A DDoS attack Mitigation in IoT Communications Using Machine Learning

Hailye Tekleselase
Hailye83@gmail.com
School of Informatics, Wolaita Sodo University, Addis Ababa, Ethiopia

Abstract
Through the growth of the fifth-generation networks and artificial intelligence technologies, new threats and challenges have appeared to wireless communication system, especially in cybersecurity. And IoT networks are gradually attractive stages for introduction of DDoS attacks due to integral frailer security and resource-constrained nature of IoT devices. This paper emphases on detecting DDoS attack in wireless networks by categorizing inward network packets on the transport layer as either "abnormal" or "normal" using the integration of machine learning algorithms knowledge-based system. In this paper, deep learning algorithms and CNN were autonomously trained for mitigating DDoS attacks. This paper lays importance on misuse based DDOS attacks which comprise TCP SYN-Flood and ICMP flood. The researcher uses CICIDS2017 and NSL-KDD dataset in training and testing the algorithms (model) while the experimentation phase. accuracy score is used to measure the classification performance of the four algorithms. the results display that the 99.93 performance is recorded.

Keywords:
Distributed denial of Service; wireless networks; Machine Learning Algorithms; Transmission Control Protocol; CNN; network security

1. Introduction

The increase of IoT devices and computation devices have completed living relaxed and suitable for us due to the debauched and correct computation of our information. But, augmented incorporation and placement of linked devices also disclosures vital capitals to DDoS threats [1]. Technological growths in current years have complete it likely to connect a variety of devices to computer networks, which brings various benefits to users. But, with the increase of the technologies elaborate, the number of cyberattacks is also increasing, using more sophisticated means to incorrectly access sensitive information and to extort money or the already mentioned interruption of services. One such technology is the Internet of Things (IoT) [2].

The idea of the Internet of Things includes various devices, sensors, objects, and intelligent nodes that are able to function autonomously and communicate with each other without human intervention. Such IoT devices are able to deliver a number of valuable facilities and, cheers to sensors and actuators, provide various data in real-time. In many cases, however, devices in the field of the Internet of Things, in particular, contain various software bugs brought in from the factory that make them vulnerable. Such vulnerabilities often allow attackers to perform various cyberattacks and compromise the security of the environment in which IoT devices are located [3].

Several defense mechanisms have been proposed in the past against DDoS attacks in IoT networks. They can be divided into two basic groups: traditional DDoS defenses and IoT-specific DDoS defenses. They fluctuate in terms of place and difficulty. While traditional DDoS defenses are applied to the target server and are fundamentally homogeneous, IoT-specific DDoS defenses are applied to IoT devices and are more complex, reflecting the heterogeneity of IoT devices. In both cases, detection techniques are used to detect abnormal activities in the network or host [4].

The rest of the paper is organized as follows: Section 2 presents related works; Section 3 elaborates the data set and describes the methodology followed in the research; Section 4 details the experimentation procedures, the result gotten and the observations from the results while Section 5 presents the conclusion of the work as well as highlighting the future work

2. Related work

According to [5] detection systems of network intrusion have traditionally been rule-based. Nevertheless, machine learning and statistical approaches have also made major contributions [5]. Machine learning have also proven to be effective in two main aspects of network security which are: feature engineering (i.e., the ability to extract the most important features from network data to assist model learning) [6] and classification. In security environment, classification tasks usually involve training both suspicious and benign data in order to create models that can detect known attacks [7].

The authors in [8] pointed out that steps such as collection of network information, feature extraction and

Manuscript received April 5, 2024
Manuscript revised April 20, 2024
https://doi.org/10.22937/IJCSNS.2024.24.4.20

analysis, and classification detection provide a means for building efficient software-based tools that can detect anomalies such as software-defined networking (SDN). Another study [9] provides a thorough classification of DDoS attacks in terms of detection technology. The study also emphasizes how the characteristics of the network security of an SDN defines the possible approaches to setting up a defense against DDoS attacks. Similarly, [10] have explored this area too. In other approaches to DDoS defense, [4] propose a scheduling based SDN controller architecture to effectively limit attacks and protect networks in DoS attacks.

The growth of cloud computing and IOT has inevitably led to the migration of denial-of-service attacks on cloud computing devices as well. Thus, cloud computing devices must implement efficient DDOS detection systems in order to avoid loss of control and breach of security [11]. Studies such as [12] aim to tackle this problem by determining the source of a DDOS attack using Trace (powerful trace) source control methods. Trace controlled such attack sources from two aspects, packet filtering and malware tracing, to prevent the cloud from becoming a tool for DDoS attacks. Other studies such as [13] approach the problem of filtering by using a set of security services called filter trees. In the study, XML and HTTP based DDOS attacks are filtered out using five filters for detection and resolution. Detection based on classification has also been proposed and a classifier system for detection against DDOS TCP flooding attacks was created [14].

These classifiers work by taking in an incoming packet as input and then classifying the packet as either suspicious or otherwise. The nature of an IP network is often susceptible to changes such as the flow rate on the network and in order to deal with such changes, self-learning systems have been proposed that learn to detect and adapt to such changes in the network [15].

Many of the existing models for DDoS detection have primarily focused on SYN-flood attacks and haven't been trained to detect botnet attributes. More studies are thus needed where models are trained to detect botnet as botnet becomes the main technology for DDoS organization and execution [16]. Botnet DDoS attacks infect multitude of remote systems turning them to zombie nodes that are then used for distributed attacks. In detecting botnet DDoS attacks, authors in [17] used a deep learning algorithm to detect TCP, UDP and ICMP DDoS attacks. They also distinguished real traffic from DDoS attacks, and conducted in-depth training on the algorithm by using real cases generated by existing popular DDoS tools and DDoS attack modes. Also, [18] proposed a DDoS attack model and demonstrated that by modelling different allocation strategies. The proposed DDoS attack model is applied to game planning strategies and can simulate different botnet attack characteristics.

According to [19] "DDoS detection approaches can operate in one of the following three modes: supervised, semi-supervised and unsupervised mode. For the detection approach in supervised mode, it requires a trained dataset (or a classifier) to detect the anomalies, where the trained dataset includes input variables and output classes. The trained dataset is used to get the hidden functions and predict the class of input variables (incoming traffic instances). This mode is similar to a predictive model. For example, Classification techniques comes under the category of supervised data mining" [20]. "For the Approaches that work in the semi-supervised mode, they have incomplete training data i.e., training data is only meant for normal class and some targets are missing for anomaly class" [21]. Unlike supervised and semi-supervised learning, unsupervised machine learning algorithms do not have any input-output pairs but the algorithm is trained such that it can accurately determine the unknown data point. The following subsections further discusses the unsupervised learning algorithms we used in this work. current effort that goes to detect IoT based attacks proposed MQTT transaction-based features Mustafa et al. (2019). But the authors used features based on the TCP protocol analysis, which do not provide sufficient information on the MQTT protocol parameters. In contrast, our proposed UDP features are based on unsupervised machine learning which can successfully detect and distinguish such attacks including the unknown attacks.

2.1 Autoencoder

According to [6], Autoencoder can be defined as a neural network that detect an identity function that can recreate the input data with high accuracy by using back propagation. The Autoencoder has two main parts (cf. Figure 1), an encoder that compresses the input into a dimensional latent subspace that is lower and a decoder that recreate the input from this latent subspace. The encoder and the decoder, can be defined as transitions ϕ and ψ such that:

$$\phi: X \rightarrow F$$

$$\psi: F \rightarrow X$$

$$\phi, \psi = \operatorname{argmin} \phi, \psi [\| X - (\psi \circ \phi) X \|]^2 \quad (1)$$

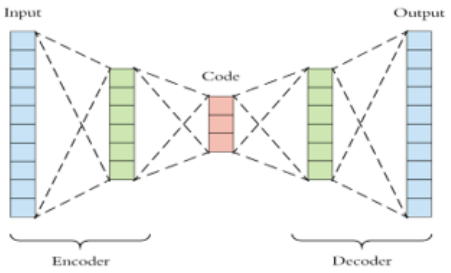

Figure 1 A generic representation of the autoencoder.

Autoencoder have been suggested for feature engineering in the network security environment, e.g., to learn important features of malware classification [22] and also for detecting DDOS attacks on application layer [23]. The linear analogue of Autoencoder is the principal component analysis (PCA) which is based on statistical techniques.

2.2 Restricted Boltzmann Machines (RBM)

The authors in [24] stated that Boltzmann machine (BM) is a bidirectionally connected network of stochastic processing units. BMs are commonly used to learn important features of an unknown probability distribution based on samples from the distribution. However, the training process of the BM is usually computationally intensive and tedious. The restricted Boltzmann machine attempts to solve the training problem of BMs by imposing key restrictions on the architecture of the BM. The BM is a fully connected network of bidirectional nodes where each node is connected to every other node. The RBM on the other hand is presented as a relatively smaller network of bidirectional nodes with the restriction that nodes on the same layer are not connected to each other horizontally [24].

The restricted Boltzmann machine is a generative model that is used to sample and generate instances from a learned probability distribution. Given the training data, the goal of the RBM is to learn the probability distribution that best fits the training data. The RBM consists of m visible units V= (V_1, V_2…., V_ m) and n hidden units H= (H_1, H_2…., H_ n) arranged in two layers.

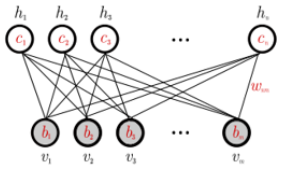

Figure 2 Restricted Boltzmann machine showing the visible and hidden units

The visible units lie on the first layer and represent the features in the training data (see Figure 2). Usually, one visible unit will represent one feature in an example in the training data. The hidden unit's model and represent the relationship between the features in the training data. The random variables (V, H) take on values (v, h) $\in$ [0,1] ^m for continuous variables and the underlying probability distribution in the training data is given by the Gibbs distribution p (v, h) = 〚1/z〛 ^ (-E (v, h)) with the energy function in equation 1;

$$E(\mathbf{v},\mathbf{h}) = -\sum_{m}^{n}\sum_{m}^{n} w_{ij}h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i \qquad (2)$$

In equation 2, w_ij are real valued weights associated with v_ j and h_ i, and b_ j and c_ i are real valued bias terms associated with units i and j respectively. The contrastive divergence learning algorithm is one of the successful training algorithms used to approximate the log-likelihood energy gradient and perform gradient ascent to maximize the likelihood [24].

After a successful training, the RBM should be able to represent the underlying probability distribution of the training data and when presented with unseen examples, the RBM should be able to generate similar representations to the example provided.

2.3 K-Means

The K-means algorithm takes the full dataset consisting of multiclass data points, then clusters the datapoints into separate clusters to the best of its ability; this classification occurs when you feed in the input and the model assigns the input into one of the computed clusters. Given a set of observations (x_1, x_2, x_3, ..., x_ n), where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k (≤ n) sets S = {s_1, s_2, ..., s_ k} so as to minimize the within-cluster sum of squares (WCSS) (i.e., variance).

2.4Expectation-Maximization (EM)

The authors in [25] stated that EM algorithm is used for solving mixture models that assume the existence of some unobserved data. Mathematically, the EM algorithm can be described as follows; given the statistical model that generates a set of observed data X, latent data Z, unknown parameters θ and the likelihood function L (θ ; X) =p (X, Z ⊣ | θ), the maximum likelihood of the unknown data θ is determined by maximizing the marginal log-likelihood of the observed data X using equation 3:

$$L(\boldsymbol{\theta}; \boldsymbol{X}) = p(\boldsymbol{X} \mid \boldsymbol{\theta}) = \int p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\theta}) dZ \quad (3)$$

The expectation and maximization steps are defined as follows:

$$Q\left(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}\right) = \boldsymbol{E}_{\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta}^{(t)}}[\log L(\boldsymbol{\theta}, \boldsymbol{X}, \boldsymbol{Z})] \quad (4)$$

$$\boldsymbol{\theta}^{t+1} = arg\ max_{\boldsymbol{\theta}}\ Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}) \quad (5)$$

In the expectation step, the likelihood of the unknown parameters is computed as the log-likelihood of the known parameter estimates, while in equation 4 the maximization step is used to select the new value that maximizes the log-likelihood given the estimates from equation 5.

In our research, we differ from current work (e.g. [26]) in two ways. First, we work with unsupervised machine learning methods using both normal and suspicious network data to train. Second, we made use of dimension reduction methods such as K-means clustering with PCA, Expectation maximization, Restricted Boltzmann Machine and Autoencoder (where K-means and EM are both trained using normal and suspicious data; RBM and AE was trained on only suspicious data), all these methods were not only for feature engineering [22] but for classification as well.

3. METHODOLOGY

A DDoS attack temporarily or indefinitely constraints the availability of a network resource to its intended users. The challenge then for the network administrator is to deploy DDoS detection systems that are capable of analyzing incoming packets to the transport layer. These detection systems may then determine if these incoming packets are suspicious or benign. In the following subsections, we present the design methodology for a DDoS detection system that makes use of unsupervised machine learning algorithms. The problem is therefore to design and train four efficient unsupervised machine learning systems that are capable of detecting a DDOS attack on the transport layer.

3.1 The Datasets

In order to train the unsupervised machine learning algorithms used in this study, we sourced for modern DDOS attack datasets including the following datasets.

The first dataset is the DDOS evaluation dataset (CICDDoS2019) [27]. The full dataset consists of both reflection and exploitation based DDOS attacks in the form of both suspicious and benign network packets. The dataset is further grouped into TCP and UDP based attacks.

The second dataset is the Mirai dataset created by [28]. Mirai is specific type of botnet malware that overrides networked Linux devices and successfully turns them into bots used for distributed attacks such as DDOS. The Mirai dataset consists of 80,000 SYN-flood instances and 65,000 UDP-lag attacks on a security camera IOT device.

Finally, the third dataset is the BASHLITE botnet attack dataset on a webcam IOT device and is also provided by [28]. Similar to Mirai, BASHLITE is a botnet malware for distributed attacks on networked devices. The BASHLITE dataset consists of 110,000 SYN-flood instances and 100,000 UDP-lag attacks. Both Mirai and BASHLITE are open-source malware that can be used for academic research purposes.

. 3.2 Data pre-processing

The dataset largely consists of numerical values, so the pre-processing steps are minimal. The most important pre-processing action taken was to normalize the values in the dataset using the standard minmax normalization expressed below in equation 6.

$$x' = \frac{x - min(x)}{max(x) - min(x)} \quad (6)$$

3.3 Selected Unsupervised Learning Algorithms

The selected unsupervised machine learning algorithms employed in this work are explained below.

Autoencoder

The autoencoder is made up of two feed forward neural networks that mirror each other in terms of the number of layers and the number of nodes. Recall that the goal of the autoencoder model is to learn the latent space of an encoded vector that can be used to efficiently reconstruct the input vector. Thus, the goal of the encoder is to encode the input vector to a lower dimensional vector space [3].

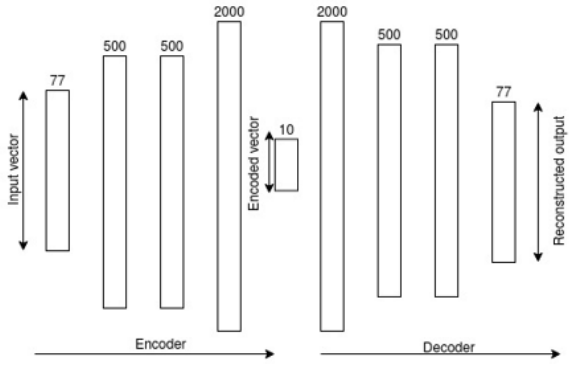

Figure 3 The Autoencoder architecture

Each neuron in the hidden layers of the encoder and decoder make use of the Rectified Linear Unit (ReLu) activation function. The hyperparameters selected for the autoencoder model are outlined as follows;

Batch size: A batch size of 2048 is selected. Number of epochs: An epoch number ranging between 10-20 is initialized with 70 showing to be the most optimal as we shall see from the results.

Loss: The mean squared error loss function is used. Optimizer: The Adaptive (ADAM) algorithm is selected. ADAM is the state-of-the-art optimizer for deep neural networks (Schneider et al, 2019).

Betas: These are ADAM optimizer coefficients used for computing running averages of gradient and its square (0.5, 0.999). Learning rate (0.0002).

3.3.2 Restricted Boltzmann Machine

The RBM used for this project is a two-layer RBM with an architecture as described in section 2.2 and figure 2. The dataset consists of continuous variables scaled between 0 and 1 so therefore we model the RBM as a continuous variable model with the hidden units and visible units taking on values between (v,h) $\in$ [0,1]^m where m is number of visible units Similar to the autoencoder algorithm, we use the reconstruction error to define the classification task for the RBM. The parameters selected for the RBM include:

Number of units: We always set the number of hidden and visible units to be the same according to the number of features present in the training data. That is, for instance for the CICDDoS2019, the number of hidden and visible units for the RBM is 77.

Training algorithm: The k-step contrastive divergence algorithm with Gibb's sampling is used for training the algorithm with k=10.

Training Epoch: An epoch of 10 is selected, experimental results show that increasing the epoch beyond 10 does not improve training results.

3.3.3. K-Means

1.1 The K-Means clustering algorithm has relatively fewer parameters to select. The default "pure" version of the K-means algorithm is used as opposed to variants such as Elkan's K-Means [29] where triangle inequality is used. The parameters for the algorithm are outlined as follows;

1.2 3.3.4 Expectation-Maximization

1.3 The expectation-maximization algorithm is setup with similar parameters as the k-means algorithm and in fact the software implementation of the two algorithms in the sci-kit learn machine learning framework borrow from each other. However, the core implementations are different and the parameters for the expectation maximization algorithm include:

1.4 Number of components: This is the number of clusters to be estimated and is set to two because of the binary classification task of suspicious or benign.

1.5 Number of iterations: The number of iterations is like the epoch of the autoencoder where they both define the number of training iterations to run the algorithm. A default value of 300 is used.

1.6 Covariance type: The covariance parameter defines the structure of the covariance matrix with respect to each component or cluster. The "full" covariance is chosen where each cluster has its own covariance matrix and has been shown to achieve the best results.

3.4 Evaluating Model Performance

3.4.1 Accuracy

In machine learning parlance, the task of the determining whether or not an incoming packet is suspicious or benign is known as classification. For the K-Means and EM algorithm, the clustering of a feature point together with highly similar feature points is in itself, a form of classification. We can evaluate these models by quantifying how accurate their classification of a packet is with respect to its clusters. A simple net accuracy score of the predicted class compared to the actual class gives us an empirical quantification of the model's performance.

. We can define a reconstruction error that describes the difference between the reconstructed output and original input vector. The reconstruction error is defined as the mean squared error (MSE) in equation 7 as follows:

1.7 $$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (7)$$

Where y_i is the original input vector and $\hat{y}_i$ is the reconstructed output vector? The mean squared error is computed over all the output of the model. Ideally, it is preferable to have a mean squared error close to zero. However, depending on the size of the values in the predicted output, a mean squared error within the range of 2-5 decimal places is acceptable. Representing the reconstruction error as the mean squared error allows one to know when the model is presented with an input that is very far off from what was contained in the training set.

Thus, if for instance the autoencoder is trained on a dataset comprising only of benign packets, whenever a benign packet is presented to the autoencoder, we expect that the reconstructed output should be quite similar and therefore the reconstruction error should be low. However, if this same model is presented with a suspicious packet that is fairly different from the features of benign packet, then we should expect the reconstruction error to be quite high. The same logic can be applied to the restricted Boltzmann machine

1.8 With this formulation established, it is easier to frame the classification problem using the autoencoder and RBM. Where in our example, a low reconstruction error means the packet is benign, while a high reconstruction error means the packet is suspicious. Using these predictions,

we can then compute the accuracy much like we did with the K-Means model. The accuracy score simply calculates a ratio of the number of correctly classified packets over the incorrectly classified packets.

3.4.2 Normalized Mutual Information (NMI)

The NMI provides a means of evaluating the clustering performance of the algorithm by comparing the correlation between the predicted class and the target class. If the predicted and target data are represented as two separate distributions, then we can also apply the NMI to determine performance of non-clustering algorithms.

3 Experimental Results

In this session, we present the experimental results for each model across all datasets. The results are presented in subsections, with each subsection dedicated to a model. For the Autoencoder and Restricted Boltzmann Machine, their subsections consist of plots showing the training and test loss, a table summarizing the performance across the datasets and a detailed discussion of the results. For the rest of the models, they do not optimize a loss function and so only the summary tables and a detailed discussion of the results were presented. Performance evaluations are also carried out using the accuracy and Normalized Mutual score. The innovation of this work lies in the exact detection of the anomality behaviour of the nodes. DDoS attacker tried to affect network in its different forms. The basic nature of DDoS attacker is to flood the network with a large number of packets and then exhaust the network.

4.1 Comparison Result and Analysis

The efficiency of planned attack detection system is also assessed against five existing works of [Bellingerite al., (2020)], [Almsgiving et al., (2017)], [Yan Naing Soe,2020] (Naeem Firdous Syed,2020) and [Ashrafi et al., (2013)] who have addressed intrusion detection against DDoS attack using KDD dataset. compares the detection accuracy of the proposed work against the existing works. Recently, [Vellinga et al., (2020)] had proposed Taylor Elephant Herd Optimisation based Deep Belief Network to detect DDoS attacks and attained a classification accuracy of only 83%. Further, [Almsgiving et al., (2017)] had realised Random forest classifier to attain a detection accuracy of 93.77% while [Ashrafi et al., (2013)] work comprising of extended Classifier System with Artificial Neural Network (ANN-XCS) demonstrated a detection accuracy of 98.1% against the proposed work that have implemented the optimization techniques with unsupervised machine learning to achieve a detection accuracy of 99.93%.

This paper focusses on detecting DDoS attack in IoT networks by classifying incoming network packets on the transport layer as either “Suspicious” or “Benign” using unsupervised machine learning algorithms. In this work, two deep learning algorithms and two clustering algorithms were independently trained for mitigating DDoS attacks. We lay emphasis on exploitation based DDOS attacks which include TCP SYN-Flood attacks and UDP-Lag attacks. We use Mirai, BASHLITE and CICDDoS2019 dataset in training the algorithms during the experimentation phase.

4.3 Autoencoder training and test results

Figure 4 shows the desired behavior of the backpropagation training algorithm where the training and validation loss decrease steadily and in unison as the training epoch increases. It is important to point out that the autoencoder is trained to reconstruct SYN-Flood data, meaning it should be unable to reconstruct benign data. We chose the SYN-Flood data for training because there were more instances than the benign data. The same choice is made for the UDP-Autoencoder model, where we train it on the UDP-Lag data instead of on benign UDP data.

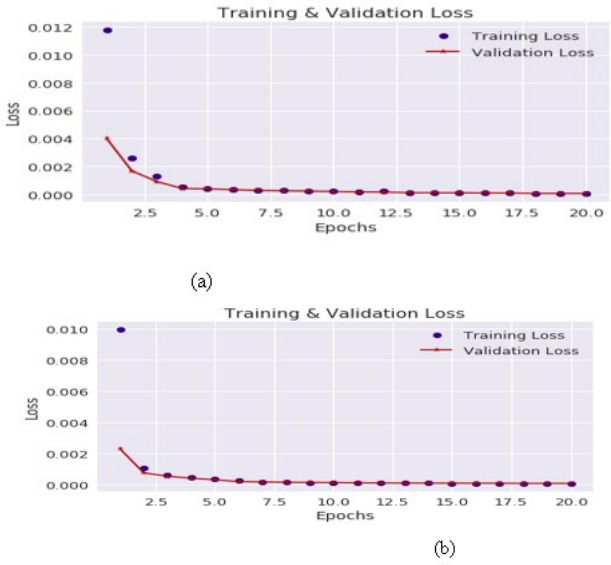

Figure 4 Autoencoder training and validation loss on the CICDDoS2019 SYN-Flood (a) and UDP-Lag (b).

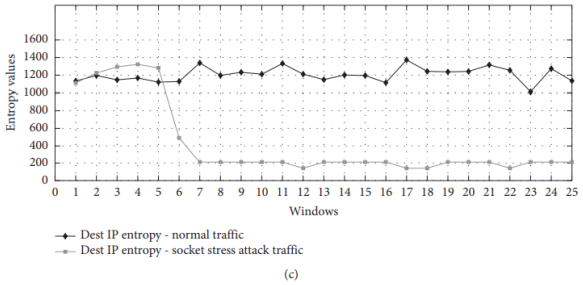

Figure 5: Autoencoder training and validation loss on the Mirai SYN-flood (a) and UDP-Lag data (b)

The training and validation loss for the BASHLITE loss shown in Figure 6 is less steep than that of the Mirai loss. Table 1 shows the accuracy of the autoencoder model

across all datasets, here we can see that the model has the highest accuracy on the BASHLITE dataset hence the reason why the loss is less steep and flattens out quickly by epoch 10 and 14 respectively.

Table 1 Test Accuracy and Normalized Mutual Information score for the autoencoder models on the SYN-Flood and UDP-Lag across all datasets.

Data	Accuracy (%)	NMI
CICDDoS2019 SYN-Flood	0.8945	0.5363
CICDDoS2019 UDP-Lag	0.8617	0.4216
Mirai SYN-Flood	0.9744	0.6211
Mirai UDP-Lag	0.9621	0.5733
BASHLITE SYN-Flood	0.9933	0.9927
BASHLITE UDP-Lag	0.9921	0.9822

In table 1, we present the accuracy and NMI scores for the autoencoder model. These scores were determined based on the formulation described in section 3.4. The result show that the autoencoder model performs best on the BASHLITE SYN-Flood data with a higher accuracy of 99%. In general, the autoencoder performs better on the Mirai and BASHLITE datasets than that of the CICDDoS2019 dataset.

4.4 Restricted Boltzmann Machine

The restricted Boltzmann machine performed poorly on the CICDDoS2019 dataset but the training process evolved smoothly with the loss dropping as the epoch increased.

Table 2 Test Accuracy and Normalized Mutual Information score for the Restricted Boltzmann machine model on the SYN-Flood and UDP-Lag across all datasets.

Data	Accuracy (%)	NMI
CICDDoS2019 SYN-Flood	0.5651	0.1919
CICDDoS2019 UDP-Lag	0.5089	0.1103
Mirai SYN-Flood	0.6067	0.1639
Mirai UDP-Lag	0.7797	0.3895
BASHLITE SYN-Flood	0.6709	0.2506
BASHLITE UDP-Lag	0.6210	0.1007

The RBMs performance on the BASHLITE dataset is similar to its performance on the Mirai data, still, the overall performance is much lower than that of the Autoencoder. The results indicate that the RBM is less suited for the kind of precise reconstruction of the continuous input value that is easily achieved by the autoencoder.

4.5 K-Means training and test results

The K-Means algorithm is not a gradient based learner so we cannot bother ourselves with iterative plots such as those presented for the autoencoder and RBM models. Also, the K-Means algorithm is trained on a distribution that contains a mixture of both suspicious and benign features. The model's accuracy and NMI are shown in Table 3.

Table 3 Test Accuracy and Normalized Mutual Information score for the K-Means model on the SYN-Flood and UDP-Lag training and validation data.

Data	Accuracy (%)	NMI
CICDDoS2019 SYN-Flood	0.7538	0.1949
CICDDoS2019 UDP-Lag	0.7139	0.1427
Mirai SYN-Flood	0.7636	0.0912
Mirai UDP-Lag	0.7478	0.1387
BASHLITE SYN-Flood	0.6451	0.1059
BASHLITE UDP-Lag	0.6823	0.1306

From the results, we observe that the K-Means model performs relatively poorer as compared to the autoencoder model but it performs better on average when compared to RBM. However, and once again, one can see some slight disparity in performance on the SYN-Flood data compared to UDP-Lag data owing to the variance amongst the features in these datasets.

The NMI scores for the K-Means model are relatively low too with well below average correlations. Although one should interpret the accuracy and NMI scores independently, the low NMI scores for the K-Means discourages one from being too optimistic about the model's performance.

4.6 Expectation-Maximization training and test results

The Expectation-Maximization algorithm performs better than the k-means algorithm on average with its highest accuracy being 80% on the Mirai UDP-lag data (Table 4).

Table 4 Test Accuracy and Normalized Mutual Information score for the EM model on the SYN-Flood and UDP-Lag training and validation data

Data	Accuracy (%)	NMI
CICDDoS2019 SYN-Flood	0.7096	0.1144
CICDDoS2019 UDP-Lag	0.6759	0.1446
Mirai SYN-Flood	0.7030	0.2771
Mirai UDP-Lag	0.8051	0.2901
BASHLITE SYN-Flood	0.7636	0.3074
BASHLITE UDP-Lag	0.7575	0.2678

Although, the EM algorithm performs better than the k-means algorithm the performance is still poor compared to the autoencoder. The clustering algorithms struggle with large high-dimensional and continuous data. The maximization step in the EM algorithm gives it an edge over the k-means algorithm in this aspect. Where the k-means algorithm struggles to compute a centroid from high dimensional continuous data, with low variance as is the case with Mirai and BASHLITE, the EM algorithm models the problem probabilistically instead, optimizing for the log-likelihood of the latent variables.
Tables 5 and 6 below summarize the accuracy and NMI scores across all datasets for all models, highlighting the autoencoder's superiority for machine learning task.

Table 5 Summary of the Accuracies across all datasets and all models.

Dataset/Model	Autoencoder	Restricted Boltzmann machine	K-Means	Expectation-Maximization
CICDDoS2019 SYN-Flood	0.8945	0.5651	0.7538	0.7096
CICDDoS2019 UDP-Lag	0.8617	0.5089	0.7139	0.6759
Mirai SYN-Flood	0.9744	0.6067	0.7636	0.7030
Mirai UDP-Lag	0.9621	0.7797	0.7478	0.8051
BASHLITE SYN-Flood	0.9933	0.6709	0.6451	0.7636
BASHLITE UDP-Lag	0.9921	0.6210	0.6823	0.7575

Table 6 Summary of the Normalized Mutual Information score across all datasets and all models.

Dataset/Model	Autoencoder	Restricted Boltzmann machine	K-Means	Expectation-Maximization
CICDDoS2019 SYN-Flood	0.5363	0.1919	0.1949	0.1144
CICDDoS2019 UDP-Lag	0.4216	0.1103	0.1427	0.1446
Mirai SYN-Flood	0.6211	0.1639	0.0912	0.2771
Mirai UDP-Lag	0.5733	0.3895	0.1387	0.2901
BASHLITE SYN-Flood	0.9927	0.2506	0.1059	0.3074
BASHLITE UDP-Lag	0.9822	0.1007	0.1306	0.2678

5. CONCLUSIONS

The unsupervised machine learning models have been developed and trained on both SYN-Flood and ICMP flood DDOS datasets. The training and test results both show that the deep learning autoencoder model performs better in the classification of incoming packets as suspicious or benign. Over the past decade, deep learning algorithms have established themselves as the state-of-the-art machine learning algorithms. Our results show that in the unsupervised machine learning space, the deep learning algorithm also outperforms traditional clustering algorithms such as the K-Means and Expectation-Maximization algorithm as well as other generative deep learning models such as the Restricted Boltzmann machine. However, when comparing unsupervised machine learning algorithms, one must be careful to formalize the performance evaluation problem properly. The project shows that it is possible to frame the autoencoder model as a classification algorithm using the value of the reconstruction error and that it is possible to apply this formulation efficiently to difficult problem domains such as network packet analysis. Once proper formulations are established, the accuracy score can then be used to evaluate both models fairly. Although the autoencoder model is clearly the superior model, the DDOS-Detection class we developed provides methods that allow one to perform network packet classification using either the autoencoder model or the Expectation-Maximization model. The simulation results show that the DDOS-Detection tool built around these models can achieve a net accuracy of as high as 99%. Future studies should aim to replicate results in a larger system to detect compromised end-points and also ensure

that algorithms are current by possible retraining approaches to handle abnormalities in network performance.

Acknowledgment

The authors would like to express their deep thanks to Dr. John Mike for his valuable advice.

Acknowledgment

The authors would like to express their cordial thanks to Dr. Mitsuo Ohta for his valuable advice.

References

[1] Shirazi, "Evaluation of anomaly detection techniques for scada communication resilience," *IEEE Resilience Week,* 2016.

[2] N. Mirai, "mirai-botnet," 2016. [Online]. Available: https://www.cyber.nj.gov/threat-profiles/botnetvariants/mirai-botnet. [Accessed 31 December 2019].

[3] H. Zhou, B. Liu and D. Wang, "Design and research of urban intelligent transportation system based on the Internet of Things," *Internet of Things,* pp. 572-580, 2012.

[4] S. Lim, S. Yang and Y. Kim, "Controller scheduling for continued SDN operation under DDoS attacks," *Electronic Letter,* pp. 1259-1261, 2015.

[5] A. Buck and E. Govan, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials,* vol. 18.2, 2016.

[6] P. Baldi, "Autoencoders, Unsupervised Learning, and Deep Architectures," *Proceedings of ICML works hop nuns supervised transfer learning,* 2012.

[7] R. Doshi, N. Althorp and N. Feemster, "Machine Learning DDoS Detection for Consumer Internet of Things Devices," *IEEE Deep Learning and Security Workshop,* 2018.

[8] Q. Yan, F. Yu and Q. Gong, "Software defined networking and Distributed denial of service attacks in cloud computing environments," *IEEE Communications Survey & Tutorial,* no. 18, pp. 602-622, 2016.

[9] N. Z. Bawany, J. A. Shamsi and K. Salah, "DDoS Attack Detection and Mitigation Using SDN," *Arabian Journal for Science & Engineering,* no. 2, pp. 1-19, 2017.

[10] B. Kang and H. Choo, "An SDN-enhanced load-balancing technique in the cloud system[J].," *Journal of Supercomputing,* pp. 1-24, 2016.

[11] O. Saniya and D. M. Choo, "Distributed denial of service (DDOS) resilience in cloud," *Journal of Network & Computer Applications,* pp. 147-165, 2016.

[12] H. Luo, Z. Chen and J. Li, "Preventing Distributed Denial-of-Service Flooding Attacks with Dynamic Path Identifiers[J]," *IEEE Transactions on Information Forensics & Security,* pp. 1801-1815, 2017.

[13] U. Dick and T. Schiffer, "Learning to control a structured-prediction decoder for detection of HTTP-layer DDOS attackers," in *Machine Learning*, 2016, pp. 1-26.

[14] Z. Gao and N. Ansari, "Differentiating Malicious DDoS Attack Traffic from Normal TCP Flows by Proactive Tests[J]," *Communications Letters IEEE,* pp. 793-795, 2006.

[15] K. Briceno, A. Rurality and A. Gurov, "Detecting the Origin of DDoS Attacks in OpenStack Cloud Platform Using Data Mining Techniques[M]// Internet of Things," *Smart Spaces, and Next Generation Networks and Systems,* 2016.

[16] N. Hoque, D. Bhattacharyya and J. Kavita, "Botnet in DDoS Attacks: Trends and Challenges[J]," *IEEE Communications Surveys & Tutorials,* pp. 1-1, 2015.

[17] A. Saeed, R. E. Overbill and T. Ridzik, "Detection of known and unknown DDOS attacks using Artificial Neural Networks," *Neurocomputing,* pp. 385-393, 2016.

[18] S. Rama nauseate, N. Geranin and A. Cents, "Modelling influence of Botnet features on effectiveness of DDoS attacks[J]," *Security & Communication Networks,* pp. 2090-2101, 2015.

[19] C. Barghini, M. J. Kavita, S. Singh and D. K. Bhattacharyya, "Anomaly based DDoS attack detection," *International Journal of Computer Applications,* pp. 35-40, 2015.

[20] A. Aggarwal and A. Gupta, "Survey on data mining and IP traceback technique in DDos attack," *International Journal of Engineering and computer science,* vol. 4(6), pp. 12595-12598, 2015.

[21] G. Naima and M. Hemal Atha, "Effective approach towards intrusion detection system using data mining technique," *Egyptian Informatics Journal,* vol. 15(1), pp. 37-50, 2014.

[22] Y. A. Mahmood, "Autoencoder-based feature learning for cybersecurity applications," *International Joint Conference on Neural Networks (IJCNN),* 2017.

[23] S. Yadav and S. Subramanian, "Detection of Application Layer DDoS attack by feature learning using Stacked Auto Encoder," *International Conference on Computational (ICCTICT),* 2016.

[24] A. Fischer and C. Ige, "An introduction to restricted Boltzmann machines. In Libero American congress on pattern recognition," *Springer, Berlin, Heidelberg,* pp. 14-36, 2012.

[25] V. G. Rydin and G. Volcano, "An expectation maximization method to estimate a rank-based," 2017.

[26] D. Ferrierite, "Extreme Dimensionality Reduction for Network Attack Visualization with Autoencoders," (IJCNN), 2019.

[27] I. Sharfuddin, A. H. Lashkar, S. Haka and A. Ghobadi, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy," International caravan conference on security (ICCST). IEEE, pp. 1-8, 2019.

[28] Y. Maidan, M. Bandana, Y. Mathur, Y. Mirsky and Shabtai, "Network based detection of iot botnet attacks using deep autoencoders," IEEE Pervasive Computing, pp. 12-22, 17(3).

[29] C. Elkan, "Using the triangle inequality to accelerate k-means," ICML-03, pp. 147-153, 2003.

[30] R. Bhatia, S. Benno, J. Esteban, T. V. Lakshman and J. Grogan, "Unsupervised machine learning for network-centric anomaly detection in IoT.," in the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks.