anomaly-based IDS dataset, split as thus: training (70%) and testing (30%) [20, 21]. We then adopt 8-parameters to adjust weights and coefficients in minimizing errors as in Table 1:

Table 1. Selected features and their data types

Features	Format	Data Types
Source IP	a.b.c.d	Object
Source Port	Numeric	Integer
Destination IP	a.b.c.d	Object
Destination Port	Numeric	Float
Protocol	String	Object
Duration	H:M:S	Float
Packets	Numeric	Integer
Attack Name/Type	String	Object

3.2. Encoding schemes used

Unclassified and unformatted data are often ambuiguos, incomplete, rippled with noise, imprecise and inconsistent. Encoding seeks to filter the dataset, mapping it unto the required format the model can easily understand. To encode the selected feats, we transform our dataset using the feats of interest as in Table 1. This mode will seek to modulate the raw data unto the require dataset-so that data gathered from varying sources, is adequate for analysis. We employ data type in Pandas library displayed by listing 1 algorithm [20, 21].

```
Input: Selected Feature
Output: Converted Feature Data type
1.  Select Feature
2.  For each Selected Feature
3.  If Selected Feature is Non-Numerical then
4.  Generate Category Data type
5.  End if
6.  End For each
Listing 1: Algorithm to Convert Data type to Category
```

3.3. Parameter tuning

We modeled the network using 8-neurons at the input layer (a neuron for each feat). 2-neurons were used for output layer (a neuron for each possible class). The parameters for the deep learning are the number of epochs, the activation function, its learning rate and the hidden layer topology. We employed the rectified linear unit (ReLU) activation function with 500-epochs (though optimal values were reached at 100, 300 and 500 epochs taking into account accuracy and time to train the model). There is no best practice in selecting the number of hidden layers/neurons therein and using more hidden layer(s) grants the model greater capability to perform more complex function on the data [1, 2]. We seek minimum training error that will also result in the best fit, selecting the number of hidden layers (and neurons for each layer) was established via a trail-and-error method, and examining the results. The best possible number of layers was determined by running tests on a single layer with 1 to 20 neurons at the first instances-which yielded the greatest f-score with the least (constant) amount of training loss time. Addition of a second hidden layer of neurons from 1 to 20 yielded scores. Finally, the addition of a third hidden layer using the best possible number of neurons produced the greatest f-score and thus, was selected as the overall best possible hidden layer configuration. Results of the first hidden layer are seen in Table 2. Table 2 shows result of the first hidden layer with configuration of 9-neurons and f-score of 92% at 18^{th}-iteration and training loss of 1.140. F-score shows accuracy of each run-since we used an unbalanced dataset to train/test model with more records in normal class than in malicious class.

Table 2. First hidden layer configuration analysis

Hidden Layer	Precision	Recall	F1-Score	Iteration	Training Loss	Epoch
1	0.84	0.92	0.88	44	0.294	500
2	0.84	0.92	0.87	24	0.278	500
3	0.84	0.92	0.88	26	0.293	500
4	0.84	0.92	0.88	9	0.501	500
5	0.89	0.55	0.64	19	1.496	500
6	0.94	0.94	0.92	18	1.400	500
7	0.86	0.53	0.63	4	2.230	500
8	0.90	0.84	0.86	16	2.071	500
9	0.92	0.93	0.92	18	1.140	500
10	0.92	0.92	0.90	16	1.779	500
11	0.88	0.91	0.89	7	2.134	500
12	0.91	0.92	0.89	8	2.320	500
13	0.87	0.87	0.87	13	2.006	500
14	0.92	0.92	0.90	8	1.970	500
15	0.92	0.92	0.90	5	1.730	500
16	0.85	0.85	0.85	10	1.540	500
17	0.90	0.84	0.86	15	1.440	500
18	0.91	0.92	0.90	8	2.320	500
19	0.92	0.93	0.90	14	2.160	500
20	0.91	0.91	0.91	5	1.772	500

Table 3 shows first layer having 9-neurons and others neurons varying from 1 to 20. With hidden layer of 9 and 11 neurons yielding f-score of 93% and training loss of 0.39. The second hidden layer is favored as it yields greater f-score. Table 4 shows third configuration with first and second layer having 9 and 11 nodes and varying third hidden layer. Best configuration is 9-11-14 neurons, yielding f-score of 92% with a training loss at 0.560.

3.4. Model evaluation

We use the accuracy, recall and error rate(s) to evaluate model performance as in (5) to (7):

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (5)$$

$$Recall = \frac{TP}{TP+FN} \quad (6)$$

$$Error\ Rate = \frac{FP+FN}{FP+TP+TN+FN} \quad (7)$$

On evaluating our parameters, result of the model is given in a confusion matrix and the classification report. The resulting classification report and confusion matrix is given in the Tables 5 and 6 respectively. Table 5 shows model has prediction accuracy of 94-percent (0.94) with an improvement rate of

97-percent. It also has a misclassification error rate of 41-percent for data inclusion that were not originally used to train the model.

Table 3. Second hidden layer configuration analysis

Hidden Layer	Precision	Recall	F1-Score	Iteration	Training Loss	Epoch
9, 1	0.84	0.92	0.88	25	0.293	500
9, 2	0.84	0.92	0.88	29	0.292	500
9, 3	0.91	0.92	0.91	15	0.583	500
9, 4	0.87	0.87	0.87	5	1.058	500
9, 5	0.92	0.92	0.90	13	1.628	500
9, 6	0.91	0.92	0.89	10	1.996	500
9, 7	0.84	0.92	0.88	24	0.281	500
9, 8	0.93	0.93	0.92	11	1.884	500
9, 9	0.92	0.92	0.89	12	1.590	500
9, 10	0.90	0.92	0.90	12	1.731	500
9, 11	0.95	0.94	0.93	14	0.390	500
9, 12	0.93	0.93	0.91	12	1.130	500
9, 13	0.91	0.92	0.91	20	1.929	500
9, 14	0.92	0.93	0.90	13	2.237	500
9, 15	0.94	0.94	0.92	7	1.765	500
9, 16	0.85	0.52	0.62	7	2.010	500
9, 17	0.94	0.94	0.94	6	1.620	500
9, 18	0.93	0.94	0.92	7	1.760	500
9, 19	0.86	0..74	0.79	13	2.059	500
9, 20	0.92	0.92	0.89	8	2.421	500

Table 4. Third hidden layer configuration analysis

Hidden Layer	Precision	Recall	F1-Score	Iteration	Training Loss	Epoch
9, 11,1	0.83	0.91	0.87	32	0.287	500
9, 11,2	0.91	0.92	0.89	6	1.592	500
9, 11, 3	0.83	0.91	0.87	29	0.280	500
9, 11, 4	0.90	0.91	0.90	16	1.564	500
9, 11, 5	0.92	0.92	0.90	18	0.741	500
9, 11, 6	0.93	0.92	0.89	21	0.282	500
9, 11, 7	0.92	0.93	0.90	6	1.322	500
9, 11, 8	0.90	0.86	0.88	6	1.239	500
9, 11, 9	0.90	0.91	0.90	7	1.886	500
9, 11, 10	0.88	0.91	0.89	8	0.623	500
9, 11, 11	0.92	0.93	0.91	5	2.000	500
9, 11, 12	0.86	0.83	0.85	11	2.370	500
9, 11, 13	0.86	0.83	0.84	8	2.350	500
9, 11, 14	0.93	0.92	0.92	15	0.560	500
9, 11, 15	0.93	0.93	0.91	8	1.204	500
9, 11, 16	0.94	0.94	0.92	8	1.730	500
9, 11, 17	0.87	0.54	0.63	12	1.730	500
9, 11, 18	0.93	0.94	0.93	6	1.850	500
9, 11, 19	0.93	0.93	0.90	9	0.660	500
9, 11, 20	0.92	0.92	0.90	28	1.180	500

Table 5. Classification report before pre-processing test dataset

	Precision	Recall	F1-Score	Support
0	0.94	1.00	0.97	11411
1	0.90	0.27	0.41	1.059
Avg/Total	0.93	0.94	0.92	12.500

Also, Table 6 shows that 11.410 instances of the dataset were correctly classified. That is, results of the test dataset (with 12.500 points) show that we have 11.411 benign instances in the first class (label 0). The model successfully identified 11.410 correctly classified and identified benign instances as true-positives; but, 31-cases incorrectly identified benign instances were marked as false-positive. Similarly, on the second row, there were 1.059 malicious instances in second class (label 1); But, 776-incorrectly identified malicious instances were marked as false-negative, and 283 correctly identified malicious instances of them were marked as true-negative. These are further explained as: (i) For true positive, the model predicted positive and it was true; (ii) For true negative, the model predicted negative and it was true; (iii) For false

positive, the model predicted positive and it was false; and (iv) for false negative, the model predicted negative and it was false (as agreed by [62-64]).

Table 6. Confusion matrix report

	Actual	Values
Predicted	11.410	31
Values	776	283

Thus, we can say that the model predicts the results of either it's a normal attack or DDoS attack 92% accurately using the total value of the f-score. The neural network in Python may have difficulty converging before the maximum number of iterations allowed if the data is not standardized. For a more meaningful result, we decided to scale our test data. There are a lot of different methods for standardization of data, we will use the built-in StandardScaler for standardization. The result gotten after this process was done is slated as in Table 7 below:

Table 7. Classification report after pre-processing test dataset

	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	11.449
1	1.00	0.98	0.99	1.051
Avg/Total	1.00	0.99	0.99	12.500

From Table 8 (confusion matrix) i.e. the predicted results, using our test data with 12.500 points we have 11.449 benign instances in the first class (label 0). Out of this, the model successfully identified 11.449 correctly identified benign instances as a True Positive but there was no incorrectly identified benign instances which is supposed to be marked as False Positive. Similarly, looking at the second row, there were 1.051 malicious instances in second class (label 1) but 24 incorrectly identified malicious instances of them were marked as false negative and 1.027 correctly identified malicious instances of them were marked as true negative.

Table 8. Confusion matrix report

	Actual	Values
Predicted	11449	0
Values	24	1027

Thus, we can say that the model predicts the results of either it is a 'normal' attack or 'DDoS' attack 99% accurately using the total value of the f-score. In turn, this resulted in predicting 1.027 points as malicious samples and 11.449 points as normal samples from our test data. Furthermore, the standardization of our test data proved to be more efficient than the previous test run that was not standardized.

4. CONCLUSION

Our DNN model solution has a total of 56-rules with top rules found to have classification accuracy range [0.8, 0.96]. This implies that an estimated over 80% of the rules can adequately classify the dataset. Achieving a set of good rules, is much better than a single optimum rule. This increases the chances of detecting malicious data packets as well as also improves the generality of rules, providing the ability for new dataset and their corresponding generated rules to be added to the knowledgebase. The impact of the DDoS attacks to users requires a concerted effort to detect intrusion. Detection schemes simply filter through the network request, analyze them to decide which clients are uncompromised and compromised, and ultimately met out intended safety measures for further actions. Their performance can be hindered as premised on their error rate for incorrectly classified and unidentified data-points that scheme/model generates. An ideal scheme will correctly classify all request and packets with almost zero error rates of false positive/negative through tradeoffs between the number of false positives and false negatives.

REFERENCES

[1] Ojugo, A. A. and Eboka, A. O., "Inventory prediction and management in Nigeria using market basket analysis associative rule mining: memetic approach," *International Journal of Informatics and Communication Technology (IJ-ICT)*, vol. 8, no. 3, pp. 128-138, 2019.

[2] Ojugo, A. A. and Eboka, A. O., "Signature-based malware detection using approximate Boyer Moore string matching algorithm," *International Journal of Mathematical Sciences and Computing*, vol. 3, no. 5, pp. 49-62, 2019.

[3] Paxson, V., "An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 3, pp. 38-47, 2001.

[4] Nuruzzaman, T. M., Lee, C., Abdullah, M. F. A., Choi, D., "Simple SMS spam filtering on independent mobile phone," *Journal of Security and Communication Networks*, vol. 5, no. 10, pp. 1209-1220, 2012.

[5] Narayan, A. and Saxena, P., "The curse of 140 characters: Evaluating the efficacy of SMS spam detection on android," *SPSM '13: Proceedings of the Third ACM workshop on Security and privacy in smartphones & mobile devices*, 2013, pp. 33-42.

[6] Dadkhah, M., and Sutikno, T., "Phishing or hijacking? Forgers hijacked DU journal by copying content of another authenticate journal," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, vol. 3, no. 3, pp. 119-120, 2015.

[7] Ojugo, A. A., and D. O. Otakore, "Improved early detection of gestational diabetes via intelligent classification models: a case of Niger Delta," *Journal of Computer Science & Application*, vol. 6, no. 2, pp. 82-90, 2018.

[8] Ojugo, A. A., and Eboka, A. O., "Memetic algorithm for short messaging service spam filter text normalization and semantic approach," *International Journal of Informatics and Communication Technology (IJ-ICT)*, vol. 9, no. 1, pp. 13-27, 2020.

[9] Hasib, S., Motwani, M., Saxena, A., "Anti-Spam Methodologies: A Comparative Study," *International Journal of Computer Science and Information Technologies*, vol. 3, no. 6, pp. 5341-5345, 2012.

[10] Ojugo, A. A and Eboka, A. O., "Comparative evaluation for high intelligent performance adaptive model for spam phishing detection," *Digital Technologies*, vol. 3, no. 1, pp. 9-15, 2018.

[11] Trinity Security Services, "Retrieved from The Distributed Denial of Service Attack," 2003. [Online]. Available: http://archive.networknewz.com/networknewz-10-20030924 The Distributed Denial of Service Attack.html.

[12] Criscuolo P. J., "Distributed Denial of Service, Tribe Flood Network, and Stacheldraht CIAC-2319," *Lawrence Livermore National Laboratory*, 2010.

[13] Monowar H. Bhuyan, H. Kashyap, D. K. Bhattacharyya, and J. K. Kalita, "Detecting Distributed Denial of Service Attacks: Methods, Tools and Future Directions," *The Computer Journal*, vol. 57, no. 4, pp. 437-556, 2014.

[14] Munivara, P. K., Rama, M., Mohan, R. A., Venugopal, R. K., "DoS and DDoS Attacks: Defense, Detection and Traceback-A Survey," *Global Journal of Computer Science and Technology (GJCST)*, vol. 14, no. 7, pp. 15-31, 2014.

[15] Ojugo, A. A., E. Ben-Iwhiwhu, O. Kekeje., M. Yerokun., I. Iyawah, "Malware propagation on time varying networks: comparative study," *International Journal of Modern Education and Computer Science*, vol. 6, no. 8, pp. 25-33, 2014.

[16] Alexander S., "An anomaly intrusion detection system based on intelligent user recognition," Ph.D Thesis, University of Jyväskylä, Faculty of Information Technology, Finland, 2012.

[17] Ahmed, E., Clark, A., Mohay, G., "A novel sliding window based change detection algorithm for asymmetric traffic," *2008 IFIP International Conference on Network and Parallel Computing*, Shanghai, 2008, pp. 168-175.

[18] Cook, D., et al., "Catching Spam before it arrives: Domain Specific Dynamic Blacklists," *The proceedings of the Fourth Australasian Symposium on Grid Computing and e-Research (AusGrid 2006) and the Fourth Australasian Information Security Workshop (Network Security) (AISW 2006)*, Hobart, Tasmania, Australia, 2006.

[19] Todd B., "Distributed Denial of Service Attacks," *LinuxSecurity.com*, 2012. [Online]. Available: http://www.linuxsecurity.com/resource files/intrusion detection/ddos-whitepaper.html

[20] Ojugo, A. A., Eboka, A. O., "Empirical evaluation on comparative study of machine learning techniques in detection of DDoS," *Journal of Applied Science Engineering Technology and Education*, vol. 2, no. 1, pp. 18-27, 2020.

[21] Ojugo, A. A., Eboka, A. O., "Modeling solution of market basket associative rule mining approaches using deep neural network," *Digital Technologies*, vol. 3, no. 1, pp. 1-8, 2018.

[22] I. P. Okobah and A. A. Ojugo, "Evolutionary memetic models for malware intrusion detection: a comparative quest for computational solution and convergence," *International Journal of Computer Applications (IJCA)*, vol. 179, no. 39, pp. 34-43, 2018.

[23] Mirkovic, P. R., "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39-53, 2004.

[24] Kerk, J., "Estonia recovers from massive DDoS attack," computerworld.com, 2017. [Online]. Available: http://www.computerworld.com/s/article/9019725/Estonia_recovers_from_massive_DDoS_attack.

[25] Hamdan O. A., Rafidah M. N., Zaidan B. B., Zaidan A. A., "Intrusion Detection System: Overview," *Journal of Computing*, vol. 2, no. 2, 2010.

[26] Moore, G. M., et al., "Inferring Internet Denial-of-Service Activity," *ACM Transactions on Computer Systems*, 2006.

[27] Akella, A., Bharambe, A., Reiter, M., and Seshan, S., "Detecting DDoS attacks on ISP networks," *Proceedings of the Workshop on Management and Processing of Data Streams*, 2013, pp. 1-2.

[28] Apoorv, K., "How to deal with IP addresses in Machine Learning algorithms," 2016. [Online]. Available: www.quora.com/how-can-IP-addresses-in-machine-learning-algorithms-in-traffic-analysis-and-anomaly-detection.

[29] Eddy, W., "TCP SYN flooding Attacks and Common Mitigations," 2017. [Online]. Available: http://tools.ietf.org/html/rfc4987.

[30] Garcia-Teodoro P., Diaz-Verdejo J., Macia-Fernandez G., and Vazquez E., "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1-2, pp. 18-28, 2012.
[31] Gil, T. M. and Poletto, M., "MULTOPS: a datastructure structure for bandwidth attack detection," *SSYM'01: Proceedings of the 10th conference on USENIX Security Symposium*, vol. 10, 2011, pp. 13-17.
[32] Gayathri Shivaraj, M. S., et al., "Using Hidden Markov Model to detect rogue access points," *Security and Communication Networks*, vol. 3, no. 5, pp. 394-407, 2010.
[33] Thomas, R., Mark, B., Johnson, T., and Croall, J., "NetBouncer: Client-legitimacy-based high performance DDoS filtering," *Proceedings DARPA Information Survivability Conference and Exposition*, Washington, DC, USA, vol. 1, 2003, pp. 14-25.
[34] Nguyen, H.-V. A., "Proactive detection of DDoS attacks utilizing k-NN classifier in an Anti-DDoS framework," *International Journal of Electrical, Computer, and Systems Engineering*, vol. 4, no. 4, pp. 247-252, 2010.
[35] Wu, Y. C., Tseng, H. R., Yang, W., and Jan, R. H., "DDoS detection and traceback with decision tree and grey relational analysis," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 7, no. 2, pp. 121-136, 2011.
[36] Lee, K., Kim, J., Kwon, K. H., Han, Y., and Kim, S., "DDoS attack detection method using cluster analysis," *Expert Systems with Applications*, vol. 34, pp. 1659-1665, 2011.
[37] Ojugo, A. A., Eboka, A. O., E. Okonta., R. Yoro., F. Aghware., "Genetic algorithm rule-based intrusion detection system," *Journal of Emerging Trends in Computing Information Sciences*, vol. 3, no. 8, pp. 1182-1194, 2012.
[38] Karimazad, R. and Faraahi, A., "An anomaly based method for DDoS attacks detection using RBF neural networks," *2011 International Conference on Network and Electronics Engineering*, vol. 11, 2012, pp. 44-48.
[39] Jalili, R., Imani-Mehr, F., Amini, M., Shahriari, H. R., "Detection of distributed denial of service attacks using statistical pre-processor and unsupervised networks," *International Conference on Information Security Practice and Experience*, vol. 3439, 2015, pp. 192-203.
[40] Chen, Z., and Delis, A., "An inline detection and prevention framework for distributed denial of service attack," *The Computer Journal*, vol. 50, no. 1, pp. 7-40, 2017.
[41] Herrmann D., "A practical guide to security engineering and information assurance," 2012. [Online]. Available: www.auerbach-publications.com.
[42] Kiran S., "Exploring a novel approach for providing software security using soft computing systems," *International Journal of Security and Its Applications*, vol. 2, no. 2, pp. 51-58, 2008.
[43] Hwang, K., Dave, P., Tanachaiwiwat, S., "NetShield: Protocol anomaly detection with data-mining against DDoS attacks," *Sixth International Symposium on Recent Advances in Intrusion Detection*, Pittsburgh, PA, 2003, pp. 1-20.
[44] Ojugo, A. A., Eboka, A. O., "Cluster prediction model for market basket analysis: quest for better alternatives to associative rule mining approach," *International Journal of Artificial Intelligence*, vol. 9, no. 2, pp. 126-132, 2020.
[45] Bengio Y., Courville A., Vincent P., "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798-1828, 2013.
[46] Hinton, G., et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," in *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82-97, 2012.
[47] Glorot, X., and Bengio, Y., "Understanding the difficulty of training deep feedforward neural networks," *Journal of Machine Learning Research*, vol. 9, pp. 249-256, 2010.
[48] Ojugo, A. A., and Otakore, D. O., "Computational solution of networks versus cluster grouping for social network contact recommender system," *International Journal of Information and Communication Technology*, vol. 9, no. 3, pp. 185-194, 2020.
[49] Erhan, D., Bengio, Y., Courville, A., Manzagol, P.A., Vincent, P. and Bengio, S., "Why does unsupervised pre-training help deep learning?," *Journal of Machine Learning Research*, vol. 11, pp. 625-660, 2010.
[50] Ma, T., Weng, F., Cheng, J., Yu, Y., Chen, X., "A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks," *Sensors*, vol. 16, no. 10, pp. 1701, 2016.
[51] Agarwal, D., "What is training, learning and testing in machine learning?," 2017. [Online]. Available: www.quora.com/What-is-training-learning-and-testing-in-machine-learning?.
[52] Amit, S., "Understanding Recurrent Neural Networks," 2018. [Online]. Available: https://medium.com/mindorks/understanding-the-recurrent-neural-network-44d593f112a2.
[53] Bone R, Crucianu M., "Multi-step-ahead Prediction with Neural Networks," *de l'equipe RFAI*, pp. 97-106, 2016.
[54] Brownlee, J., "Machine Learning Mastery," 2018. [Online]. Available: machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/.
[55] Christos, S and Dimitrios, S., "Neural Networks," 2017. [Online]. Available: https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html.
[56] Michael N., "Using Neural Networks," *Chapter-1, neuralnetworksanddeeplearning*, 2018. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap1.html.
[57] Probst, Philip, Brend Bischi and Anne-Laure Boulesteix, "Tunability: Importance of Hyperparameters of Machine Learning Algorithms," *Journal of Machine Learning Research*, vol. 20, pp. 1-32, 2019.
[58] Rodriguez J., "Knowledge Tuning: Hyperparameters in Machine Learning Algorithms," 2017. [Online]. Available: http://medium.com/knowledge-tuning-hyperparameters-in-machine-learning-algorithms-67a31b1f7c88.
[59] Ring M., Wunderlich S., Grudl D., Landes D., Hotho A., "Flow-based benchmark data sets for intrusion detection," *Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS)*, 2017, pp. 361-369.
[60] Xavier A., "What are Hyperparameters in Machine Learning?," 2016. [Online]. Available: www.quora.com/What-are-Hyperparameters-in-Machine-Learning.

[61] A. A. Ojugo., J. Emudianughe., et al., "A Hybrid neural network gravitational search algorithm for rainfall runoff modeling and simulation in hydrology," *Progress in Intelligent Computing and Applications*, vol. 2, no. 1, pp. 22-33, 2013.
[62] Li-Chiou Chen, K. M., "Modelling Distributed Denial of Service Attacks and Defenses," *Project Summary*, 2002.
[63] Markus R., Wunderlich S. and Grudl D., "Flow-based benchmark data sets for intrusion detection," *Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS)*, 2017.
[64] Vasilev, I., "A Deep Learning Tutorial: From Perceptrons to Deep Networks," 2018. [Online]. Available: www.toptal.com/machine-learning/an-introduction-to-deep-learning-from-perceptrons-to-deep-networks.

BIOGRAPHIES OF AUTHORS

Arnold Adimabua Ojugo received BSc Computer Science from the Imo State University Owerri in 2000, MSc in Computer Science from the Nnamdi Azikiwe University Awka in 2005, and PhD in Computer Science from the Ebonyi State University Abakiliki in 2013. He currently lectures with the Department of Computer Science at the Federal University of Petroleum Resources Effurun, Delta State, Nigeria. His research interests: Intelligent Systems Performance, Machine Learning, CyberSecurity, IoT, and Graphs. He is also an Editor with the Progress for Intelligent Computation and Application, SceincePG etc. He is a member of: The Nigerian Computer Society, Computer Professionals of Nigeria and The International Association of Engineers. He is married to Prisca Ojugo with five children: Gregory, Easterbell, Eric, Elena and Elizabeth.

Rume Elizabeth Yoro received her BSc in Computer Science from the University of Benin Edo State in 2000, MSc in Computer Science from both Benson Idahosa University and the University of Benin respectively in 2009 and 2013. She currently lectures with the Department of Computer, Delta State Polytechnic Ogwashi-Uku Nigeria. Her research interests: Network Management, Computer Forensics/IoT and Machine Learning. She is a member of: Computer Professionals of Nigeria, Nigerian Computer Society, Computer Forensics Institute of Nigeria and Nigeria Women in Information Technolgy the International Association of Engineers. She is married to Fred Yoro with five children.