Towards DDoS Attack Detection using Deep Learning Approach

Article *in* Computers & Security · April 2023

DOI: 10.1016/j.cose.2023.103251

CITATIONS
56

READS
1,369

2 authors:

Sharmin Aktar
Saint Mary's University, San Antonio
6 PUBLICATIONS **72** CITATIONS
SEE PROFILE

Abdullah Yasin Nur
University of New Orleans
17 PUBLICATIONS **202** CITATIONS
SEE PROFILE

Towards DDoS Attack Detection using Deep Learning Approach

Sharmin Aktar, Abdullah Yasin Nur

Department of Computer Science

University of New Orleans, New Orleans, LA, US

saktar@uno.edu, ayn@cs.uno.edu

Abstract

Due to the extensive use and evolution in the cyber world, different network attacks have recently increased significantly. Distributed Denial-of-Service (DDoS) attack has become one of the fatal threats to the Internet, where attackers send massive amounts of packets to the target system to make online systems unavailable to legitimate users. Proper attack detection measurement is crucial to defend against these attacks. This paper proposes a deep learning-based model using a contractive autoencoder to detect anomalies. We train our model to learn the normal traffic pattern from the compacted representation of the input data, and then apply a stochastic threshold method to detect the attack. Three renowned Intrusion Detection System datasets have been used for evaluation—CIC-IDS2017, NSL-KDD, and CIC-DDoS2019. We have assessed the results against a basic autoencoder and other deep learning approaches to show our model efficacy. Our results indicate a successful intrusion detection of the proposed method with an accuracy ranging between 93.41% and 97.58% on the CIC-DDoS2019 dataset. Moreover, it achieved an accuracy of 96.08% and 92.45% on NSL-KDD and CIC-IDS2017 datasets, respectively.

Keywords:
Denial of service attack, Distributed denial of service attack, Intrusion detection systems, Deep neural networks, Anomaly detection

1. Introduction

The fast advancement of cyberspace makes it crucial to identify intrusions that breach network security. With the evolvement of different sophisticated intrusion techniques, attackers can damage the network system within a short period. For example, Amazon Web Services (AWS) experienced a 2.3 Tbps Distributed Denial of Service (DDoS) attack in February 2020 [13]. More recently, Google reported that one of their cloud customers was targeted with 46 million requests per second during peak time in June 2022 [34]. According to Kaspersky Lab [35], the number of DDoS attacks hit a record high in Q4 2021, and the trend is increasing significantly. Therefore, the Intrusion Detection System (IDS) is a vital tool to ensure the data's availability, confidentiality, and reliability [15].

An Intrusion Detection System (IDS) is defined as a system or software that discovers abnormal activity via monitoring the network [12]. Usually, there are two types of IDSs: the Network Intrusion Detection System (NIDS) and the Host Intrusion Detection System (HIDS). In NIDS, the anomalous traffic is detected utilizing all packet metadata and contents across the network. In contrast, HIDS performs intrusion detection on a particular endpoint and protects it against internal and external threats. IDSs can be classified either as Signature-based or Anomaly-based, depending on their detection methods. Signature-based detection works best for identifying known threats, where it detects malicious traffic based on predefined rules. Anomaly-based IDS detects abnormal behavior by modeling normal behavior via pattern extraction. Typically, Anomaly-based IDS can uncover complex and unknown attacks, thus, performs better than signature-based IDS.

Denial-of-Service (DoS) attack is one of the most harmful cyberattack types where perpetrators aim to exhaust the target's system by flooding traffic until the target is inaccessible to intended users. Typically, these attacks are executed by flooding the aimed machine with superfluous traffic before the target becomes unresponsive. Distributed Denial of Service (DDoS) attack, a variant of the DoS attack, is more pernicious than a DoS attack. It is more difficult to defend because of employing many compromised machines to deluge the victim with spurious traffic. These attacks cause a substantial financial loss in the industries by impeding licit users' access via exhaustion of the victim server. According to Kaspersky Lab research, the global financial impact of a DDoS attack is above $120K for small and medium-sized businesses, and over $2M for enterprises per attack on average [39]. Hence, efficacious detection methods are essential to protect online services from attackers. This work introduces a novel deep learning model for detecting network traffic attacks.

Most existing IDSs often fail to detect unknown attacks because they rely on predefined patterns and signatures, although they achieve high detection accuracy for the known ones. Besides, they experience high false-positive cases, which circumscribe their real-life deployments. To address these issues,

conventional machine learning techniques have been extensively used for intrusion detection. However, almost all traditional machine learning models fail to detect the attack from an enormous dataset since they follow shallow learning methods [16, 17]. Deep learning models are helpful for complex, large-scale network environments, which have the potential to extract distinctive self-generated features without using handcrafted feature extractions [22]. As a result, most researchers in this field focus on developing deep learning-based IDS.

The attack detection system presented in this paper employs the principles of contractive autoencoder. Our model targets to learn necessary information from the input data and reconstructs the given traffic sample using the intermediate compressed hidden layers. It only uses non-anomalous instances from the training data to extract the regular traffic pattern on the network. We have utilized a stochastic anomaly threshold approach based on the reconstruction loss to distinguish between attack and non-attack instances. The choice of this threshold depends on the fact that non-attack samples will produce a low reconstruction error while a high error will be generated when using the attack data. Rather than using a fixed threshold value, we have run our model several times with different thresholds to select the best one.

Our method can precisely detect known and unknown attacks with the selected optimal threshold. We evaluate the proposed method using three datasets, CIC-IDS2017, NSL-KDD, and CIC-DDoS2019 [2, 3, 9]. We achieve the highest accuracy of our model as 97.58% on the CIC-DDoS2019 dataset, whereas the model shows an accuracy of 96.08% and 92.45% on the NSL-KDD and CIC-IDS2017 datasets, respectively. We demonstrate the outperformance of our approach over widely used deep learning models, including the Basic Autoencoder (AE), Variational AE (VAE), and Long Short Term Memory AE (LSTM AE). Our results show that the proposed model achieves significant improvement. Specifically, the results show that the proposed method outperforms other methods significantly in the CIC-DDoS2019 dataset, where our accuracy range is [93.41% – 97.58%] whereas the accuracy range for LSTM AE [70.46% – 95.40%], VAE [70.46% – 90.20%], and Basic AE [75.82% – 93.09%]. Additionally, our accuracy for the CIC-IDS2017 dataset is 92.45%, whereas the accuracy for LSTM AE is 88.69%, VAE 88.73%, and Basic AE 89.84%. Moreover, our accuracy for the NSL-KDD dataset is 96.08%, whereas the accuracy for LSTM AE is 93.2%, VAE 93.68%, and Basic AE 90.45%.

The main contributions of this paper can be summarized as follows:

(a) We propose a deep learning method based on the contractive autoencoder model for attack detection for Distributed Denial of Service (DDoS) attacks. Our model uses a semi-supervised learning approach where only non-attack instances are used while training our model. The model reconstructs the input with less reconstruction error for the non-attack data at the output layer. In the case of an anomalous instance, the trained model gives a high error rate, failing to regenerate it properly. We have utilized this reconstruction error to distinguish between normal and anomalous instances.

(b) We employed a stochastic approach for identifying anomalous instances based on reconstruction loss. Instead of relying on a fixed threshold value, we conducted multiple runs of our model with varying thresholds to choose the optimal one.

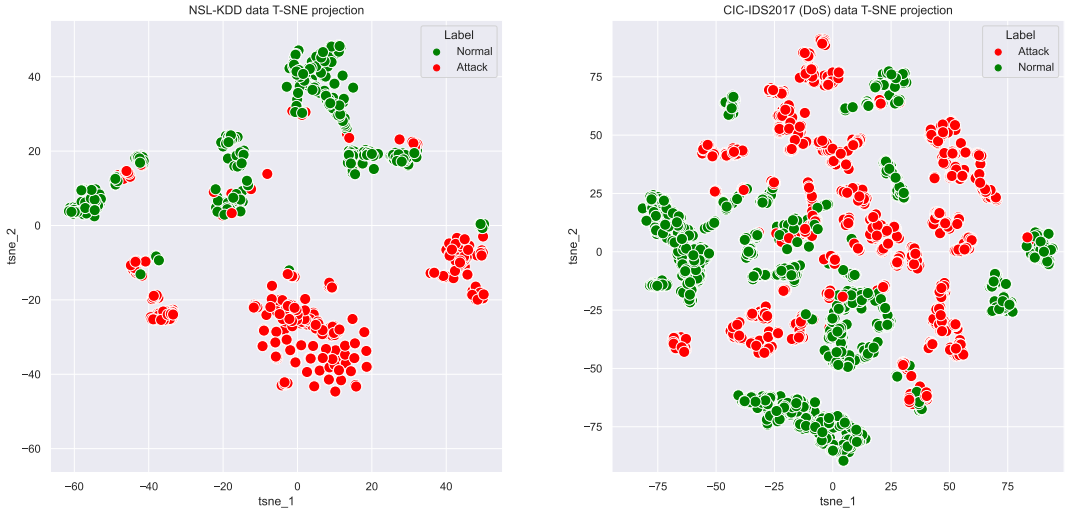
(c) The performance of the proposed method is evaluated by using three benchmark datasets, NSL-KDD, CIC-IDS2017, and CIC-DDoS2019. Our results show that the proposed method can achieve up to 97.58% accuracy in detecting anomalies. Additionally, we compare our method with other deep learning models and show that our model surpasses those models significantly.

The rest of the paper is arranged as follows. Section 2 presents the background and motivation. In Section 3, various network attack detection methods have been reviewed. Section 4 provides our methodology. Our experimental results have been demonstrated in Section 5. Lastly, Section 6 provides discussion for our key findings and Section 7 concludes the paper.

2. Background and Motivation

Network traffic data usually contains high-dimensional features. Therefore, proper data visualization techniques are needed to select a suitable IDS model. There are existing techniques for displaying these types of data. In this work, we have shown the data arrangement of our evaluated samples using t-Distributed Stochastic Neighbor Embedding, known as t-SNE [29]. The t-SNE models high-dimensional data by the low-dimensional point such that the neighbor structure among the data points remains maintained. Figures 1a and 1b display a t-SNE visualization of the NSL-KDD dataset and the CIC-IDS2017(DoS) dataset, where the red color corresponds to the attack samples, and the green color represents the normal samples. It shows that normal and attack samples share some identic feature spaces. Thus, linear separation of these two classes seems impossible, demonstrating the complication of intrusion detection problems in network traffic. Additionally, we can visualize the high degree of nonlinearity in our data sample by observing the Andrews curve [30]. This curve helps to visualize a high-dimensional data structure representing a high-dimensional feature space as a finite Fourier series. Figures 2a and 2b show the Andrews curve for the NSL-KDD and CIC-IDS2017 (DoS) dataset, where each curve corresponds to an observation in the dataset. The figures show that the normal and attack data curves are entangled, showing the existence of nonlinearity in the feature space. Hence, shallow machine learning models cannot distinguish the attack samples because of the nonlinearity in such datasets. Moreover, it is reported that the performance of traditional techniques falls short in capturing complex structures in data [41]. Therefore, deep learning based techniques have garnered significant interest among researchers [17, 27, 41].

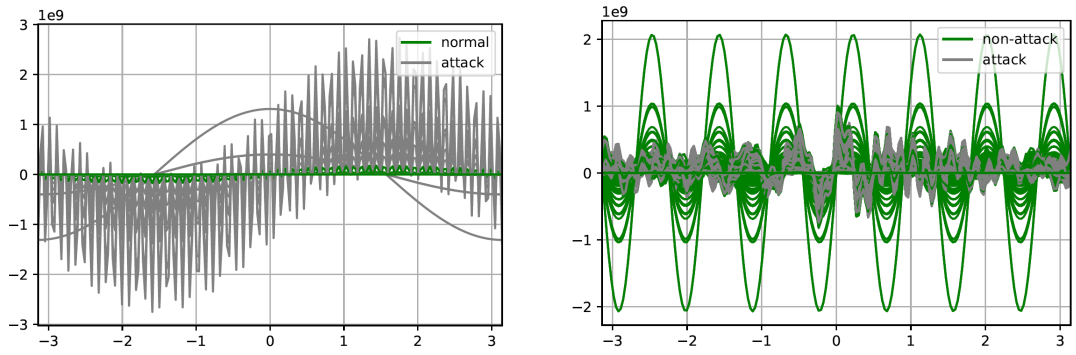
Deep learning based anomaly detection models can be categorized as supervised, unsupervised, and semi-supervised. Supervised learning models learn to make predictions or classify new data by being trained on labeled data. Even though the performance of supervised models can be relatively higher than



(a) t-SNE visualization for NSL-KDD

(b) t-SNE visualization for CIC-IDS2017

Figure 1: t-SNE visualization



(a) Andrews curve for the NSL-KDD dataset

(b) Andrews curve for the CIC-IDS2017(DoS) dataset

Figure 2: Andrews Curve Visualization

other models, they are less popular due to the shortage of labeled training data in practice [41]. Additionally, supervised models have problems with training set imbalance since normal instances are far more than anomaly instances [18]. Unsupervised learning models address these problems by using unlabeled data for training. The models aim to uncover patterns or structures in the data instead of making predictions or categorizing new data points. Unsupervised methods tend to be vulnerable to noise and data corruption, leading to decreased accuracy compared to supervised or semi-supervised techniques [41]. Finally, semi-supervised learning models fill the gap between supervised and unsupervised learning models by leveraging the existing label of a single class to separate outliers. Note that obtaining a normal data class is easier than getting the anomalous data classes, where obtaining anomalous data classes is often a costly process in many practical domains [15]. Using a one class labeled data can significantly enhance performance compared to unsupervised methods [41]. Therefore, we used semi-supervised learning model by using non-anomalous instances from the training data to extract the regular traffic pattern on the network.

Complex networks contain an extensive amount of features. To simplify, researchers use autoencoders (AE) to reduce the feature set from a high dimension to a lower dimension. However, this leads to loss of informative features. Researchers have proposed various variants of autoencoders in the literature to address this issue [1]. Basic autoencoder consists of an encoder function that maps the input data to a lower-dimensional latent space and a decoder function that maps the latent representation back to the original input space. Variational autoencoder (VAE) is a generative model that uses an autoencoder architecture to learn a probabilistic mapping from the input data to a latent space and from the latent space back to the input space. VAEs are used for generating new data and unsupervised representation learning. One major problem of VAE is that VAEs are susceptible to failing to identify outliers when the anomalies in the test data have the same distribution as the anomalies in the training data [43]. Sparse Autoencoder (SAE) is a type of autoencoder that is trained to reconstruct the input data with a smaller number of neurons in the hidden layer being activated to learn a compact representation of the input data. The sparsity constraint is imposed through a sparsity regular-

ization term in the loss function, which encourages the model to activate only a small number of neurons to reconstruct the input data. SAE models may experience over-specification problems in case the sparsity constraints are not optimized properly, which leads to poor generalization performance. Concrete Autoencoder is designed explicitly for discrete or categorical data by adding a concrete relaxation to the encoding process [42]. The use of continuous relaxation in the encoding process enables the model to handle discrete variables in a smooth and differentiable manner, facilitating training through gradient-based optimization methods. However, the stochastic nature of concrete autoencoders can lead to issues with reproducibility across multiple runs. Moreover, the scalability issues of continuous relaxation based techniques pose difficulties in their application and may not be feasible for use in complex networks [44].

Reconstruction Error (RE) and Reconstruction Probability (RP) are two popular scoring measures for anomaly detection in neural networks. RE evaluates the discrepancy between the input data and the data generated by the autoencoder. This discrepancy is commonly quantified using a loss function, like the mean squared error. The objective of the autoencoder is to minimize the reconstruction error, thus ensuring that the reconstructed data resembles the original input data as closely as possible. On the other hand, RP calculates the chance of a particular input data point being reconstructed by the autoencoder. The autoencoder calculates a probability distribution across all possible reconstructions of the input data and the reconstruction probability represents the most probable reconstruction. This metric is frequently utilized in variational autoencoders for creating new data samples that resemble the input data. VAE determines the reconstruction probability by estimating both the mean and variance of the output dimensions. It is reported that the joint optimization of both mean and variance in the VAE creates the variance shrinkage problem and underestimation of variance [45]. These problems can lead to poor generalization of new data.

In this paper, we propose Deep Contractive Autoencoder (DCAE) model which is designed to learn a hierarchy of encoder-decoder pairs, where each pair is responsible for reconstructing a different level of abstraction in the input data. The incorporation of the contractive regularization term in the deep contractive autoencoder enhances its robustness against slight variations in the input data, thereby boosting its performance in noisy or real-world scenarios. We have utilized a stochastic anomaly threshold approach based on the reconstruction error. The proper reconstruction of normal traffic makes our model efficient in separating the attack data from the non-attack data since our detection method is based on the fact that the anomalous sample will deviate from the normal one by generating higher reconstruction loss.

3. Related Work

Many approaches have been suggested to defend systems against DDoS attacks [7, 19, 20, 21]. DDoS defense mechanisms can be categorized as attack detection, attack reaction, and attack source identification. Attack detection techniques analyze the incoming packets and identify attacks in case of an anomaly in the observed traffic [18, 36]. Attack reaction techniques aim to mitigate the impact of attacks by applying resource management [11]. The last category is attack source identification, where the victim site aims to detect the attacker's position even if the attacker spoofs its Internet Protocol (IP) address [14, 33, 37]. This paper falls into the first category, where we aim to detect anomalies by applying a deep learning method based on the contractive autoencoder model.

The notion of Intrusion Detection System (IDS) was primarily introduced by James Anderson at the National Security Agency in 1980 [23]. His idea comprised several tools for reviewing system audit trails to detect abnormal activities. Later, many studies have been examined, which have made significant progress for IDS. Machine learning (ML) is suitable for intrusion detection due to its data modeling and prediction capability. Moreover, deep learning based models have further advancements in attack detection accuracy. This section discusses current attack detection methods suggested by various researchers.

Elsayed et al. [15] proposed a model using Long Short Term Memory (LSTM) autoencoder combining One-class Support Vector Machine (OC-SVM) algorithms to enhance performance. They utilized only normal instances during training to learn the normal traffic behavior and to achieve the compressed representation of the input data. Then OC-SVM approach was applied to the reduced representation to achieve better classification outcomes. They evaluated their model using the recent InSDN dataset. In another work, Elsayed et al. [24] proposed a deep learning framework using the LSTM autoencoder for network attack identification. They trained their model using only the normal samples from the dataset. By observing the distribution of the reconstruction loss in the training data, they picked the optimal threshold value, providing the best accuracy for attack detection. They compared the model with widely used classical ML algorithms and some modern techniques in the NSL-KDD dataset for evaluation purposes.

The authors in [25] pointed out the over-generalization problem with the AE-based anomaly detection method. To overcome it, they suggested a model using the Memory-Augmented Deep Autoencoder (MemAE). Their model includes an additional memory module between the encoder and decoder, which targets reconstructing the attack samples similar to the normal ones during training. To separate anomalous data from the benign sample, they chose a threshold from the reconstruction loss percentiles of normal samples providing the best F1- score, the harmonic mean of the precision and recall, in the validation set. They evaluated the classification results on NSL-KDD, UNSW-NB15, and CIC-IDS2017 datasets, based on the AUROC value and F1-Score. The model achieved an AUROC value of at least 0.9 for all datasets. At the same time, it achieved the highest F1-Score of 95% for the NSL-KDD dataset. The authors also compared the results with an AE and a one-class SVM (OCSVM) model.

Ding and Zhai [16] proposed an Intrusion Detection System (IDS) based on Convolutional Neural Network (CNN) with multi-stage features. In the model, they created a deep layer

of input features using three stacked stages, containing a convolution layer followed by max pooling for feature extraction. Before the final layer, two dense layers accompanied by one softmax layer were added and concatenated with the staged features. Lastly, a softmax classifier was used to extract the target. The authors compared the results with conventional machine learning and deep learning methods on the full NSL-KDD dataset. They showed the outperformance of their model compared with other methods.

Farahnakia and Heikkonen [12] suggested an IDS model (DAE-IDS) containing four autoencoders where each autoencoders output in the current layer proceeds to the next one as an input. Moreover, they followed a greedy layer-wise training manner which means an autoencoder is trained once the previous training is finished. After training the autoencoders, a softmax classifier was used to identify the attack instances. They evaluated the model's performance using a widely known dataset, KDD-CUP'99. The experimental results showed a high detection rate of 94.53% which outperformed other methods.

Aygun and Yavuz [28] suggested two anomaly detection models using autoencoder and denoising autoencoder. They introduced a novel stochastic anomaly threshold determination method to enhance their model's performance. They trained their models using semi-supervised learning on the recent NSL-KDD dataset and analogized the result with other singular and hybrid models. The proposed methodology achieved an accuracy of 88.28% and 88.65% for AE and DAE models, respectively.

Wang et al. [31] proposed a novel IDS model combining stacked contractive autoencoder (SCAE) and support vector machine (SVM) algorithm. They utilized SCAE to extract necessary low-dimensional features from raw input data automatically. The model's training process contained three stages: unsupervised pretraining, unrolling, and supervised fine-tuning. Once the training was finished, the SVM classifier was used to detect anomalous samples from the extracted features. The researchers conducted some experiments to evaluate the detection performance of the model on two well-known datasets, NSL-KDD and KDD'99. The approach achieved an accuracy of 88.73% for binary classification tasks on the NSL-KDD dataset.

Kim et al. [32] designed an IDS model based upon a Convolutional Neural Network (CNN) and evaluated its performance through comparison with a Recurrent Neural Network (RNN). For the experimental purpose, they utilized two popular datasets, KDD CUP 1999 dataset (KDD) and CSE-CIC-IDS2018, mainly focusing on the DoS category. Furthermore, they proposed an optimal CNN design for performance enhancement.

The method called DeepDefense is proposed for detecting DDoS attacks by using a deep neural network [27]. The suggested model looks for the repeated pattern representing attack and locates them in a long-term traffic pattern, formulating them as a sequence classification problem. Their approach combined different neural network models: CNN, LSTM, and GRU. They evaluated the DeepDefense model on an extracted part from the large-scale dataset, ISCX2012, containing the DDoS attack. The experimental results surpassed the traditional machine learning algorithms showing an error rate reduction of 5.4% on the larger data set.

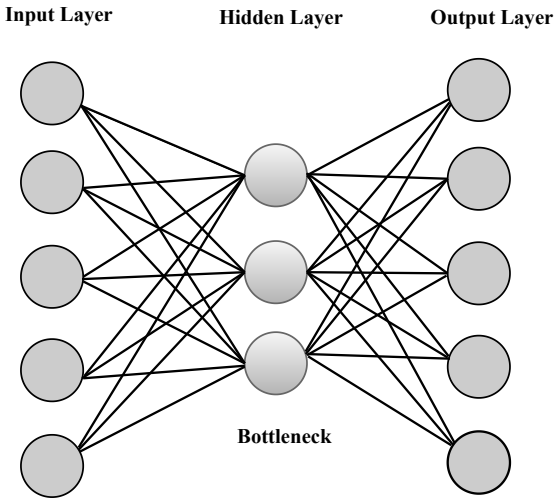


Figure 3: Structure of a Basic Autoencoder

Our model considers a deep learning approach based on a contractive-autoencoder for detecting attacks in network traffic. We propose a novel framework where we devise our model as a binary classification problem, in which each data sample will be classified into either normal or attack categories.

4. Methodology

In this section, we present the attributes of our framework and the system architecture of our suggested intrusion detection model.

4.1. Autoencoder

The proposed attack detection model uses autoencoder attributes based on a semi-supervised learning approach. Autoencoder was first introduced [1] as a dimensionality reduction approach, where the output of the encoder denotes the lessened representation, whereas the decoder's output targets to recreate the original input from the encoder's representation through a cost function minimization process. An autoencoder aims to reconstruct its input vectors as the target output by extracting the most representative data features.

An autoencoder consists of a single input and output layer with at least one hidden layer. The input and output layer of the autoencoder always contains the same unit except in the hidden layer. Typically, the hidden layer's size is less than the input or output layer.

There are two stages called encoding and decoding in the autoencoder. In the encoding phase, an autoencoder compresses the input with fewer units by using its hidden layer. During the decoding part, it attempts to rebuild the initial input by utilizing the encoded representation from the hidden layer.

In case an autoencoder consists of only one hidden layer, it is called the basic autoencoder. Figure 3 illustrates the structure

of the basic autoencoder. During the encoding stage of a basic autoencoder, an encoding function f is used, which converts the initial input x into a latent representation h, generally stated as code or bottleneck. It has the following form:

$$h = f(x) = \sigma(Wx + b), \tag{1}$$

where σ is usually a nonlinear activation function, such as a logistic sigmoid function or a rectified linear unit. A weight matrix W and a bias vector b are the encoding parameters utilized during training the autoencoder. Those are initialized randomly and then updated iteratively through backpropagation. The decoder function g is utilized in the decoding phase, which converts the hidden representation h into a reconstruction x' with the same shape of x:

$$x' = g(h) = \sigma'(W'h + b'), \tag{2}$$

where σ' is the decoder's activation function, b' and W' represent the bias vector and weight matrix, respectively.

The training of an autoencoder aims to minimize the reconstruction error on a training dataset, D_n by finding parameters $\theta = \{W, b, b'\}$ [1]. It corresponds to the minimization of the following objective function:

$$\mathcal{J}_{AE}(\theta) = \sum_{x \in D_n} L(x, g(f(x))), \tag{3}$$

where L is the reconstruction loss, defined as the gap between the original and reconstructed input. The typical loss function used in the autoencoder is squared error $L(x, x') = \|x - x'\|^2$, which is the measurement of similarity between x and x'. When the inputs range between 0 to 1, the cross-entropy loss is used, which is represented in the below equation:

$$L(x, x') = -\sum_{i=1}^{d_x} x_i \log(x'_i) + (1 - x_i) \log(1 - x'_i) \tag{4}$$

4.2. Contractive Autoencoder

A simple autoencoder tries to compress the information of a given data while keeping the reconstruction loss as less as possible. In contrast, a contractive autoencoder aims to learn useful information from the input data, reducing the representation's sensitivity towards training the input data [1]. It makes the hidden representation, $f(x)$ of the autoencoder, invariant to small perturbation of the training inputs x, which is ensured by penalizing its sensitivity towards that input. The penalized term follows the Frobenius norm of the Jacobian $J_f(x)$ for the encoder activation sequence of the input [1]. The Frobenius norm, also known as the Euclidean norm, is the square root of the sum of the absolute squares of elements of an m x n matrix. The Jacobian matrix is the matrix of all first-order partial derivatives of a vector-valued function.

The penalty term is formally defined as the squared Frobenius norm of the Jacobian matrix of partial derivatives associated with the encoded features:

$$\|J_f(x)\|_F^2 = \sum_{ij} \left(\frac{\partial h_j(x)}{\partial x_i}\right)^2 \tag{5}$$

The objective function of Contractive AutoEncoder (CAE) obtained with the regularization term of equation 5 is as follows:

$$\mathcal{J}_{CAE}(\theta) = \sum_{x \in D_n} L(x, g(f(x))) + \lambda \|J_f(x)\|_F^2 \tag{6}$$

where the first part represents the reconstruction loss and the second one denotes the penalty or the regularizer. λ corresponds to the penalty coefficient utilized to adapt the consonance of the regularization in the objective function.

4.3. Our Model

This section introduces our proposed framework based on the deep learning model for network attack detection. We can state the intrusion detection problem as assigning a label indicating a normal or attack sample based on reconstruction attributes on a given dataset. Our model targets to learn necessary information from the input data and reconstructs the given traffic sample. It only uses non-anomalous instances from the training data to extract the regular traffic pattern on the network. The model reconstructs the input with less reconstruction error for the non-attack data at the output layer. In the case of an anomalous instance, the trained model gives a high error rate, failing to regenerate it properly.

Deep learning methods outperform traditional machine learning approaches since they can represent the input feature precisely with automatic extraction of the discriminatory features using multiple processing layers. Our proposed approach uses a contractive autoencoder, which can estimate a good representation of the input feature space by extracting the essential features. A contractive autoencoder utilizes an optimum loss function by optimizing the penalty term to achieve the invariant representation of the input data. The lower-dimensional latent representation forces the model to learn only the essential features of the input. It can reconstruct the data in the output layer very well by extracting the significant features from the normal sample. The proper reconstruction of normal traffic makes our model efficient in separating the attack data from the non-attack data since our detection method is based on the fact that the anomalous sample will deviate from the normal one by generating higher reconstruction loss.

Our model uses Deep Contractive Autoencoder (DCAE), containing two encoder and decoder layers, to learn the representations of the network sample in a semi-supervised manner. Figure 4 presents our methodology. The input layer of our DCAE model takes the input from the training dataset containing 121 features of the NSL-KDD datasets and 66 features

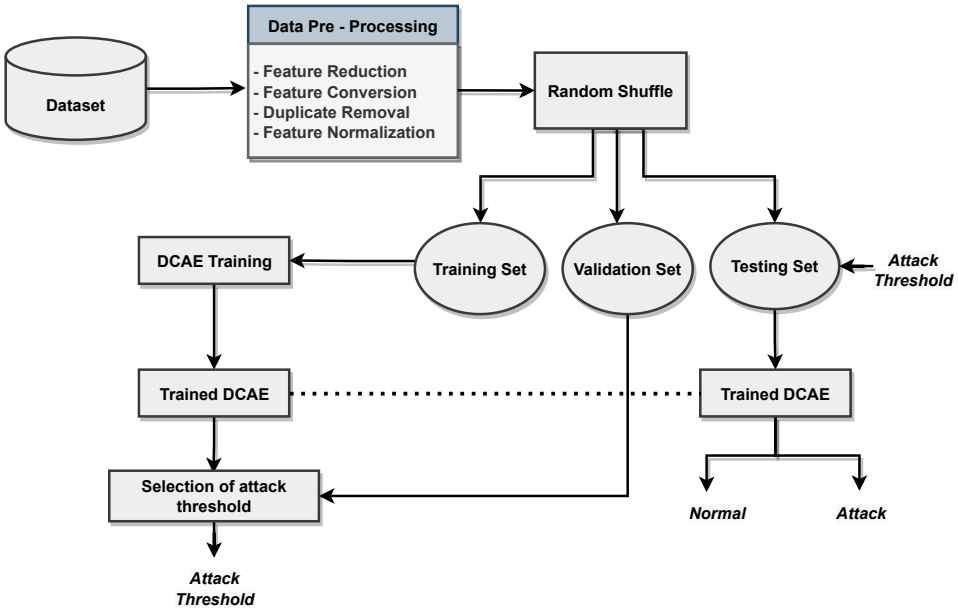


Figure 4: The methodology for our model

for the CIC-IDS datasets. The encoder block generates a fixed range feature vector h from the input data x. Then it decreases the initial feature vector's dimension sequentially. In our experiments, the first and second encoder layers reduce the dimensions to 32 and 16 for the CIC-IDS dataset and 60 and 30 for the NSL-KDD dataset, respectively.

After the encoding stage, the decoder block generates the output feature vector, x', from the encoded data. The layers in the decoder block are placed in the opposite sequence of the encoder layers. The dimensions are incremented to 16 and 32 after the first and second decoder layers for the CIC-IDS datasets, whereas they are increased to 30 and 60 for the NSL-KDD dataset for the subsequent decoder layers. The last layer of the decoder block goes through a fully connected layer to produce the output feature vector, x'. Several activation functions can be utilized in the hidden layers, such as linear, softmax, sigmoid, tanh, and rectified linear units. Our model utilizes a non-linear activation function, *sigmoid*, in the hidden layers since it can capture more valuable features from the input data. Figure 5 shows the structure of our proposed model for the NSL-KDD dataset. The design will be changed in the CIC-IDS dataset since the number of units in the layer will vary.

We aim to reconstruct an analogous output feature vector x' to the input feature vector x. Since a contractive autoencoder aims to learn useful information from the input data, we have used this framework for our attack detection model. We have employed the contractive loss as a reconstruction error between input data x and output representation x'. An additional penalty term is used with the classical reconstruction loss function of autoencoders in this loss function. This penalty is the Frobenius norm of the Jacobian matrix of the encoder activations concerning the input, stated in equation 5. This penalized term results in a localized space contraction which extracts robust features on the activation layer. The details of this autoencoder and the

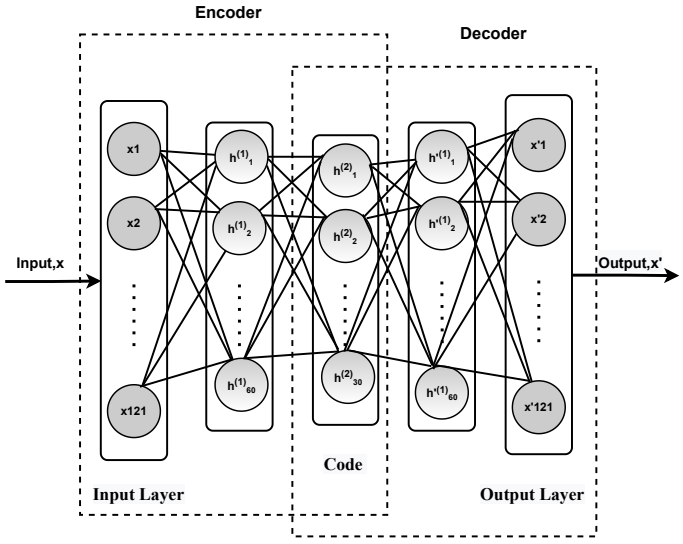


Figure 5: The structure of our proposed model (NSL-KDD dataset). For the CIC-IDS datasets, neuron size will differ.

loss function calculation can be found in section 4.2. Moreover, Algorithm 1 presents the contractive loss calculation. We used Kristiadi's implementation for Keras [38]. The first line shows the calculation of the mean squared error between the true and predicted one, which is added to the penalty term of the contractive autoencoder. The penalty is measured as the squared Frobenius norm of the Jacobian matrix of partial derivatives associated with the encoder function. Line 3-7 demonstrates the necessary calculation for acquiring the penalty term. Finally, our model's contractive loss is measured as shown in line 9.

Our proposed methodology for attack detection is based on the observation that an autoencoder is trained using only normal data, it will provide a good reconstruction by generating a low

Algorithm 1 Calculation of Contractive Loss [38]

```
1: function Contractive Loss(y_pred, y_true)
2:     mse = K.mean(K.square(y_true − y_pred), axis = 1)
3:     W = K.variable(value=model.get_layer('encoded_2')
4: .get_weights()[0])
5:     W = K.transpose(W)
6:     h = Model.get_layer('encoded_2').output
7:     dh = h ∗ (1 − h)
8:     lam = .01
9:     contractive = mse + lam ∗ K.sum(dh^2 ∗
   K.sum(W^2, axis = 1), axis = 1)
10:    return contractive
11: end function
```

Algorithm 2 Threshold Measurement Algorithm

Input: Training Data (X_Train), Validation Data(X_Val)
Output: Optimal Threshold

```
1: Train Model DCAE with X_train
2: Calculate Maximum and Minimum Reconstruction Error (RE)
   from X_Train :
3:     min_normal_re = np.min(train_loss_normal)
4:     max_normal_re = np.max(train_loss_normal)
5: Initialize Threshold with the Maximum RE:
       best_threshold = threshold = max_normal_re
6: while threshold > min_normal_re do
       Calculate the Accuracy of DCAE on X_Val by using
   threshold
7:     if accuracy > best_accuracy then
8:         best_accuracy = accuracy
9:         best_threshold = threshold
10:    end if
11:    threshold = threshold - 0.001
12: end while
```

Algorithm 3 Detection Method

Input: Model(*DCAE*), Threshold(*th*), Testing Data(X_Test)
Output: Labeled Sample

```
1: while x in X_Test do
2:     x' ← Model_Predict(x)
3:     δ ← RE(x, x')
4:     if δ ≤ th then
5:         label it as normal
6:     else
7:         label it as an attack
8:     end if
9: end while
```

reconstruction error while yielding a high reconstruction error for an anomalous one. Thus, successful detection of an attack scenario depends on the optimized reconstruction of the normal data. Anomalous data can be identified by selecting a proper threshold based on the reconstruction error (RE). This threshold can be called an *anomaly threshold*. A test sample will be labeled an anomaly when the trained autoencoder reconstructs it with a higher RE than the threshold. Otherwise, a normal label will be assigned. This threshold needs to be chosen wisely for determining an attack sample since it can cause substantial false detection by selecting weak thresholds.

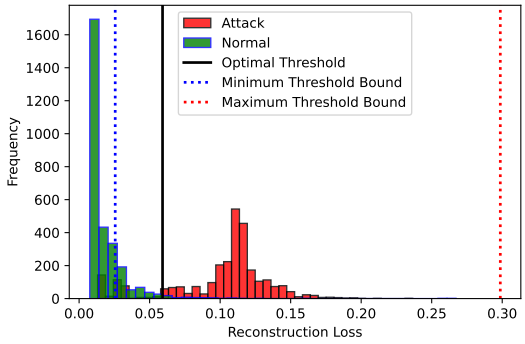
In this work, we have proposed an intrusion detection model utilizing a stochastic approach to select the anomaly threshold. We analyze the threshold value's significance for our model by looking at the dataset's reconstruction loss distribution. In the model, only normal samples have been used during training, while the validation and testing datasets comprise a combination of normal and anomalous samples. We have calculated the optimal threshold from the validation dataset instead of obtaining it directly for the test set and applied that threshold during testing. The detailed algorithm for threshold measurement has been demonstrated in Algorithm 2. We examine the range of (minimum and maximum reconstruction error) from the training set with 0.001 intervals to determine the best-performing threshold. In order to distinguish normal data from anomalous one, we start with the highest RE (stated as *max_normal_re* in line 4) of training data as the anomaly threshold, which has been stated in line 5. After that, we calculate the accuracy of our model using the validation dataset. If the obtained accuracy is better than the previous, we save it along with the threshold value. We run several iterations to find our optimal threshold giving the best accuracy until it reaches the limit value, which we have fixed to be the lowest reconstruction error of the training data, denoted as *min_normal_re* in line 3. Line 6-10 calculates the accuracy of our model with the obtained threshold in the validation data and stores the threshold with the best accuracy. After each iteration, the threshold value has been decreased by 0.001.

Figure 6 presents our threshold selection method. We have demonstrated the Reconstruction Loss Distribution Plot (normal and attack data) for the CIC-IDS2017(DoS) dataset in Figure 6a. Blue color denotes the normal data and red denotes the attack data. A bar plot depicts the histogram, and the solid black line indicates the optimal threshold value, which divides the normal and the attack classes. A blue and red dotted line denotes our model's minimum and maximum threshold bound, which we have observed from the training sample. The X-axis denotes reconstruction loss, and the Y-axis is the frequency. In Figure 6b, we have shown the Accuracy variation for different thresholds within the range. Instead of using any fixed threshold, we have iterated our experiment for different reconstruction loss value from the training data as the anomaly threshold. The optimal threshold for obtaining the best accuracy is then recorded for our model evaluation.

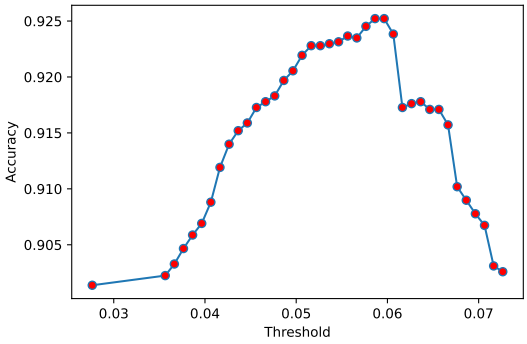
Algorithm 3 describes the attack detection method of our model. After the best threshold has been found from the validation set, our model predicts the class label using that threshold from the test dataset. First, a reconstruction is generated from the test data using the trained model as shown in line 2. After that, the reconstruction loss is calculated between the original test data and its reconstruction. Next, the loss is compared with the threshold generated from the algorithm 2. In lines 3-7, the algorithm compares the loss value with the threshold. In case the loss value is less than or equal to the threshold, it is labeled as normal; otherwise, it is classified as an attack sample.

5. Experiments

In this section, we first summarize the datasets utilized for our experiments. After that, we explain the evaluation metrics



(a) Reconstruction Loss Distribution Plot for the CIC-IDS2017(DoS) dataset



(b) Accuracy vs Different threshold

Figure 6: Threshold Selection Method

Table 1: CIC-IDS2017 Dataset

Day	Traffic
Monday	Benign
Tuesday	SSH & FTP Brute Force
Wednesday	DoS/DDoS & Heartbleed
Thursday	Web Attack & Infiltration
Friday	Botnet, Portscan & DDoS

Table 2: NSL-KDD Attack Categories

Category	Attacks
Probe	ipsweep, mscan, nmap, portsweep, saint, satan
DoS	apache2, back, land, mailbomb, neptune, pod, processtable, smurf, teardrop, udpstorm, worm
U2R	buffer_overflow, loadmodule, perl, ps, rootkit, sqlattack, xterm
R2L	ftp_write, guess_passwd, httptunnel, imap, multihop, named, phf, sendmail, snmpgetattack, snmpguess, spy, warezclient, warezmaster, xlock, xsnoop

of our proposed model's performance, followed by details about our experimental setup. Finally, we show the experimental results of our proposed model and compare them with the related works.

5.1. Datasets

Several intrusion detection evaluation datasets are available, consisting of benign and anomalous network traffic data. Since we focus our experiments on DoS/DDoS attack detection, we selected three datasets, NSL-KDD, CIC-IDS2017, and CIC-DDoS2019.

The first evaluation dataset for our experiment is the CIC-IDS2017 dataset [2], which was developed by the Canadian Institute for Cybersecurity (CIC). This dataset covers the most recent DoS/DDoS attacks and the realistic benign traffic. It contains a varied types of protocols along with attack variations. Hence, the CIC-IDS2017 dataset is suitable for our model evaluation. The CIC-IDS2017 dataset captured both normal and attack data for five days, from Monday, July 3, 2017, to Friday, July 7, 2017. Benign data was captured only on Monday, while the other days included attack data. The executed attacks contain Botnet, Brute Force FTP, Brute Force SSH, Heartbleed, Web Attack, Infiltration, DoS, and DDoS. Table 1 summarizes the traffic recorded per day. In this work, we have experimented with our model using only the DoS & DDoS attack dataset, which comes from the sample of the Wednesday in CIC-IDS-2017 dataset.

The second dataset used in our model evaluation is the NSL-KDD [3], which was published by the CIC to solve some inherent problems of the KDD Cup'99 dataset [5]. The classical KDD Cup data set [4] was established by the Defense Advanced Research Projects Agency (DARPA) and has been widely used as a benchmark for Intrusion Detection model evaluation [6]. However, the KDD Cup'99 has multiple problems [5], including class imbalance and redundant records. These drawbacks were resolved in the NSL-KDD dataset; therefore, the NSL-KDD data set has been widely used in several studies [6, 8] as a benchmark data set in the development of NIDSs for real-world applications. Hence, NSL-KDD fits our work's evaluation purpose and the comparison with relevant research. The NSL-KDD dataset covers DoS, probing, Remote to Local (R2L), User to Root (U2R), and benign classes. The details of this dataset have been summarized in Table 2. This dataset is extracted directly through TCP/IP connections and contains forty-one features, such as connection duration, protocol type, and accumulated traffic characteristics in each interval [6].

Our last evaluation dataset, which we have used for our model, is the recently released CIC-DDoS2019 [9], developed by the Canadian Institute for Cybersecurity (CIC). The dataset contains benign and contemporary DDoS attacks, which resemble real-world data. Several new attacks have been implemented using TCP/UDP-based protocols at the application layer, and a new taxonomy has been proposed in terms of reflection-based and exploitation-based attacks. For this dataset, the B-Profile system [10] has been utilized to build users' abstract behavior based on the HTTP, HTTPS, FTP, SSH, and email protocols. The dataset was collected on two separate days for training and testing evaluation. The train-

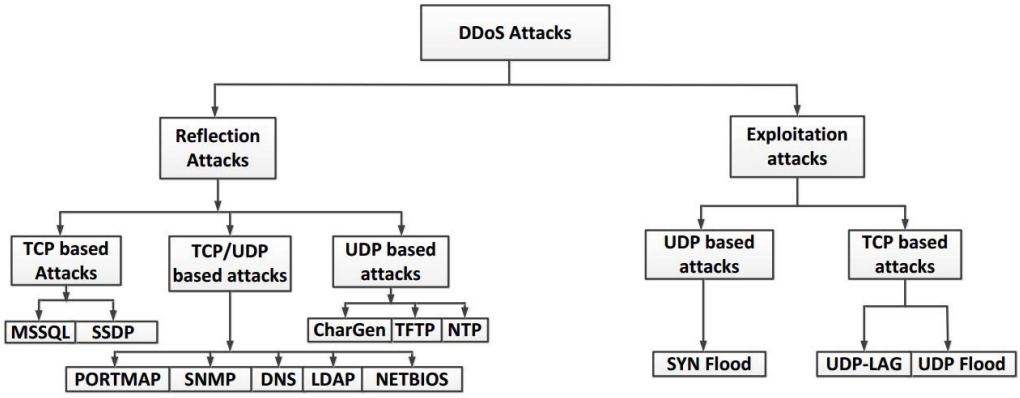


Figure 7: The attack distribution inside the CIC-DDoS2019 dataset [9]

ing set contains 12 DDoS attacks, including SNMP, NetBIOS, LDAP, TFTP, NTP, SYN, UDP, WebDDoS, MSSQL, UDP-Lag, DNS, and SSDP DDoS-based attacks. The testing data includes 7 DDoS attacks PortScan, SYN, MSSQL, UDP-Lag, LDAP, UDP, and NetBIOS. Figure 7 shows the distribution of the different attacks in the dataset. The researchers extracted more than 80 flow features in the CIC-DDoS2019 dataset using CICFlowMeter tools [26]. The dataset is publicly available in both PCAP file and flow format on the Canadian Institute for Cybersecurity website.

5.2. Dataset Pre-processing

This paper emphasizes a binary classification problem for anomaly detection wherein each observation is categorized as a normal or attack class. Before training the IDS model, we enacted the following pre-processing steps on our selected datasets:

- The datasets from CIC-IDS2017 and CIC-DDoS2019 contain different socket information, namely Source/Destination IP, Source/Destination Port, and flow ID. We removed socket-involved features from the data samples to eliminate the overfitting problem since such data can vary from network to network. The final dataset contains 77 & 78 various features, besides the traffic label of the CIC-IDS2017 and CIC-DDoS2019, respectively.

- We used one-hot encoding to convert the categorical features, such as protocol type, services, and flag, of the NSL-KDD dataset into numerical features. For example, TCP, UDP and ICMP protocols have been mapped to (1,0,0), (0,1,0) and (0,0,1), respectively. Similarly, the 'flag' feature containing 11 values and the 'services' feature with 70 values have been mapped to numerical features. Thus, 41 original features are finally transformed into 121 numeric features.

- The non-numerical class labels are also converted into numeric categories using binary encoding. Since we have considered only binary classification in this model to identify the anomalous and normal traffic from input data, those instances are assigned to 1 and 0, respectively.

- Duplicity in a dataset may lead to a bias towards more frequent records while training the anomaly detection model. Hence, we removed all the duplicate records from the data and kept only one copy of each record to resolve this issue. After the operation, the number of samples is reduced to 587, 966 from the CIC-IDS2017 (DoS) dataset. We also removed those samples containing the NaN and INF feature values from the dataset.

- The numeric features have been normalized to remove the effect of the original feature value scales. We have used the Min-Max Normalization for each feature, which re-scales the range of features to scale the range in [0, 1]. The below equation represents the formula for Min-Max Normalization:

$$z_i = \frac{x_i - \min(x)}{max(x) - \min(x)} \tag{7}$$

  where x_i is a d-dimensional feature vector from the training dataset and z_i is the *ith* normalized data.

- In case all the values in the columns are the same, it does not affect the learning but increases the data dimension. Therefore, those constant valued features have been removed from our dataset. For example, the column '**num_outbound_cmds**' in NSL-KDD consists of only zero values; hence, it has been removed from the samples. Moreover, table 3 shows the list for the CIC-IDS2017 DoS dataset, which contains ten features containing zero values.

Table 3: List of Constant Valued Features in CIC-IDS2017 (DoS) Dataset

Bwd PSH Flags	Fwd URG Flags
Bwd URG Flags	CWE Flag Count
Fwd Avg Bytes/Bulk	Fwd Avg Packets/Bulk
Fwd Avg Bulk Rate	Bwd Avg Bytes/Bulk
Bwd Avg Packets/Bulk	Bwd Avg Bulk Rate

5.3. Evaluation Metrics

Four performance metrics have been used for evaluating our proposed model: Precision, Recall, F1-Score, and Accuracy. These metrics are calculated by using four different measures, true positive (TP), true negative (TN), false positive (FP), and false negative (FN):

- *TP*: If an attack instance is correctly labeled, it is measured as *TP*.
- *FP*: If a normal instance is labeled as an attack, it is measured as *FP*.
- *TN*: If a normal instance is labeled as normal, it is measured as *TN*.
- *FN*: If an attack instance is labeled as normal, it is measured as *FN*.

Precision: Ratio of the number of correctly classified attack samples to the total number of instances which are classified as an attack.

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

Recall: Ratio of the number of correctly classified anomalous instances to the number of all actual anomalous instances.

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

Accuracy: Ratio of the number of correctly classified anomalous and normal instances to the number of all instances.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{10}$$

F1-score: Harmonic average of the precision and recall metrics to express the performance of the model.

$$F1\text{-}score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{11}$$

We have also utilized Confusion Matrix, a specific table layout that allows visualization of the performance of an algorithm in our work. Table 4 shows the definition of confusion matrix.

Table 4: Confusion Matrix

		Predicted Class	
		Anomaly	**Normal**
Actual Class	**Anomaly**	TP	FN
	Normal	FP	TN

Table 5: Data Distribution Of Training, Validation and Test Sets

	Label	Training Set	Validation Set	Test Set
CIC-IDS2017 (DoS)	Normal	291777	9157	9157
	Attack	-	9157	9157
NSL-KDD	Normal	47215	5668	5668
	Attack	-	5668	5668

Table 6: Hyperparameter for our Model

Parameter	**Value**
Epoch & Batch Size	100 / 32
Activation Function	Sigmoid(hidden) / Linear(Output)
Optimizer	Adam
Loss Function	Contractive
No of Hidden Layers	2
Hidden Neuron Size	60/30, 32/16

5.4. Experimental Setup

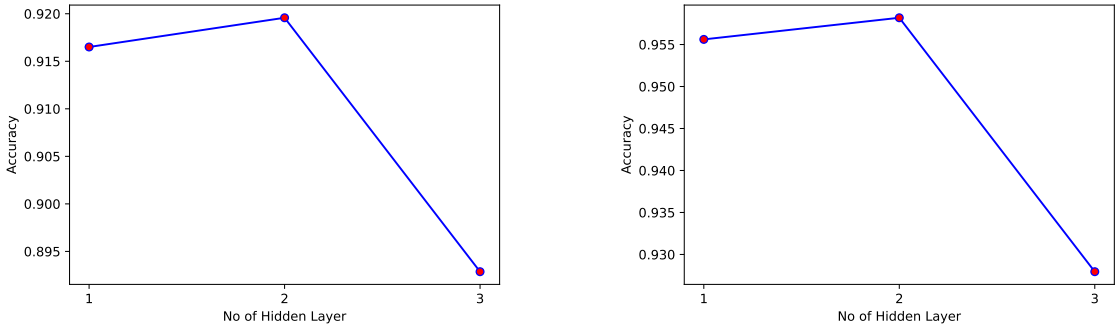
We have used Keras as a deep learning framework in our model. The experiment is performed on Jupyter Notebook using 11th Gen Intel(R) Core(TM) i7-1185G7 @ 3.00GHz 1.80 GHz with 16GB RAM.

After the dataset pre-processing, normal and attack samples in the dataset were shuffled separately. We partitioned each dataset into three sections training, validation, and testing subset. The training dataset contains only normal samples, whereas test and validation datasets individually have the same number of normal and anomalous samples. Our model is built using the training set while the validation set is used to fine-tune the model's hyper-parameters e.g., the number of hidden layers in the proposed model. Besides, the test set is used to evaluate the model performance. Table 5 displays the data partitioning of our model.

In our experiment, the linear layer takes the decoder output and reconstructs the input data. We used Contractive Loss as a cost function with Adam Optimizer and Sigmoid function for activation in hidden layers. We trained the model using 100 epochs and a batch size of 32. Table 6 lists all the hyperparameters of our model.

We executed several experiments using different values of hyperparameters to get optimal results. Selection of the best values of the hyper-parameters is crucial for creating a successful neural network architecture, as the trained model behavior depends on these values. We evaluated the model's performance using different number of hidden layers, number of neurons per hidden layer, batch sizes and the activation functions. The best performance is achieved when we use two hidden layers, batch size of 32 , sigmoid activation function in the hidden layer.

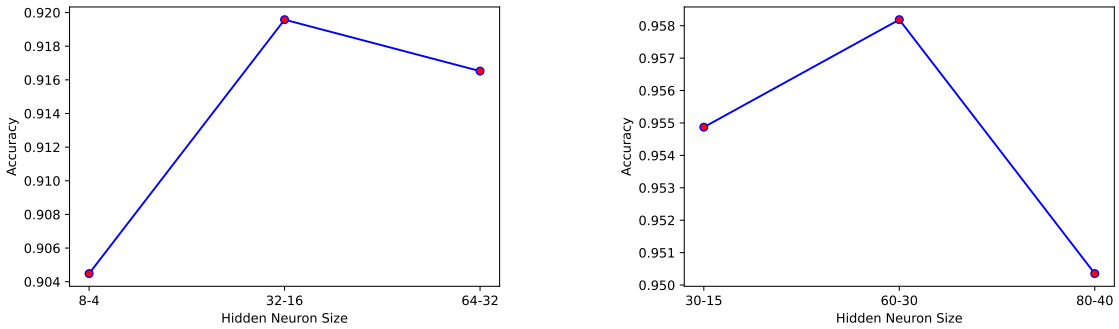
It is essential to carefully analyze the number of hidden layers and the number of neurons in each of these layers. With the increase in hidden layer numbers in the autoencoder, the quality of latent representation also increases. However, it may also lead to poor performance if not properly tuned. To find the best-performing one, we have tested our model's performance using 1, 2, and 3 hidden layers. Figure 8 illustrates the experimental result for both CIC-IDS2017 and NSL-KDD datasets.



(a) CIC-IDS2017 Dataset

(b) NSL-KDD Dataset

Figure 8: The result of Accuracy change as hidden layer changes



(a) CIC-IDS2017 Dataset

(b) NSL-KDD Dataset

Figure 9: The result of Accuracy change as hidden neuron size changes

The model architecture with two hidden layers slightly outperforms the model with one hidden layer, which tends to decrease significantly when three hidden layers are used.

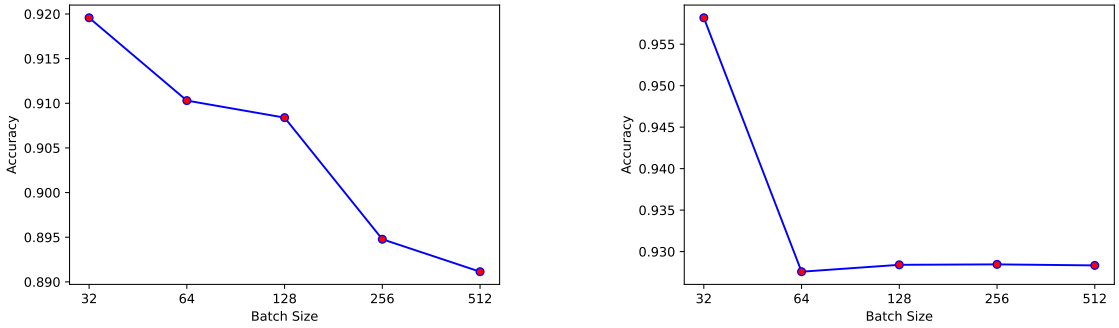
Typically, the hidden layer of an autoencoder contains fewer neurons than the input and output layers, creating a bottleneck and forcing the network to learn a higher-level representation of the input. Using too few or too many neurons in the hidden layers may result in performance degradation. We have experimented with our model's performance containing two hidden layers, with different combinations of neuron sizes for each hidden layer. Each hidden layer in the encoder is set to be half the size of the layer before it. For the NSL-KDD dataset, we have experimented with "30-15", "60-30", and "80-40," where the first one denotes the first hidden layer's neuron size and the second one denotes the latent/code layer's neuron size. For the CIC-IDS2017 data, we have tested with "8-4", "32-16", and "64-32". We discovered that changing the neuron size in the latent layer affects the performance of our model. Figure 9a presents the results of our experiments when the CIC-IDS2017 data are employed. This figure shows that model performance decreases using only four neurons ("8-4") in the latent layer. Again, it also gives less accuracy while using 32 neurons ("64-32"). The best result is observed when we choose "32-16" with the code layer's size containing 16 neurons. Similarly, in Figure 9b, we can see that using 30 neurons in our model's latent layer outperforms the others with 15 and 40. Hence, we have selected the hidden neuron sizes showing better accuracy results.

Batch size, which determines how many samples must be processed before the internal model parameters are updated, is another crucial hyperparameter in deep learning model training. We assessed our model's performance with different batch sizes of 32, 64, 128, 256, and 512 on the CIC-IDS2017 and NSL-KDD datasets, as shown in Figure 10. According to the graphs, our model's performance tends to decline as the batch size grows. This observation is because learners' ability to generalize declines as the number of batches increases [40]. The term "generalization" describes a model's capacity to respond to and perform when presented with unknown data. Hence, we have chosen 32 as our model's batch size.

We experimented with several non-linear activation functions, such as tanh, sigmoid, and ReLU (Rectified Linear Unit), to find the best-performing activation function in the hidden layer. Both the NSL-KDD and CIC-IDS2017 datasets show the best performance results with the sigmoid function, as shown in Figure 11.

5.5. Experimental Results

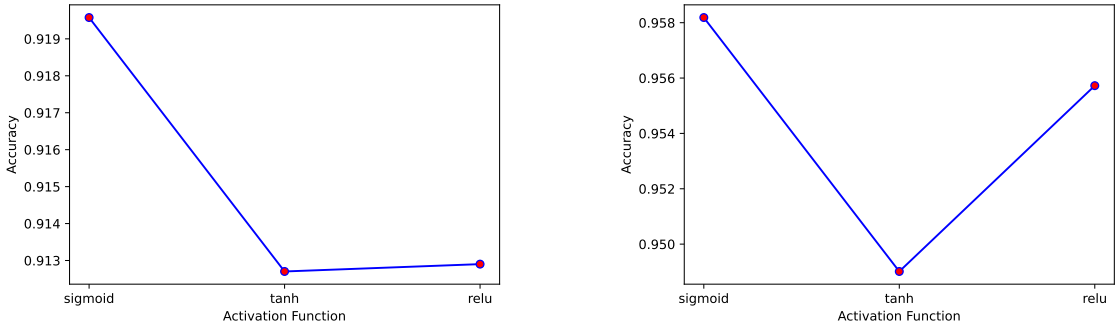
We compared our model with some of the most well-known deep learning models, Basic AE, Variational AE, and LSTM AE, to evaluate our proposed model for each of chosen datasets.



(a) CIC-IDS2017 Dataset

(b) NSL-KDD Dataset

Figure 10: The result of Accuracy change as batch size changes



(a) CIC-IDS2017 Dataset

(b) NSL-KDD Dataset

Figure 11: The result of Accuracy change as activation function changes

Table 7: The Evaluation Metric Comparison for CIC-IDS2017 dataset. We present the precision, recall, F-score and accuracy for the different deep learning algorithms

Algorithm	Precision	Recall	F1-Score	Accuracy(%)
LSTM AE	0.8977	0.8869	0.8862	88.69
VAE	0.9015	0.8873	0.8863	88.73
Our Approach	**0.9246**	**0.9245**	**0.9245**	**92.45**
Basic AE	0.8990	0.8984	0.8984	89.84

Table 8: The Evaluation Metric Comparison for NSL-KDD dataset. We present the precision, recall, F-score and accuracy for the different deep learning algorithms

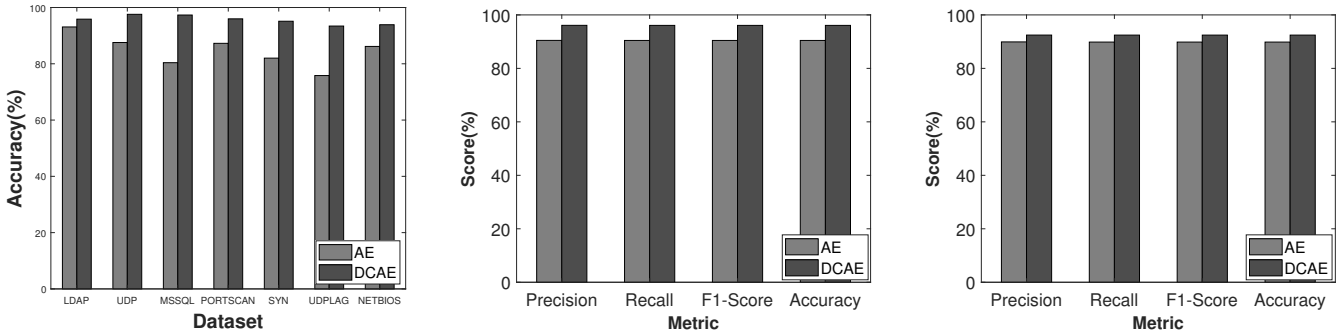
Algorithm	Precision	Recall	F1-Score	Accuracy(%)
LSTM AE	0.9322	0.9320	0.9320	93.20
VAE	0.9369	0.9368	0.9368	93.68
Our Approach	**0.9610**	**0.9608**	**0.9608**	**96.08**
Basic AE	0.9047	0.9045	0.9045	90.45

Table 9: The Evaluation Metric Comparison for CIC-DDoS2019 dataset. We report the accuracy of the different deep learning algorithms

Algorithm	LDAP	UDP	MSSQL	PORTMAP	SYN	UDPLAG	NETBIOS
LSTM AE	94.10	87.98	95.33	88.60	95.40	70.46	78.32
VAE	90.20	70.46	84.48	77.15	78.35	70.50	79.50
Our Approach	**95.86**	**97.58**	**97.33**	**95.97**	**95.12**	**93.41**	**93.88**
Basic AE	93.09	87.56	80.39	87.27	82.01	75.82	86.17

We analyzed all methods' precision, recall, F-score, and accuracy values. Our model obtains an accuracy of 92.45% and 96.08% for the CIC-IDS2017 (DoS) and NSL-KDD datasets, respectively. Table 7 and 8 represents the evaluation results of our model for the CIC-IDS2017 (DoS) and NSL-KDD datasets respectively along with other techniques. For the CIC-DDoS2019 dataset, our accuracy for each attack classes is ranging from 93.41% - 97.58% shown in Table 9. From the obtained results, it is visible that our approach has the best performance metrics in comparison to the other methods. Specifically, the results show that the proposed method outperforms other methods significantly in the CIC-DDoS2019 dataset, where our accuracy range is [93.41% – 97.58%] whereas the accuracy range for LSTM AE [70.46%–95.40%], VAE [70.46%–90.20%], and Basic AE [75.82% – 93.09%]. Additionaly, we illustrated the comparative results between basic AE and our model (DCAE), shown in Figures 12a, 12b, 12c.

Figure 13 presents our method's confusion matrix for NSL-KDD and CIC-IDS2017 datasets. Figure 13a shows that our method accurately detected 5523 true negatives and 5369 true positives out of 11336 instances in NSL-KDD datasets. The method could not correctly detect 444 cases, where 145 of them were false positives and 299 false negatives. Figure 13b shows that our method accurately detected 8428 true negatives and 8504 true positives out of 18314 instances in CIC-IDS2017 datasets. The method could not correctly detect 1382 cases,

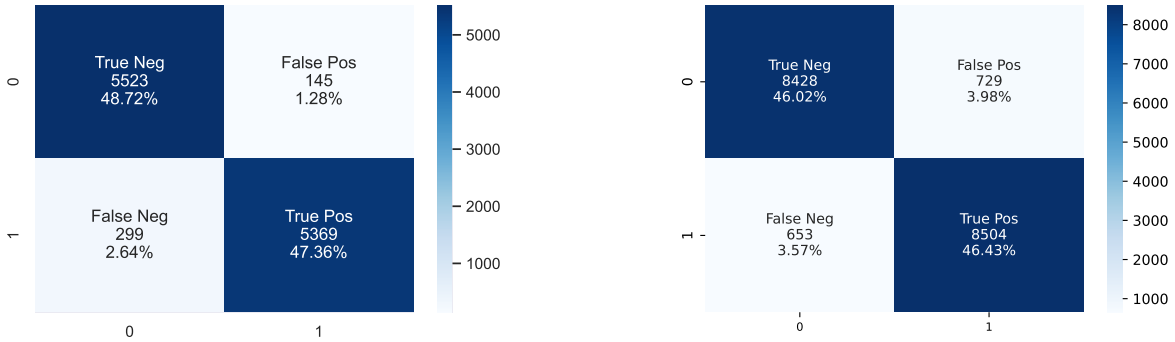


(a) AE and DCAE performance comparison on different attack dataset of CIC-DDoS2019 in terms of accuracy

(b) AE and DCAE performance comparison on NSL-KDD dataset

(c) AE and DCAE performance comparison on CIC-IDS2017 dataset

Figure 12: AE and DCAE performance comparison on different attack datasets



(a) Confusion Matrix of our proposed approach for NSL-KDD

(b) Confusion Matrix of our proposed approach for CIC-IDS2017

Figure 13: Confusion Matrix of the evaluation dataset of our proposed approach

where 729 of them were false positives and 653 false negatives.

We have also used the Receiver Operating Characteristic (ROC) curve to show the efficacy of our model. The ROC curve shows the relationship between two parameters: true and false classes. The area under the ROC Curve (AUC) measures how well a model can distinguish between classes. Figure 14a shows that our model gives an AUC of 96.08 for the NSL-KDD dataset, which means that our proposed model can separate 96.08% of positive and negative classes successfully. It gives an AUC of 92.45 for the CIC-IDS2017(DoS) dataset, depicted in Figure 14b.

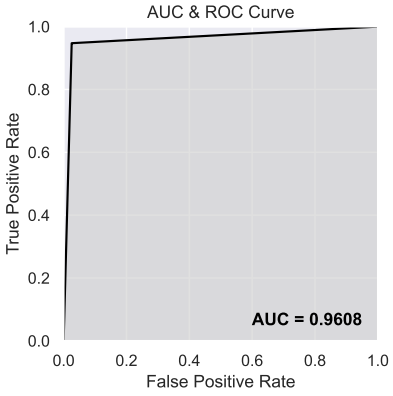
6. Discussion and Future Work

Autoencoder-based techniques have proven to be valuable tools for detecting Distributed Denial-of-Service (DDoS) attacks in Intrusion Detection Systems. AE-based techniques can identify deviations from normal behavior that may indicate a DDoS attack by learning a low-dimensional representation of normal network behavior. Specifically, utilizing semi-supervised learning helps us to detect new types of attacks that were not present in the training data. Considering how alive are the network attacks and easy to introduce new type of attacks, addressing zero-day DDoS attacks are crucial for IDSs.

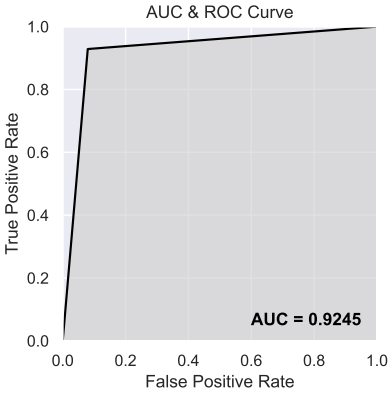
Besides the many advantages of AE-based techniques, it is essential to understand their limitations. AE-based techniques may also produce false alarms, particularly when the attack is similar to normal network traffic. In the case of using nonoptimal hyperparameters of the AE-based techniques, the accuracy can significantly drop. Therefore, fine-tuning the hyperparameters of the AE is crucial for reducing false alarms. We conducted several trials with different hyperparameter values in our experiments to choose the optimal one. Our results showed that our model's performance was significantly improved by using the optimal hyperparameters. This highlights the importance of carefully choosing hyperparameters in deep learning models.

The significance of choosing the optimal anomaly threshold has been analyzed in this work by examining the reconstruction loss distribution of normal and anomalous data. We performed several iterations using the validation dataset to select the optimal threshold and observed the model's accuracy with different threshold values. This highlights the importance of using a dynamic threshold selection method, as the fixed threshold might not be effective for different datasets.

We plan to apply our current model to other benchmark datasets in our future work. Moreover, we intend to expand the binary classification problem into a multi-class classification problem.



(a) Receiver Operating Curve (ROC) of our proposed approach for NSL-KDD



(b) Receiver Operating Curve (ROC) of our proposed approach for CIC-IDS2017

Figure 14: Receiver Operating Curve (ROC) of our proposed approach

7. Conclusions

Traditional Intrusion Detection Systems fail to detect sophisticated attacks with unexpected patterns; hence detection of these attacks has become one of the most challenging problems on the Internet. In this paper, we presented a new Deep Learning based Intrusion Detection model, explicitly focusing on the Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks. Our proposed Deep Learning framework is based on a contractive-autoencoder that can efficiently model the normal traffic data. It detects the anomaly from the dataset using a stochastic threshold strategy based on the reconstruction error. The performance of the proposed method is evaluated by using three benchmark datasets, NSL-KDD, CIC-IDS2017, and CIC-DDoS2019. Our results show that the proposed method can achieve up to 97.58% accuracy in detecting anomalies.

References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors", Nature 323, pp. 533-536, 1986

[2] Intrusion Detection Evaluation Dataset (CIC-IDS2017), `https://www.unb.ca/cic/datasets/ids-2017.html`

[3] NSL-KDD dataset, `https://www.unb.ca/cic/datasets/nsl.html`

[4] KDD Cup 1999 Data, `https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`

[5] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set", IEEE Symposium on Computational Intelligence for Security and Defense Applications, IEEE, 2009

[6] H. Choim, M. Kim, G. Lee, and W. Kim, "Unsupervised learning approach for network intrusion detection system using autoencoders", The Journal of Supercomputing, vol 75, no 9, pp. 5597-5621, 2019

[7] A. Y. Nur and M. E. Tozal, "Record Route IP Traceback: Combating DoS Attacks and the Variants", Computers & Security vol 72, pp 13-25, 2018

[8] H. Hindy, R. Atkinson, C. Tachtatzis, J. Colin, E. Bayne, and X. Bellekens, "Utilising deep learning techniques for effective zero-day attack detection", Electronics vol 9 no 10, 2020

[9] DDoS Evaluation Dataset (CIC-DDoS2019), `https://www.unb.ca/cic/datasets/ddos-2019.html`

[10] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy", International Carnahan Conference on Security Technology (ICCST), IEEE, 2019

[11] A. Y. Nur, "Combating DDoS Attacks with Fair Rate Throttling", IEEE International Systems Conference (SYSCON), 2021

[12] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system", International Conference on Advanced Communication Technology, IEEE, 2018

[13] BBC News, "Amazon 'thwarts largest ever DDoS cyber-attack'", Jun 18, 2020, retrieved Sep 27, 2022, `https://www.bbc.com/news/technology-53093611`

[14] A. Y. Nur and M. E. Tozal. "Single Packet AS Traceback against DoS Attacks", IEEE SYSCON, 2021

[15] M. S. Elsayed, N. Le-Khac, S. Dev, A. D. Jurcut, "Network anomaly detection using LSTM based autoencoder", ACM Symposium on QoS and Security for Wireless and Mobile Networks, 2020.

[16] Y. Ding and Y. Zhai, "Intrusion detection system for NSL-KDD dataset using convolutional neural networks", International Conference on Computer Science and Artificial Intelligence, 2018

[17] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks", IEEE Access, vol 5, pp. 21954-21961, 2017

[18] K. Yang, J. Zhang, Y. Xu, and J. Chao, "DDoS attacks detection with autoencoder", IEEE/IFIP Network Operations and Management Symposium, IEEE, 2020

[19] S. T. Zargar, J. Joshi, and D. Tipper, "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks." IEEE Communications Surveys & Tutorials, 2013

[20] G. Carl, G. Kesidis, R. R. Brooks, and R. Suresh, "Denial-of-service attack-detection techniques," IEEE Internet Computing, vol. 10, pp. 82-89, 2006

[21] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of Network-Based Defense Mechanisms Countering the DoS and DDoS Problems", ACM Computing Surveys, 2007

[22] M. S. Elsayed, N. Le-Khac, S. Dev, A. D. Jurcut, "DDoSNet: A deep-learning model for detecting network attacks", International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), IEEE, 2020

[23] J. P. Anderson, "Computer security threat monitoring and surveillance", Technical Report, James P. Anderson Company, 1980

[24] M. S. Elsayed, N. Le-Khac, S. Dev, A. D. Jurcut, "Detecting abnormal traffic in large-scale networks", International Symposium on Networks, Computers and Communications (ISNCC), IEEE, 2020

[25] B. Min, J. Yoo, S. Kim, D. Shin, and D. Shin, "Network anomaly detection using memory-augmented deep autoencoder", IEEE Access, vol 9, pp. 104695-104706, 2021

[26] CICFlowMeter project, `https://github.com/ISCX/CICFlowMeter`

[27] X. Yuan, C. Li, and X. Li, "DeepDefense: Identifying DDoS Attack via Deep Learning", IEEE International Conference on Smart Computing (SMARTCOMP), IEEE, 2017

[28] R. C. Aygun and A. G. Yavuz, "Network anomaly detection with stochastically improved autoencoder based models", International Conference on Cyber Security and Cloud Computing, IEEE, 2017

[29] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE", Journal of machine learning research, vol 9, no 11, 2008

[30] M. S. Elsayed, N. Le-Khac, S. Dev, A. D. Jurcut, "Machine-learning techniques for detecting attacks in SDN", International Conference on Computer Science and Network Technology, IEEE, 2019
[31] W. Wang, X. Du, D. Shan, R. Qin, and N. Wang, "Cloud intrusion detection method based on stacked contractive auto-encoder and support vector machine." IEEE transactions on cloud computing (2020).
[32] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, "CNN-based network intrusion detection against denial-of-service attacks." Electronics 9.6 (2020): 916.
[33] S. Aktar and A. Y. Nur, "Hash Based AS Traceback against DoS Attack", International Conference on Advanced Communication Technologies and Networking (CommNet), IEEE, 2021
[34] Google Cloud, How Google Cloud blocked the largest Layer 7 DDoS attack at 46 million rps, retrieved September 27 2022, https://cloud.google.com/blog/products/identity-security/how-google-cloud-blocked-largest-layer-7-ddos-attack-at-46-million-rps
[35] Kaspersky Lab, DDoS attacks hit a record high in Q4 2021, retrieved September 27 2022, https://www.kaspersky.com/about/press-releases/2022_ddos-attacks-hit-a-record-high-in-q4-2021
[36] M. T. Gil and M. Poletto, "MULTOPS: A Data-Structure for Bandwidth Attack Detection", USENIX Security Symposium, 2001
[37] A. Y. Nur, "Efficient Probabilistic Packet Marking for AS Traceback", IEEE International Symposium on Networks, Computers and Communications (ISNCC), 2021
[38] Deriving Contractive Autoencoder and Implementing it in Keras - https://agustinus.kristia.de/techblog/2016/12/05/contractive-autoencoder/
[39] Kaspersky Lab - Retrieved 10/09/2022 - https://usa.kaspersky.com/about/pressreleases/2018ddos-breach-costs-rise-to-over-2m-for-enterprises-findskaspersky-lab-report
[40] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. arXiv preprint arXiv:1609.04836
[41] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey", arXiv preprint arXiv:1901.03407, 2019
[42] M. F. Balın, A. Abid, and J. Zou, "Concrete Autoencoders: Differentiable Feature Selection and Reconstruction", International Conference on Machine Learning, 2019
[43] H. Arami, A. A. Joshi, J. Li, S. Aydore, and R. M. Leahy, "A Robust Variational Autoencoder Using Beta Divergence", Knowledge-Based Systems, 238 (2022): 107886
[44] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Learning semantic segmentation of large-scale point clouds with random sampling", IEEE Transactions on Pattern Analysis and Machine Intelligence 44.11 (2021): 8338-8354
[45] H. Akrami, A. A. Joshi, S. Aydore, and R. M. Leahy, "Addressing variance shrinkage in variational autoencoders using quantile regression", arXiv preprint arXiv:2010.09042 (2020)

Sharmin Aktar is a Ph.D. candidate in the Department of Computer Science at the University of New Orleans. She received her B.Sc. degree in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET) in 2016. Her research interests focus on Network Security and Machine Learning.

Abdullah Yasin Nur is an Assistant Professor in the Department of Computer Science at the University of New Orleans. He received his Ph.D. degree in Computer Science from the University of Louisiana at Lafayette in 2018. His research interests are Network Measurement and Analysis, Network Topology Discovering and Modeling, Network Security, and Graph Theory.