Network Intrusion Detection Using a Deep Learning Approach

Thesis · December 2022

DOI: 10.13140/RG.2.2.33853.23522

Sharmin Aktar
University of New Orleans
4 PUBLICATIONS **9** CITATIONS

Network Intrusion Detection Using a Deep Learning Approach

Sharmin Aktar
saktar@uno.edu

Network Intrusion Detection Using a Deep Learning Approach

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science

by

Sharmin Aktar

B.S. Bangladesh University of Engineering and Technology, 2016

December, 2022

ACKNOWLEDGMENTS

I want to acknowledge the support and guidance I received from my supervisor Dr. Abdullah Yasin Nur, Assistant Professor of Computer Science at the University of New Orleans (UNO) who made this work possible. His assistance carried me through all the stages of finishing this work. I would also like to thank my thesis committee: Dr. Vassil Roussev and Dr. Phani Vadrevu for their insightful comments and encouragement.

I would also like to give special thanks to my husband Mahfuzul Nissan, who is also a Ph.D. student in the Department of Computer Science, University of New Orleans, for his continuous support and inspiration in fulfilling my thesis work.

I am grateful to my parents and siblings for understanding when undertaking my research and writing my thesis. Your prayer for me was what sustained me this far.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

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 work 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 IDS 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

Chapter 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 Intrusion Detection Systems 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 work employs the principles of contractive autoencoder, an unsupervised deep learning technique. 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 work can be summarized as follows:

(a) We propose a deep learning method based on the contractive autoencoder model for attack detection. We have used only non-attack instances while training our model. The model reconstructs the input with a 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 utilize three benchmark datasets, NSL-KDD, CIC-IDS2017, and CIC-DDoS2019, to evaluate our model's performance in different environments.

(c) We analogize our model with the Basic Autoencoder and other deep learning models. The empirical results illustrate that our model surpasses those models.

Chapter 2

RELATED WORK

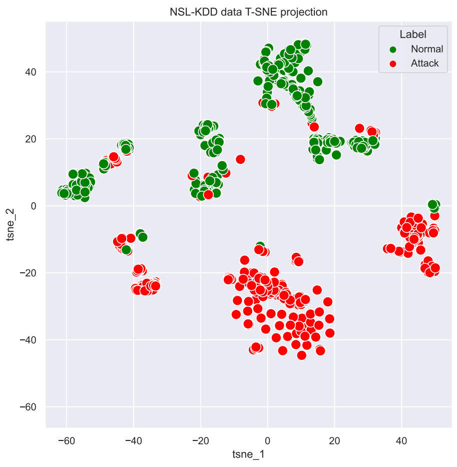
Many approaches have been suggested to defend systems against DDoS attacks [**7**; **19**; **20**; **21**]. DDoS defense mechanisms can be categorized as attack detection, attack reaction, and attack source identification. Attack detection techniques analyze the incoming packets and identify attacks 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 IP address [**14**; **33**; **37**]. This work 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.

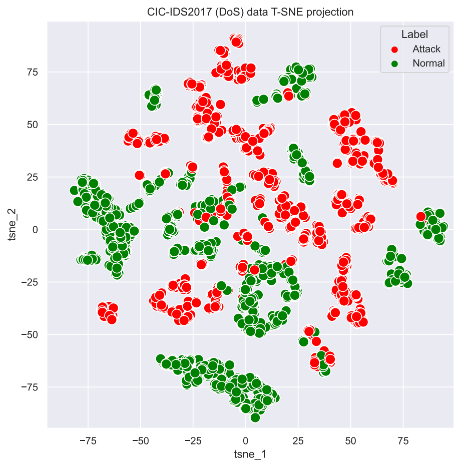
Elsayed et al. proposed a model using Long Short Term Memory (LSTM) autoencoder combining One-class Support Vector Machine (OC-SVM) algorithms to enhance performance [**15**]. They utilized only normal instances during training to learn the normal traffic behavior and to achieve the compressed representation 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. proposed a deep learning framework using the LSTM autoencoder for network attack identification [**24**]. 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 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.



(a) NSL-KDD



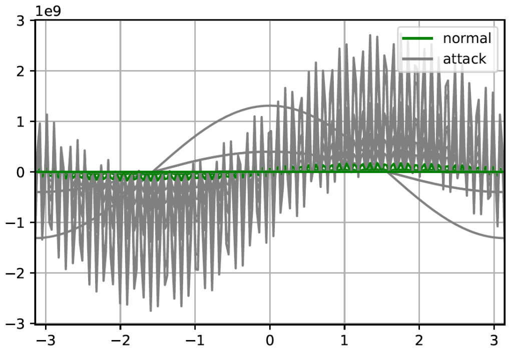
(b) CIC-IDS2017

Figure 2.1: t-SNE visualization

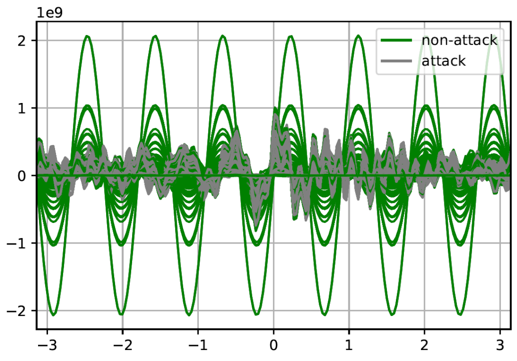
Farahnakia and Heikkonen suggested an IDS model (DAE-IDS) containing four autoencoders where each auto-encoders output in the current layer proceeds to the next one as an input [**12**]. Moreover, they followed a greedy layer-wise training manner which means an autoencoder is trained once the previous 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 [**26**] 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



(a) NSL-KDD dataset



(b) CIC-IDS2017(DoS) dataset

Figure 2.2: Andrews Curve Visualization

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.

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. Figs. 2.1a and 2.1b 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 2.2a and 2.2b 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.

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.

Chapter 3

METHODOLOGY

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

3.1 Autoencoder

The proposed attack detection model uses autoencoder attributes based on the unsupervised 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 3.1 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

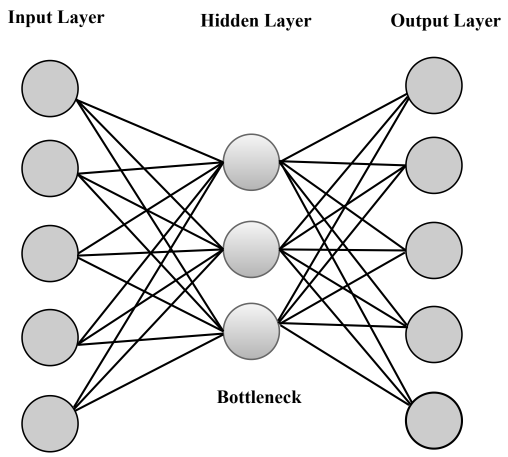


Figure 3.1: Structure of a Basic Autoencoder

the following form:

$$h = f(x) = \sigma(Wx + b), \tag{3.1}$$

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:

$$x' = g(h) = \sigma'(W'h + b'), \tag{3.2}$$

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:

$$\mathcal{J}_{AE}(\theta) = \sum_{x \in D_n} L(x, g(f(x))), \tag{3.3}$$

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:

$$L(x, x') = -\sum_{i=1}^{d_x} x_i \log(x'_i) + (1 - x_i) \log(1 - x'_i) \tag{3.4}$$

3.2 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:

$$\|J_f(x)\|_F^2 = \sum_{ij}(\frac{\partial h_j(x)}{\partial x_i}^2) \tag{3.5}$$

The objective function of Contractive AutoEncoder (CAE) obtained with the regularization term of equation 3.5 is as follows:

$$\mathcal{J}_{DCAE}(\theta) = \sum_{x \in D_n} L(x, g(f(x))) + \lambda \|J_f(x)\|_F^2 \tag{3.6}$$

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.

3.3 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. Deep learning methods outperform the 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.

Our model uses Deep Contractive Autoencoder (DCAE), containing two encoder and decoder layers, to learn the representations of the network sample in an unsupervised manner. Figure 3.2 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 for the CIC-IDS datasets. The encoder block generates a fixed range feature

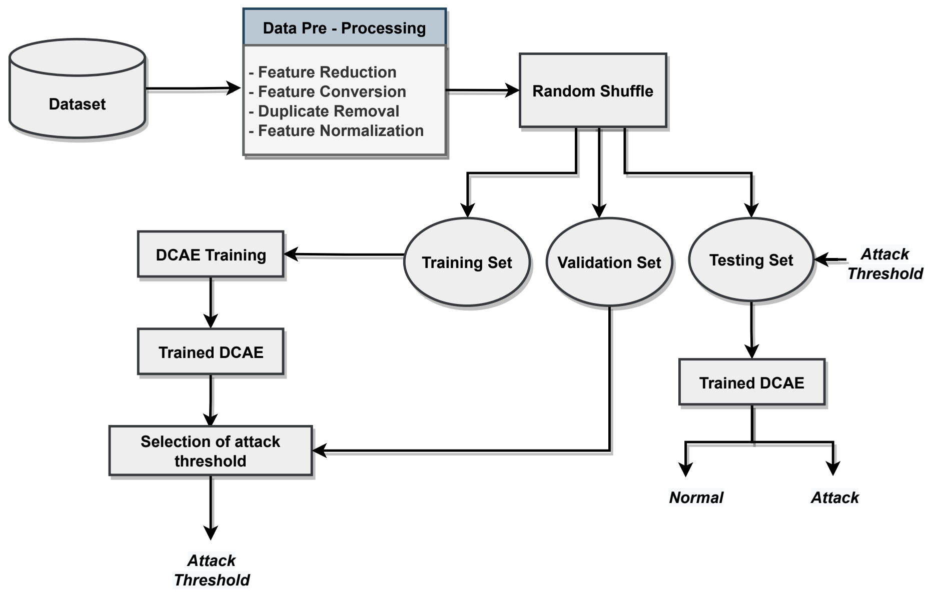


Figure 3.2: The methodology for our model

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. Fig 3.3 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. Fig 3.4 illustrates detailed design of our model for both NSL-KDD and CIC-IDS datasets.

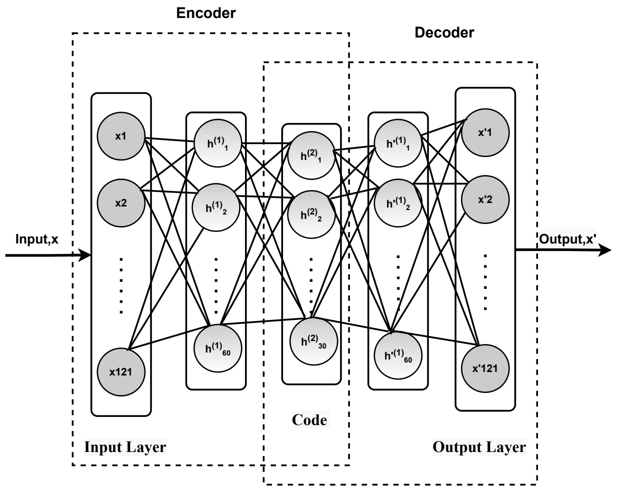
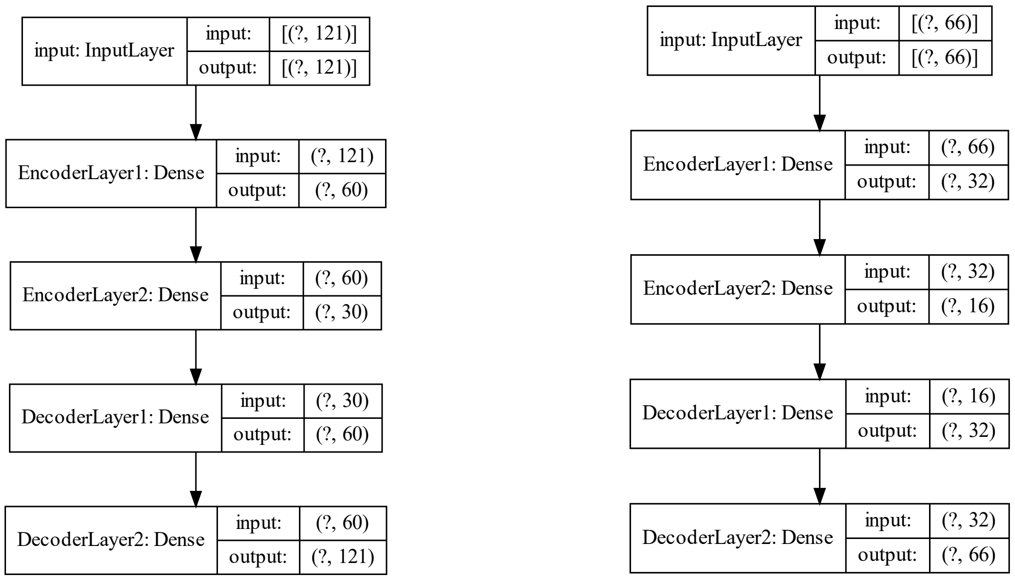


Figure 3.3: The structure of our proposed model (NSL-KDD dataset). For the CIC-IDS datasets, neuron size will differ.


input: InputLayer | input: [(?, 121)] | output: [(?, 121)]
EncoderLayer1: Dense | input: (?, 121) | output: (?, 60)
EncoderLayer2: Dense | input: (?, 60) | output: (?, 30)
DecoderLayer1: Dense | input: (?, 30) | output: (?, 60)
DecoderLayer2: Dense | input: (?, 60) | output: (?, 121)
input: InputLayer | input: [(?, 66)] | output: [(?, 66)]
EncoderLayer1: Dense | input: (?, 66) | output: (?, 32)
EncoderLayer2: Dense | input: (?, 32) | output: (?, 16)
DecoderLayer1: Dense | input: (?, 16) | output: (?, 32)
DecoderLayer2: Dense | input: (?, 32) | output: (?, 66)

(a) NSL-KDD (b) CIC-IDS2017

Figure 3.4: Model Design for NSL-KDD and CIC-IDS2017 dataset

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 3.5. This penalized term results in a localized space contraction which extracts robust features on the activation layer. The details of this autoencoder and the loss function calculation can be found in section 3.2.

Additionally, 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.

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
```

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 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

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
```

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. 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. In order to distinguish normal data from anomalous one, we start with the highest RE of training data as the anomaly threshold, which has been stated in line 5. We run many iterations to find our optimal threshold 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.

Algorithm 3 describes the attack detection method of our model. After the best

Algorithm 3 Detection Method

```
Input: Model($DCAE$), Threshold($th$), Testing Data(X_Test)
Output: Labeled Sample
1: while $x$ in $X\_Test$ do
2:     $x' \leftarrow Model\_Predict(x)$
3:     $\delta \leftarrow RE(x, x')$
4:     if $\delta \leq th$ then
5:         label it as normal
6:     else
7:         label it as an attack
8:     end if
9: end while
```

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 contractive RE 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.

Chapter 4

EXPERIMENTS

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

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 4.1 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**].

Table 4.1: CIC-IDS2017 Dataset

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

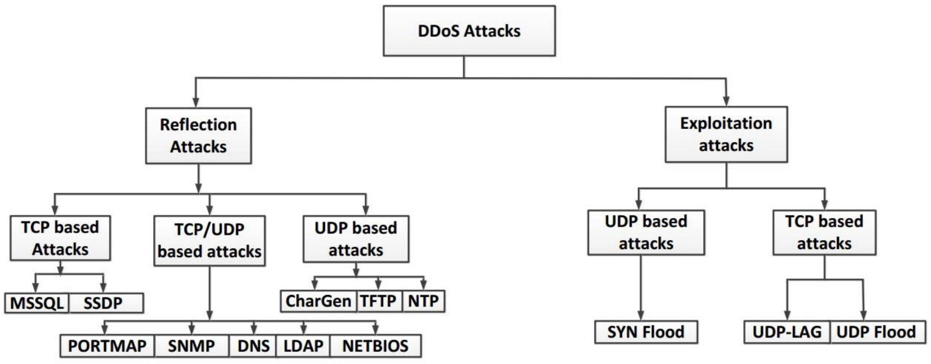


Figure 4.1: The attack distribution inside the CIC-DDoS2019 dataset [9]

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 4.2. 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).

Table 4.2: NSL-KDD Attack Categories

Category	Attacks
DoS	back, land, neptune, pod, smurf, teardrop, Mailbomb, Processtable Udpstorm, Apache2,Worm
U2R	buffer-overflow, loadmodule, perl,rootkit, Sqlattack, Ps
R2L	ftp-write, guess-passwd, imap, multihop, spy, Xlock,Xsnoop, Snmpguess, Snmpgetattack, Sendmail
Probe	ipsweep, nmap, portsweep, satan, Portsweep, Mscan

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 training 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 4.1 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.

4.1 Dataset Pre-processing

This work 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:

$$z_i = \frac{x_i - \min(x)}{max(x) - \min(x)} \tag{4.1}$$

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, the CIC-IDS2017 DoS dataset contains ten features containing zero values. Table 4.3 shows the list:

Table 4.3: 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

4.2 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.

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

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

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

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

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.4}$$

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

$$F1\text{-}score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4.5}$$

We have also utilized Confusion Matrix, a specific table layout that allows visualization of the performance of an algorithm in our work. Table 4.4 shows the definition of confusion matrix.

Table 4.4: Confusion Matrix

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

Table 4.5: 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 4.6: 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

4.3 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 4.5 displays the data partitioning of our model.

Table 4.7: 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 4.8: 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 4.9: 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

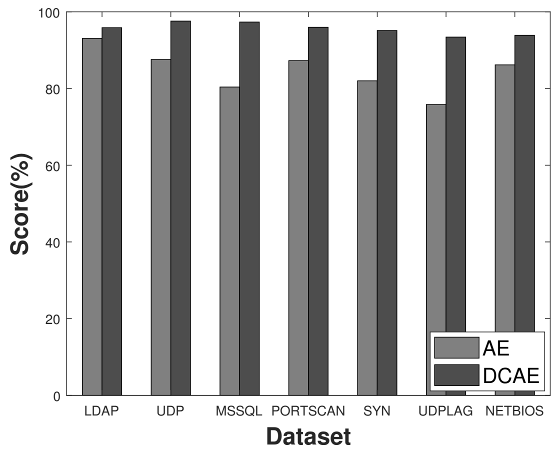
In our experiment, the linear layer takes the decoder output and reconstructs the input data. We used Contractive Loss as a cost function with Adam Optimizer and Sigmoid function for activation in hidden layers. We trained the model using 100 epochs and a batch size of 32. Table 4.6 lists all the hyper-parameters of our model. We executed several experiments using different values of experiment 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 tested the model's performance using different hidden layers, iteration, number of neurons per hidden layer, and the activation function. The best performance is achieved when we use two hidden layers. When the number of hidden layers increases, the model accuracy tends to decrease. Therefore, we used two layers in our proposed framework.

4.4 Experimental Results

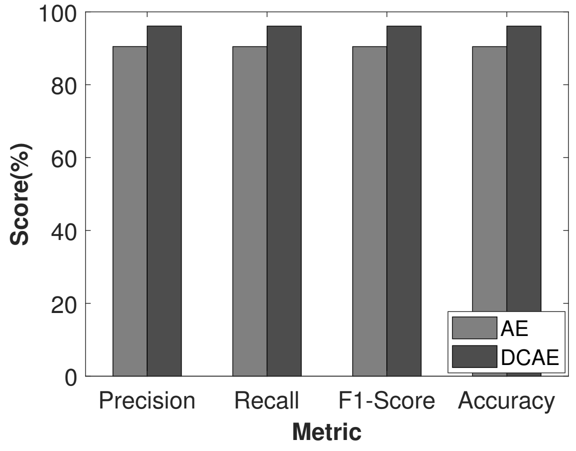
We compared our model with some of the most well-known deep learning models, Basic AE, Variational AE, and LSTM AE, to evaluate our proposed model for each of chosen datasets. We analyzed all methods' precision, recall, F-score, and accuracy values. Our model obtains an accuracy of 92.45% and 96.08% for the CIC-IDS2017 (DoS) and NSL-KDD datasets, respectively. Table 4.7 and 4.8 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 4.9. 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 4.2a, 4.2b, 4.2c.

Figure 4.3 presents our method's confusion matrix for NSL-KDD and CIC-IDS2017 datasets. Figure 4.3a 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 4.3b 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, where 729 of them were false positives and 653 false negatives.

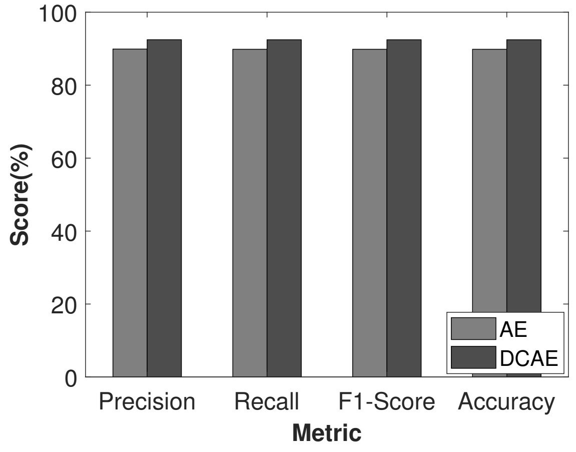
We have also used the Receiver Operating Characteristic (ROC) curve to show the efficacy of our model. The ROC curve shows the relationship between two parameters: true and false classes. The area under the ROC Curve (AUC) measures how well a model can distinguish between classes. Figure 4.4a shows that 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 4.4b.



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

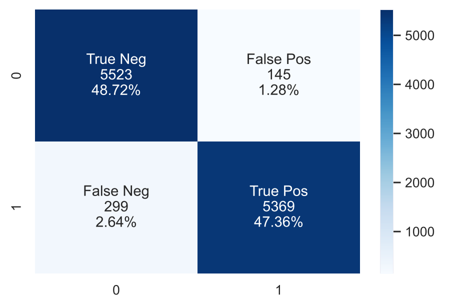


(b) AE and DCAE performance comparison on NSL-KDD dataset

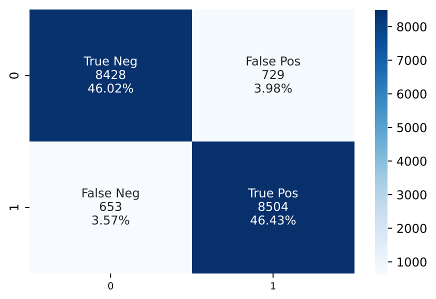


(c) AE and DCAE performance comparison on CIC-IDS2017 dataset

Figure 4.2: AE and DCAE performance comparison on different attack datasets

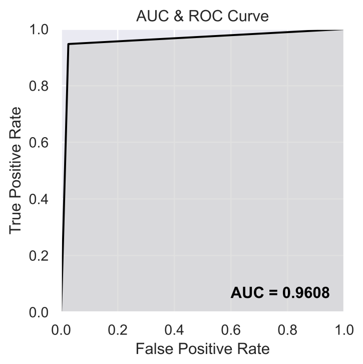


(a) Confusion Matrix of our proposed approach for NSL-KDD

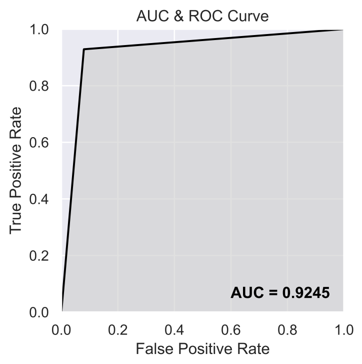


(b) Confusion Matrix of our proposed approach for CIC-IDS2017

Figure 4.3: Confusion Matrix of the evaluation dataset of our proposed approach



(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 4.4: Receiver Operating Curve (ROC) of our proposed approach

Chapter 5

CONCLUSIONS AND FUTURE WORK

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 work, 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. Our experiments show that the proposed model has the potential to detect anomalies in network traffic data. 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.

BIBLIOGRAPHY

[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`

VITA

Sharmin Aktar was born in Pabna, Bangladesh. She is the youngest member in her family. She completed her undergraduate degree of Bachelor of Science in Computer Science and Engineering (CSE) from Bangladesh University of Engineering and Technology (BUET) in 2016. After three years of her professional career as a software engineer in a tech company in Bangladesh, she joined the University of New Orleans (UNO) in Spring 2020 as a Graduate Research Assistant in the Computer Science department to pursue her Ph.D. This research work was supervised by Dr. Abdullah Yasin Nur (Assistant Professor, CS at UNO). Sharmin's research interest focuses on Network Security and Machine Learning. Apart from the study and research, she loves travelling and cycling.