

ENHANCEMENT OF ML BASED NETWORK ANOMALY DETECTION SYSTEMS WITH GENERATIVE MODELS

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY

IN

INFORMATION SYSTEMS

Submitted By:

MADHUR RAJPUT

(2K17/ISY/08)

Under the supervision of
Ms. ANAMIKA CHAUHAN



DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

JULY, 2019

CANDIDATE’S DECLARATION

I, MADHUR RAJPUT, Roll No. 2K17/ISY/08 student of M.Tech Information Systems, hereby declare that the project Dissertation titled “Enhancement Of ML Based Network Anomaly Detection Systems With Generative Models” which is submitted by me to the Department of Information Technology, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi

Madhur Rajput

Date:

CERTIFICATE

I hereby certify that the Project Dissertation titled “Enhancement Of ML Based Network Anomaly Detection Systems With Generative Models” which is submitted by Madhur Rajput, Roll No 2K17/ISY/08 Information Technology, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi

Ms Anamika Chauhan

Date:

SUPERVISOR

ACKNOWLEDGEMENT

I express my gratitude to my major project guide Ms. Anamika Chauhan, Assistant Professor, IT Dept., Delhi Technological University, for the valuable support and guidance she provided in making this major project. It is my pleasure to record my sincere thanks to my respected guide for her constructive criticism and insight without which the project would not have been shaped as it has.

I humbly extend my words of gratitude to other faculty members of this department for providing their valuable help and time whenever it was required.

Madhur Rajput

Roll No. 2K17/ISY/08

M.Tech (Information Systems)

E-mail: raajpootmadhur@gmail.com

ABSTRACT

A huge rise in network traffic data have brought challenges towards the security of data over network, servers and computers. This has been a challenge to the traditional intrusion detection system also, as the rapid change in technology there changes the intrusion style also. In recent years, new varieties of anomaly attacks have emerged rigorously which can't be detected by out dated intrusion detection systems. In order to tackle those intrusions we propose a machine learning based approach which implements an autoencoder and a dense neural network. Both autoencoder and dense neural network are types of artificial neural network, but they differ in the processing. Therefore the feature extraction phase of our IDS model is designed on the basis of autoencoder technology and model creation is done on the basis of dense neural network.

We also implement a convolutional neural network, another subclass of artificial neural network as intrusion detection system.

The dataset used is Intrusion Detection Evaluation Dataset (CICIDS2017). This dataset is of new generation in terms of attacks it contains. The attacks present in the dataset are new types of attacks which are generally used by attackers in real network for the purpose of stealing data.

The results obtained by two models are compared i.e. the accuracy in the detection rate is compared and thus after comparing the results the model implemented on dense neural network and autoencoder shows the better accuracy and lesser false alarm rate.

CONTENTS

CANDIDATE’S DECLARATION	ii
CERTIFICATE	iii
ACKNOWLEDGEMENT	iv
ABSTRACT	v
CONTENTS	vi
List of Figures	viii
List of Tables	x
CHAPTER 1 INTRODUCTION	1
1.1 INTRUSION DETECTION SYSTEM	1
1.1.1 Introduction	1
1.1.2 Data Preprocessing	2
1.1.3 Model Generation	3
1.1.4 Evaluation	3
1.1.5 Deployment in Real Network	3
1.2 ARTIFICIAL NEURAL NETWORK	3
1.2.1 Structure of ANN	4
1.2.2 Types of Artificial neural network	5
1.2.3 CNN (Convolutional Neural Network)	6
1.2.4 Machine learning and ANN	7
1.3 AUTOENCODERS	8
1.3.1 Structure of Autoencoders	8
1.3.2 Properties of an Autoencoder	10
1.3.3 Hyperparameters to Train an Autoencoder	11

1.3.4	Types of Autoencoders	11
CHAPTER 2 RELATED WORK		14
CHAPTER 3 EXPERIMENTAL APPROACH		20
3.1	DATASET	20
3.1.2	Types of Attacks in Dataset	21
3.2	DATA PREPROCESSING	22
3.3	FEATURE EXTRACTION	25
3.3.1	Vanilla Autoencoder	25
3.4	MODEL GENERATION	29
3.5	CNN MODEL	32
CHAPTER 4 RESULTS		33
CHAPTER 5 CONCLUSION AND FUTURE WORK		36
References		37

List of Figures

Figure 1.1 Intrusion Detection System	3
Figure 1.2 Nerve Cell	4
Figure 1.3 General Structure of ANN	5
Figure 1.4 Supervised Learning	6
Figure 1.5 K-means Clustering	8
Figure 1.6 Block diagram of Autoencoder	8
Figure 1.7 Structure of Autoencoder	9
Figure 1.8 (a) Denoising Autoencoder	12
Figure 1.8 (b) Sparse Autoencoder	12
Figure 1.9 Variational Autoencoder	13
Figure 1.10 Contractive Autoencoder	13
Figure 3.1 Raw Dataset	20
Figure 3.2 Types of Attacks	21
Figure 3.3 Number of Attacks	22
Figure 3.4 Data with all Numeric Values	23
Figure 3.5 Training Set	23
Figure 3.6 Validation Set	24
Figure 3.7 Test Set	24
Figure 3.8 ReLU Function	26
Figure 3.9 Sigmoid Function	26
Figure 3.10 Autoencoder Summary	27
Figure 3.11 Binary Cross Entropy Function	28
Figure 3.12 Architecture of Autoencoder	29
Figure 3.13 Summary of Dense Neural Network	29
Figure 3.14 Softmax Function	30
Figure 3.15 Block Diagram of Model	31
Figure 3.16 Summary of CNN Model	32
Figure 4.1 Accuracy Comparison	34
Figure 4.2 Detection Rate Comparison	34

List of Tables

Table1. Accuracy of DNN-Autoencoder and CNN

33

CHAPTER 1

INTRODUCTION

1.1 INTRUSION DETECTION SYSTEM

1.1.1 Introduction

With the rapid development of internet nowadays, the network is getting more and more prone to the cyber-attacks, due to which the threat of losing the data and information increases. There are many ways in which a cyber-attack can be performed and also there are many different types of attacks for example Denial of Service (DoS), SQL injection, Man-in-the-middle attack, phishing, as stated in [20]. As we have discussed, there are a lot of ways in which an attacker can attack a network to steal the data; therefore in order to prevent the attack, different technologies are being used by the firms like firewall and access control.

According to [10], In spite of the fact that there are so many technologies to prevent the intrusion, still the network is vulnerable to so many undetected attacks. Therefore in order to prevent such attacks Intrusion Detection Systems (IDS) are developed and still more are being developed day by day using different kind of technologies.

As discussed in [11] [14] [21], IDS can be categorized as HIDS and NIDS i.e. host based intrusion detection system and network based intrusion detection system. HIDS operate on an individual system and possess the complete knowledge about the data and processing of that system. These are equipped with the servers and provide security to the mails, servers of a firm. HIDS provide high security as they have the complete knowledge of data and servers of a computer system. But here our concern is NIDS, NIDS operate on network. NIDS are deployed to handle the traffic over network. NIDS are used to restrict the malicious activities taking place over the internet, they does not replace the security implementations like firewall rather they add to them. The NIDS are used to detect those attacks which can't be detected by the basic security parameters. There is one more class of IDS i.e. Hybrid IDS, which possesses the properties of both HIDS and NIDS. These IDS can be deployed on both host and network Now onwards we shall use the term IDS for NIDS.

Further IDS can be categorized into signature based and anomaly based IDS. Anomaly based IDS analyses the network traffic and if in the received data, there is any kind of deviation from the normal behavior then it raises a flag but False alarm rate (FAR) is high with anomaly based IDS which makes its implementation quite onerous, the reason for the same is that attacker attacks the network with the values close to the training values of IDS. Whereas on the other hand Signature based IDS has its own database of known intrusions and it detects those intrusions only.

Intrusion detection can be stated as a classification problem, hence many Machine Learning techniques are used in order to build IDS like SVM, Random Forest, Swarm Optimization and Convolutional Neural Networks [17]. These IDS are first trained on the basis of nature of intrusions then they are tested with some sample data after that these systems are deployed in the networks.

These IDS face some problems like the first problem is that nature of attack is variable, it changes vigorously and in order to cope with such changing attacks, it is important to make IDS which can detect such types of attacks. The second problem is of overfitting, the machine learning methods usually face the problem of overfitting. Therefore it is immensely important to distribute the dataset into training data and test data and also balance must be maintained among the features i.e. irrelevant features and redundant features must be taken care of. The third problem is labeling the dataset of intrusion. The dataset are so complex that labeling becomes a typical task. Efforts have been made to simplify the task of labeling the dataset.

The problems faced are tackled by the IDS throughout its working phases. An IDS works in different phases and these phases are explained below.

1.1.2 Data Preprocessing

Raw dataset contains different types of data, redundant data and also some features which have non-numeric values. Therefore in order to handle these problems, data preprocessing is carried out. Data normalization is also the part of data preprocessing, min-max normalization is a generally used method for data normalization. Also important features which contribute in the implementation of IDS are extracted and labeled. Feature extraction can be carried out using different methods of machine learning like SVM, PCA [23].

1.1.3 Model Generation

Specific machine learning tool is wielded to generate IDS model. Model building can be seen as the core of IDS because model detects the intrusion. The dataset is divided into training set, validation set and test set. Further on the basis of data, training and learning processes of IDS model are carried out. Training is validated using validation data. After validation, on the basis of test data the model is tested and this process is performed again and again in order to improve the accuracy of IDS.

1.1.4 Evaluation

The results of IDS are evaluated and on the basis of evaluation, modifications are made to the IDS. All the parameters are evaluated like accuracy, time taken to detect the attack.

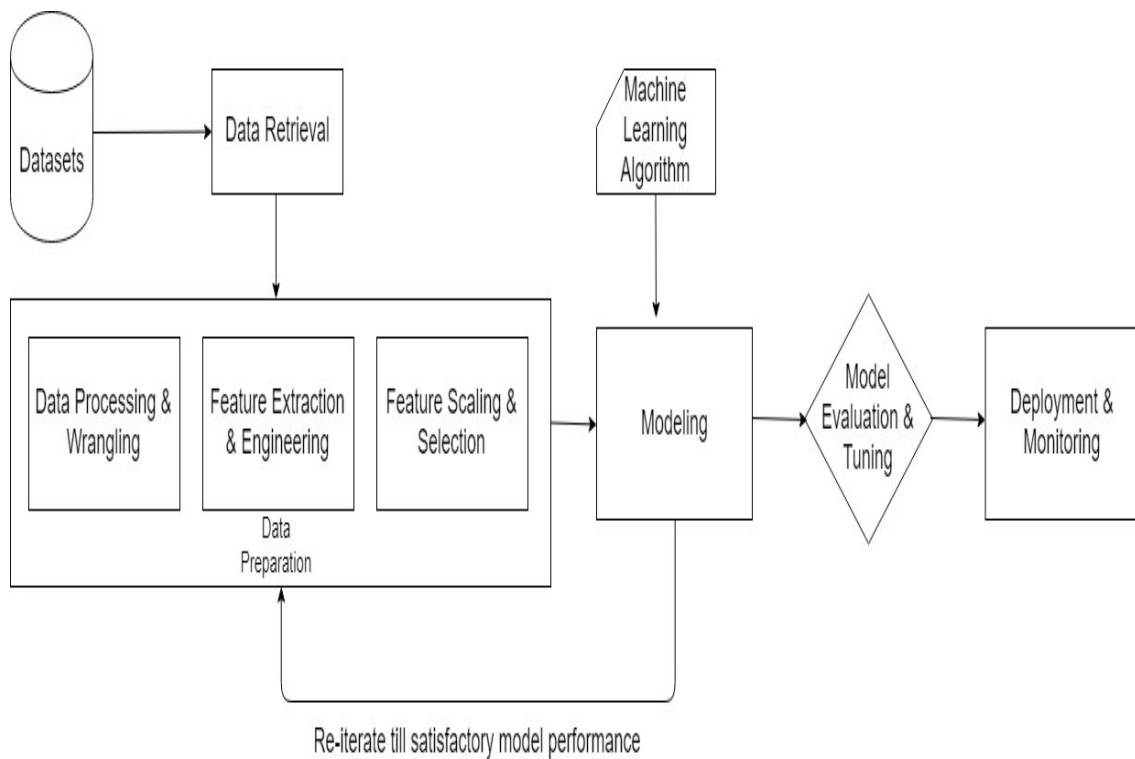


Fig.1.1 Intrusion Detection System

1.1.5 Deployment in Real Network

After all the analysis of system, the model is implemented in the real time network to work as intrusion detection system.

1.2 ARTIFICIAL NEURAL NETWORK

1.2.1 Structure of ANN

A computer system consists of a series of simple and highly interconnected processing elements that processes information in response to the dynamic state of external inputs.

Basic Structure of ANNs

The basic thought behind an ANN is to build an analogy with the working of the human brain. The human brain is composed of dendrites and neurons and in the same manner the ANN is built. So the idea is to emulate the working of human brain using wires and silicon as dendrites and silicon respectively.

There are billions of nerve cells in the human brain and they are connected to other thousands of cells by axons. Dendrites accept the stimuli from the external environment via sensory organs, further these stimuli are converted to the electric impulse which travels through the neural network and then the neurons communicate with the other neurons in order to handle the issue. According to [25] a biological neuron can be seen as in figure 1.2.

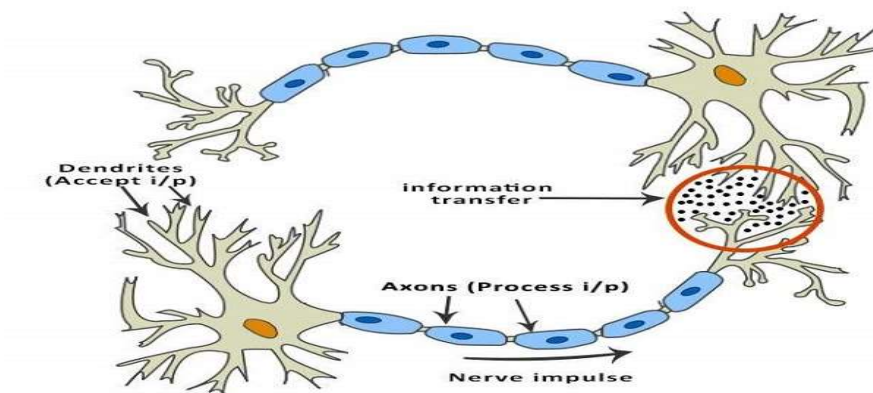


Fig.1.2 Nerve Cell

In the similar way, ANNs are comprised of multiple nodes which work like neurons present in the human brain. The input data can be accepted by nodes for the purpose of simple data operations and the results of these operations are passed to other nodes. The output of every node is known as its activation or node value. The nodes are connected with the links and these links possess some weights. Just like human brains, ANNs also possess the ability to learn and the learning process is carried out by adjusting the weight values of the links. The simple example of an ANN is shown in the figure 1.3.

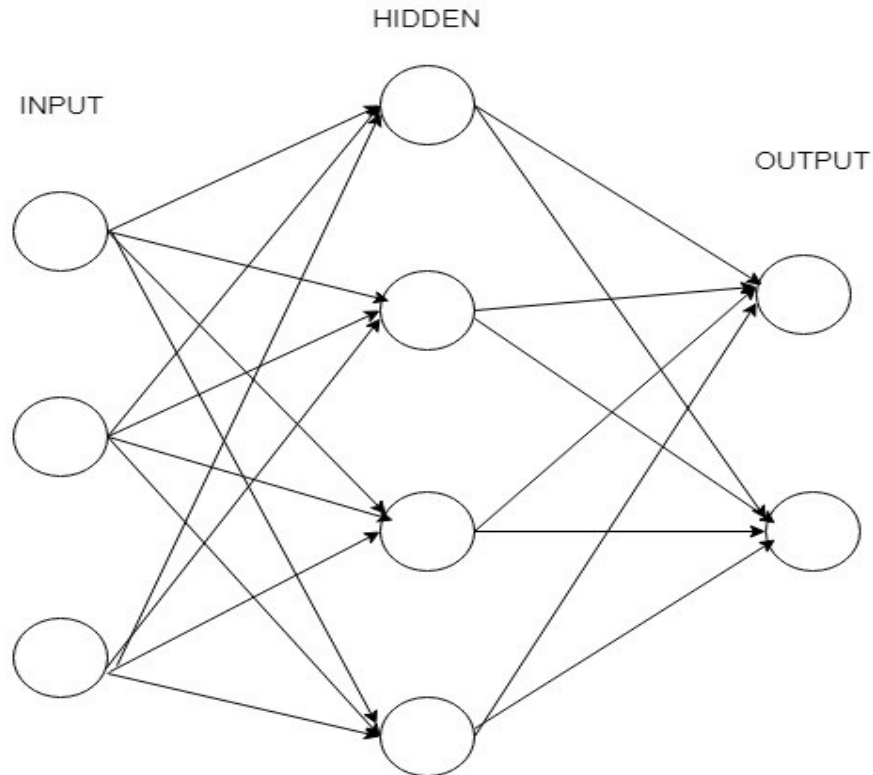


Fig.1.3 General Structure of ANN

1.2.2 Types of Artificial neural network

- a) Feed Forward ANN
- b) Feedback ANN

1.2.2.1 Feed Forward ANN

In FFANN the information flow is unidirectional i.e. information flows from the input nodes to the hidden nodes and from hidden nodes to the output nodes. There is no loop in feed forward ANN. FFANN is generally used in pattern recognition, image classification.

1.2.2.2 Feedback ANN

In Feedback neural network, information flow is bidirectional which means information can flow from output layer to hidden layer and input layer. Feedback ANN is also known as recurrent neural network (RNN). On the basis of feedback, this network can adjust its weights and can reduce the error in the next iteration. Thus recurrent neural network can come in the state of equilibrium by traversing the information forward and backward.

1.2.3 CNN (Convolutional Neural Network)

A Convolution Neural Network can capture the Spatial and Temporal dependencies successfully in an image by application of various relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

The main role of the CNN is reduction of times into something which is easier to process, without losing features which are used for prediction purpose. Its major applications are in the areas of Computer Vision, Image & Video recognition, Image Analysis & Classification, Media Recreation, Recommendation Systems and Natural Language Processing. In a CNN model each input image passes through a series of convolution layers with filters, Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1.

Convolution layer:

It is the first layer which helps in extraction of features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters.

ReLU function:

ReLU stands for Rectified Linear Unit and is used for non-linear operations. Its equation is given as:

$$f(x) = \max(0, x)$$

ReLU function is used to introduce non linearity in our CNN. The real world data requires CNN to learn from non-negative linear values, hence we use ReLU.

Pooling:

Pooling layers are used to reduce the number of parameters in very large images. It is used to reduce dimensionality size. Spatial pooling also known as subsampling or down sampling are used to reduce the dimensionality of each map but retains the important information [19].

Different types of spatial pooling are:

1. Max Pooling: It takes the largest element from the rectified feature map
2. Average Pooling: It takes the average of all elements in the feature map

3. Sum Pooling: It takes the sum of all the elements in the feature map

1.2.4 Machine learning and ANN

Machine Learning can be defined as making the machine learn on their own from the data, previous experience without explicitly programming them. There are broadly 2 types of machine learning: supervised learning and unsupervised learning [24].

1.2.4.1 Supervised learning

In supervised learning we have a set of data which is labeled i.e. we have sample data from which the algorithm can learn.

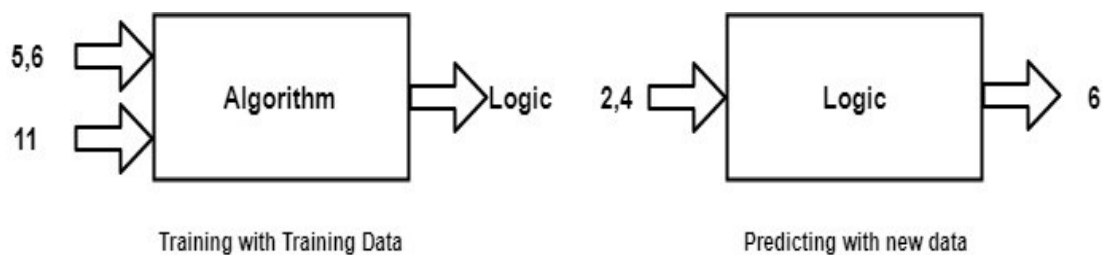


Fig.1.4 Supervised Learning

As shown in Fig. we can see that training data is used to train the model to form the logic. After the model has been trained, it is used to make a prediction or decision when new data is given to it.

Types of Supervised Learning

- **Classification:** Classification problems deal with the prediction of a discrete value output zero or one. Here the output is divided into a certain number of classes and we try and classify data into these given classes. For example: We have a database of customer accounts and we would like to examine if the customer data has been compromised or not. This is an example of a classification problem where the 2 classes will be compromised data and non-compromised data.
- **Regression:** Regression problems deal with prediction of a continuous valued output. Regression doesn't have classes and the outputs are not discretely defined are are continuous. For example: There is a large inventory of identical items and we have to predict which items will sell in the next 3 months. This is a regression problem as we have thousands of items so we'll treat it as a continuous value and the number of items to be sold will be a continuous value.

1.2.4.2 Unsupervised learning

Unsupervised learning is called as learning with unlabeled data. In this, we do not have a set of labelled data to train our model. The computer tries to find patterns in data to find results. Unsupervised learning can be used for market segmentation, astronomical data analysis, and social network analysis.

An example of unsupervised learning is clustering. Clustering is known as the technique in which a set of objects are grouped together such that objects in the same group are more similar to each other than they are to objects in the other group.

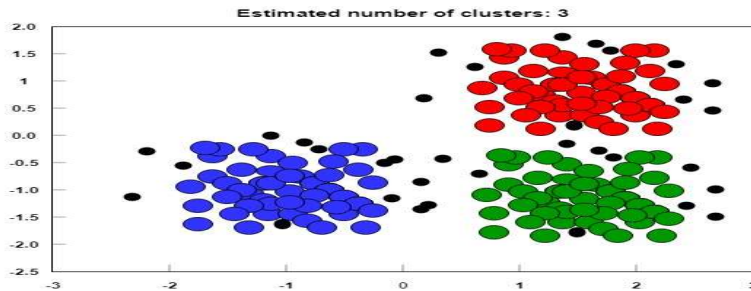


Fig.1.5 K-means Clustering

1.3 AUTOENCODERS

1.3.1 Structure of Autoencoders

Autoencoders are a specialized class of feedforward neural networks that are compressed into integrated code and then regenerated to produce the output same as input using machine learning. But the objective of an autoencoder is not just duplicating the input to the output. But the training an autoencoder in latent space makes it worth as an autoencoder acquires some properties during the training [22]. An autoencoder is composed of three parts.

- i. Encoder
- ii. Code
- iii. Decoder

A simple example is shown in the figure 1.6.

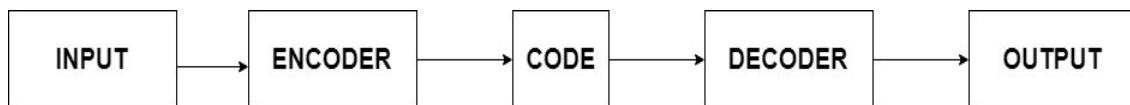


Fig.1.6 Block Diagram of Autoencoder

Encoder:

The input data is encoded and compressed in this layer with the reduction in the dimensionality. This compressed image is also known as the distorted version of the image. The encoding function can be written as

$$h = f(x)$$

Code:

This layer has the compressed input for the decoder part. It is also known as the bottleneck layer and this is the most important part of the network because the compression part is done by this layer. If the bottleneck layer is not there then the data will move as it is throughout the network. Therefore in order to compress the data, the bottleneck layer is needed. It decides which features are to be kept and which should be removed from the input data. Only the relevant features are retained from the input space.

Decoder:

It decodes the compressed data and gives back the lossy original data i.e. the data obtained from the decoder does not have the complete information as it had in the input. The decoding function can be written as

$$r = g(h)$$

Therefore an autoencoder can be represented as

$$g(f(x)) = r$$

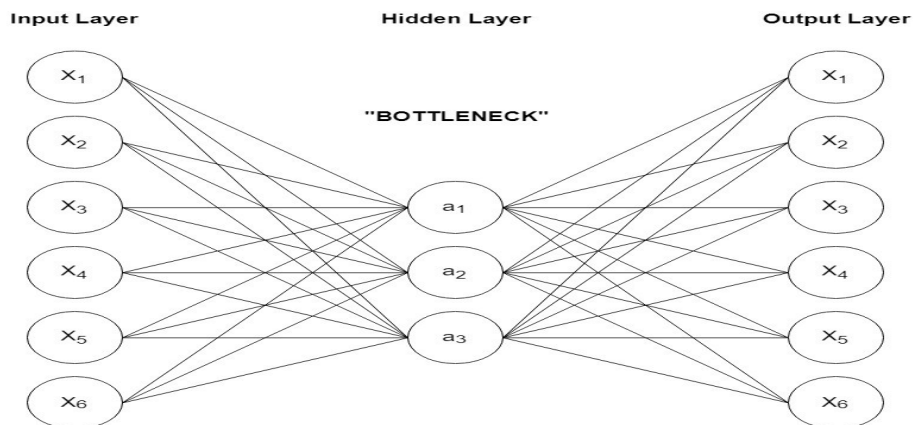


Fig.1.7 Structure of Autoencoder

If the bottleneck layer has lesser dimensions than the input layer, then autoencoder can extract the suitable features from the input and the process of imitating the input is justified. In this case the autoencoder is considered as undercomplete. But if the bottleneck layer has been assigned more dimensionality than the input layer, the autoencoder is called as overcomplete because in this case autoencoder is not extracting the useful features from the input domain, it is just performing the duplicating task as it is. This can also be achieved if the code layer has the same dimensionality as the input. But in these cases the encoder does not learn anything important about the dataset. An autoencoder, unlike the other compression and decompression algorithms learns from the data itself whereas the training of the algorithms like JPG and ZIP have different datasets.

An autoencoder is used for the purpose of data denoising and the reduction in the data dimensionality with the help of compression method. Noise is the random variation in the data, autoencoder first generates the noise and then denoise the data in order to compress the data. The reduction in dimensionality in the bottleneck layer also helps to compress the data.

1.3.2 Properties of an Autoencoder

Data Specific:

Autoencoders can compress the data which is similar to their training data, since the compression is dependent on the training data, therefore autoencoders are different from traditional compression algorithms.

Lossy:

The decoded outputs will not exactly be same as the inputs. There will be some degradation or loss of information in the output. Lossless compressions can't be obtained using the autoencoders.

Unsupervised:

Autoencoders learn in unsupervised manner, moreover they are known as self-supervised neural networks as they don't need any raw data to train on rather they generate the labels on their own.

1.3.3 Hyperparameters to Train an Autoencoder

Bottleneck size:

Number of nodes in the bottleneck layer depicts the degree of compression i.e. lesser the number of nodes in the layer, more will be the compression. Therefore in order to get higher compression the size of bottleneck layer is kept small.

Number of layers:

We can create as many layers as we want in an autoencoder i.e. an autoencoder can be as deep as we want.

Number of nodes per layer:

The structure of an autoencoder is a stacked structure where the number of nodes in the encoder part decreases with the number of layers and on the other hand, the number of nodes in the decoder part increases with number of layers. Structures of both encoder and decoder are identical, speaking in terms of layers.

Loss Function:

Loss function is used depending on the input, if the input is in the range $[0, 1]$ then binary crossentropy is used as loss function otherwise mean squared error (mse) is used.

1.3.4 Types of Autoencoders

- i. Sparse Autoencoders
- ii. Denoising Autoencoders
- iii. Variational Autoencoders
- iv. Contractive Autoencoders

1.3.4.1 Sparse Autoencoders

According to [12], Sparse Autoencoders do not rely on reducing the number of nodes from the hidden layer rather they depend on the activation function. A fixed number of nodes are kept active in this type of autoencoder and others are kept inactive i.e. only a small number of neurons will have a non-zero value. Therefore input is given as combination of a few number of neurons and autoencoder is thus forced to discover the information from the data. Training and feature extraction also depend on the number of active nodes.

1.3.4.2 Denoising Autoencoders

It is the type of a basic autoencoder, the input fed into the network is a noisy input. The idea is to force to the hidden layer to find some more features from the input so that the training and learning processes should not depend on the raw input. Autoencoder should be able to reconstruct the data from the noisy input and can generate the output similar to the input.

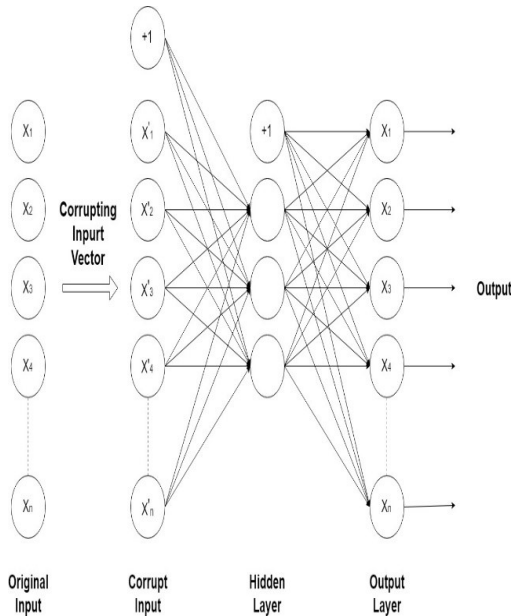


Fig.1.8 (a) Denoising Autoencoder

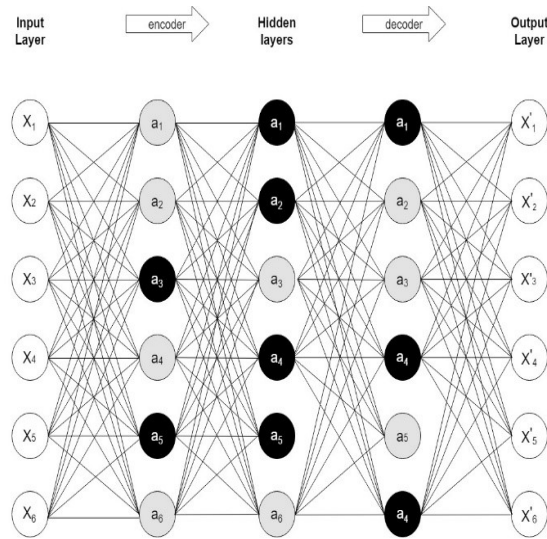


Fig.1.8 (b) Sparse Autoencoder

1.3.4.3 Variational Autoencoder

Variational Autoencoders are no different than other autoencoders. The only difference is that, in encoder part of variational autoencoder, we know the distribution of features whereas in normal autoencoder we don't know the distribution of features. It can use different types of distribution like Gaussian distribution and unit normal distribution [12]. Encoder can do sampling of data from hidden features using the distribution and further when this data is fed into the decoder part, the decoder can generate the objects similar to the input. Apart from compression variational autoencoder can also generate the new objects with a minute difference from the original object.

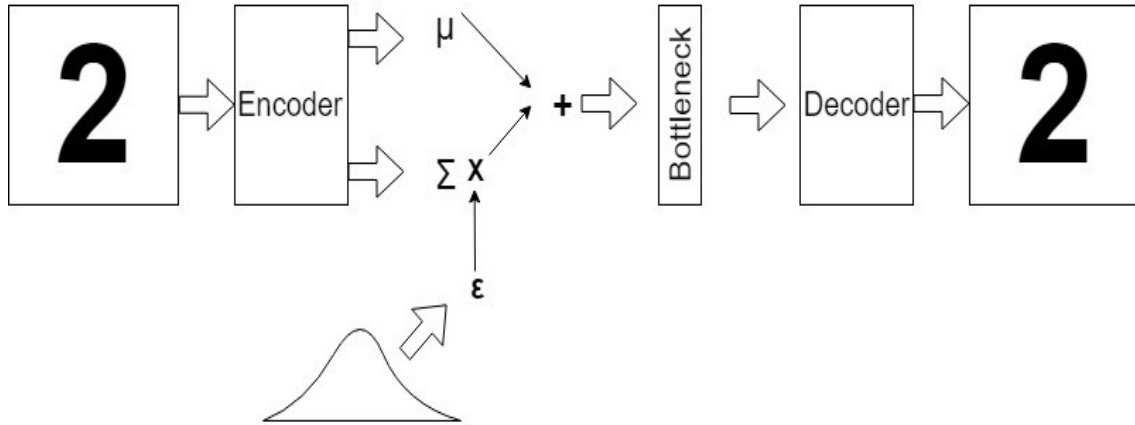


Fig.1.9 Variational Autoencoder

1.3.4.4 Contractive Autoencoder

Contractive autoencoders are similar to denoising autoencoders because in both the types, manipulations are done with input in order to make the learning process of encoder part more effective. A penalty term is applied to feature extraction algorithm in the encoding part, so that it becomes less sensitive to a small manipulation. Therefore the output can be seen as the result of reduced sensitivity of encoder. Let's understand with the help of an example, suppose we have the dataset of bikes and encoder learns about the left view and right view of a bike then contractive autoencoder is sensitive about both left and right view, but it will be insensitive towards other views of the bike. This is how a contractive autoencoder works.

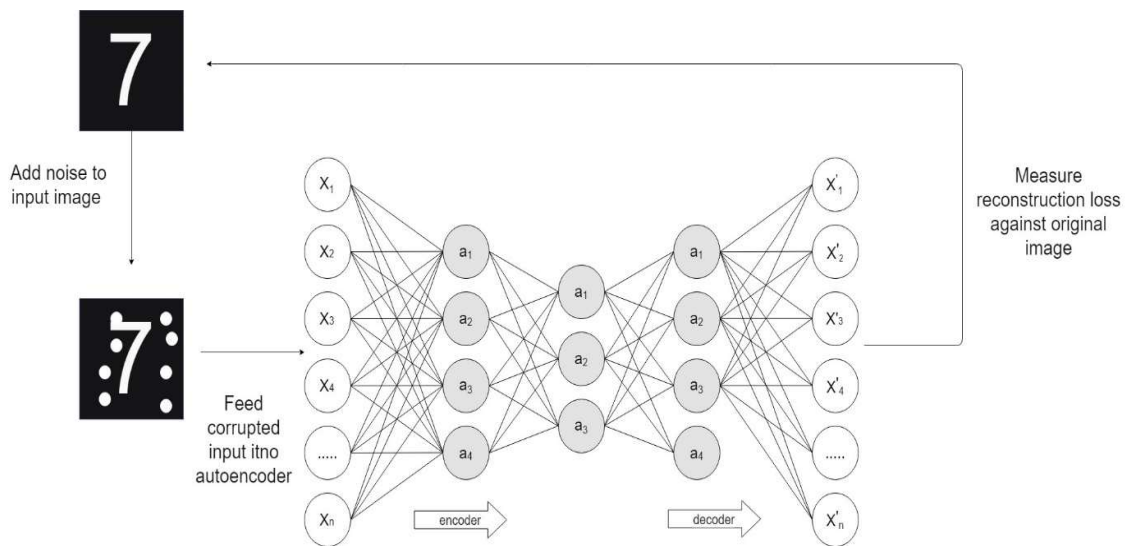


Fig.1.10 Contractive Autoencoder

CHAPTER 2

RELATED WORK

In this chapter we have reviewed the work done in the field of intrusion detection system. We studied the techniques deployed so far and how they are efficient and up to date.

Neural nets have always been the source of attraction for the researchers in the field of machine learning and intrusion detection. RNN is different from traditional ANN in terms of the data-flow. The data flow in ANN is unidirectional that is the data flows from input layer to the hidden layer and from hidden layer to the output layer whereas in RNN the information processed at some point can be used further, as given in [16]. This purpose of remembering and forgetting is served by GRU (Gated Recurrent Units). A hypothesis function is used to estimate the probability in order to build a multi-classifier so that the diverse network attacks can be classified. The data preprocessing is done by GRU module, the GRU layers present in the module extract and store the features, as discussed in [4]. The GRU output is mapped by MLP module and thus makes a non-linear classification decision. The GRU and MLP are both different types of neural networks. GRU has memory and complex structure whereas MLP is a simple structure and has easy calculations. The classification probability is normalized by Softmax layer and shown as the final result.

As mentioned in [1], Convolutional neural network (CNN) is a method described in the context of deep learning. In other machine learning methods the training can be called as hand-based training whereas CNN is a self-trained method i.e. it learns itself on the basis of previous outputs and later correcting them by adjusting the weights attached to the nodes. Basically CNN is used in the field of artificial intelligence for the purpose of image classification. When implemented as an IDS, features dataset is a $l*m$ dataset, therefore in order to implement IDS as CNN the dataset is converted to $n*n$ dataset. The method used for feature removal or feature selection is coefficient of variance, the features having the minimal CV should be removed All neurons can share the same convolution kernel and the number of kernels determines the

number of weights. The activation function used in this technique is rectified linear unit (ReLU) in order to generate the feature vector and the method used for the back propagation is gradient descent method for the purpose of training and reduced error rate. The method also faces the problem of imbalanced distribution of data and in order to solve the classification imbalance problem the cost function-based method is used. As different samples are present in different proportions therefore the weights of the cost functions are adjusted accordingly, if samples are high in proportion then weights they will have smaller weights and if they are small in number then those will have higher weights.

As stated in [2], the particular method of intrusion detection the raw data is separated into training data and test data further the training data is divided into the reduced training set and the evaluation set. The purpose of the evaluation set is to validate the training set, but the test set is completely different from the training set. The non-numeric data or symbolic data is converted to the numerical data. The next step is to normalize the data, the process of normalization is done to so that the values can be kept in the range of optimal processing. In this method the normalization is done in two steps. First the logarithmic normalization is applied so that the data can be kept in an acceptable range and in the second step the values are linearly capped in order to keep them in a particular range. After the normalization process the process of feature extraction takes place. Feature Extraction algorithm is based on Information Gain, a threshold value is set for the information gain and the features having information gain more than the threshold are loaded in the algorithm.

For the training purpose the following steps are followed

- (i) Forward Propagation
- (ii) Back-propagation of the computed error
- (iii) Updating the weights and biases.

The neural network used in this technique is trained using back propagation supported by stochastic gradient descent for the weights update. The cost function calculates the difference between the target and the obtained output.

The particle swarm optimization technique is developed on the basis of social behavior metaphor. It can be referred as exploration-exploitation tradeoff [18]. Exploration means to find an optimum solution globally i.e. assessing the various regions of the problem space and exploitation means to look for the solution in the neighborhood so that the optimum solution

can be found quickly. The dataset used in this technique is KDD Cup 99. The velocity and the position of a particle are formulated in order to find the optimal solution in the dataset, mentioned in [5]. PSO based optimized technique helps an artificial neural network to choose its weights and the problem of hidden neurons is solved by the activation function used.

IDSGAN method emphasises on the attacks which can deceive the traditional IDSs and this misclassification is done desirably by using adversarial malicious traffic examples to train the model, as mentioned in [9].

The model is composed of two networks i.e. the generator network and the discriminator network. The weights of generator are set different so that the adversarial malicious traffic can be generated it can dodge the pre-implemented IDS. The generator has a five layered neural network. Training of Generator depends on the feedback of Discriminator. Weights of Generator are updated according to the outputs of the Discriminator. The Discriminator is used as IDS, it detects the both normal attacks and adversarial malicious attacks and send the feedback to the Generator. The Discriminator is a multi-layer neural network and its training data comprise of normal traffic and malicious traffic. For the training purpose of the Discriminator, the traditional IDS classifies the normal attack data and the adversarial attack data. Then these classified attacks are used to train the discriminator so that discriminator can work as a traditional IDS. In order to calculate the performance of IDSGAN the detection rate and the evasion increased rate are calculated comparatively. Hence the suggested method shows the compatibility with the modified features and works better with slight changes in the dataset whereas other traditionally implemented IDSs can't perform expectedly.

In KNN-Mars algorithm as the name suggests the method uses two algorithms i.e. KNN (k-nearest neighbour) and MARS (multivariate adaptive regression splines). MARS is a non-linear regression function, it works like a step function. Points where non-linearity arise are called knots. The knots or cut-points in a polynomial regression model are captured by MARS function. The knots are created by MARS according to the data and the polynomial distribution function. First the knots are generated and after that those knots which have minimal impact on the distribution are pruned in order to avoid overfitting. KNN is a clustering algorithm defined in machine learning aspect. Functioning of KNN depends on similarity in data units. Clusters of similar objects are formed. K is a natural number which depicts the number of clusters to be made. The algorithm works on the basis of distance calculation. It measures the distance from a query to other points in the dataset and thus the clusters are made accordingly

i.e. those nearer to a particular query point will be enclosed in the cluster of that point. The basic version of KNN-MARS method is slow. In order to fasten the technique, those neighbours which have negligible impact on the model are pruned. MARS algorithm also helps in speeding up the technique because the training data points lie close to the decision boundary and are used repeatedly. Therefore this method can be seen as an improvement to the basic NN classifier. If k is small then model behaves as KNN and if k is large then the model behaves as MARS, so the method KNN-MARS is best suited for the moderate values of k , as discussed in [6]. The method uses 10-fold cross validation i.e. dataset is divided into 10 sets and out of these 10 sets, 9 are training sets and 1 is test set. Model is trained each time with a particular dataset and then it is tested on the test set, further on the basis of test results the model is evaluated and summarized.

As discussed in [13], SVM stands for support vector machine, is a machine learning technique generally used for the purpose of classification. It's a kind of supervised learning and thus need a labelled training data. The classifier generated by SVM is a hyperplane. Dimensions of hyperplane depend on the number of classes in which data needs to be classified. If we have n classes then hyperplane would be n -dimensional. SVM works better with the small samples of data in comparison to other algorithms. SVM also solves the problem of non-linear and high dimensional data but in case of big data, the training and the testing times are long. SVM also faces the problem of high error rates. Therefore in order to solve these problems SVM is implemented with GA i.e. genetic algorithms. Genetic algorithms work on the basis of mutation and crossover i.e. they follow biological genetics. In order to converge the algorithm, the crossover probability and mutation probability are adjusted according to the fitness function and the evolutionary algebra. The selection process in GA is designed so that the better individuals can be chosen and the diversity can be maintained. The process of feature selection in this technique is performed by GA. The feature chromosomes are created by process of mutation and their fitness function values are calculated. Features having the maximum fitness function values are selected and added to the features subset and further these features are used to detect the intrusion, mentioned in [7]. SVM classifier is trained on the basis of feature subsets generated by GA. Both the training and testing of SVM are based on the features extracted by GA. Thus this method works with optimum accuracy with the combination of duet.

In the method CNN with feature reduction, the first step in the method is same as used in other methods as well i.e. data pre-processing. At first the features having non-numeric values are converted to the numerical values and further the data is normalized, MIN-MAX formula is

used for the purpose of same. The 1-D dataset is converted to 2-D dataset in order to make it suitable for CNN. But in order to handle the data redundancy and increase the learning rate of the network, dimensionality of features is reduced. For the purpose of dimensionality reduction PCA (principal component analysis) and auto-encoder methods are used, as mentioned in [3]. Further this reduced dimensional data is converted to 2-D data and mapped into the input layer of CNN. In the next step, CNN learns the feature information, on the basis of which intrusion properties are derived. The output layer converts the result into 1-D array for the classification purpose. At last output is presented with softmax classifier and CNN and intrusion characteristics are shown. The method also uses batch normalization for the fast learning rate of the network. Batch normalization helps the learning process to cop up with change in distribution of input. Thus these different methods implemented with CNN make a better IDS.

Naïve Bayes and Adaboost IDS is a combination of one weak and one strong classifier, as discussed in [8]. Naïve Bayes is considered as a weak classifier and AdaBoost is considered as a strong classifier. Naïve Bayes classifier works on the principle of Bayes theorem. According to [15], the equation of Bayes theorem can be given as

$$P(c/x) = P(x/c)P(c)/P(x)$$

- $P(c/x)$ is the posterior probability of *class (target)* given *predictor (attribute)*
- $P(c)$ is the prior probability of *class*
- $P(x/c)$ is the likelihood which is the probability of *predictor* given *class*
- $P(x)$ is the prior probability of *predictor*

On the basis of probability Naïve Bayes classifies the data in to the different classes.

Adaboost stands for adaptive boosting, the algorithm work to strengthen the weak classifiers. The output of Naïve Bayes is combined with Adaboost to give the final output. Adaboost is adaptive in nature as it can adjust with noisy data and outliers, also it is easy to implement.

For the implementation of IDS, the stepwise process takes place. At first the features are extracted and the method used is two-second time window. Adaboost is also used to label the data. Therefore combination of strong and weak classifier works well in terms of IDS.

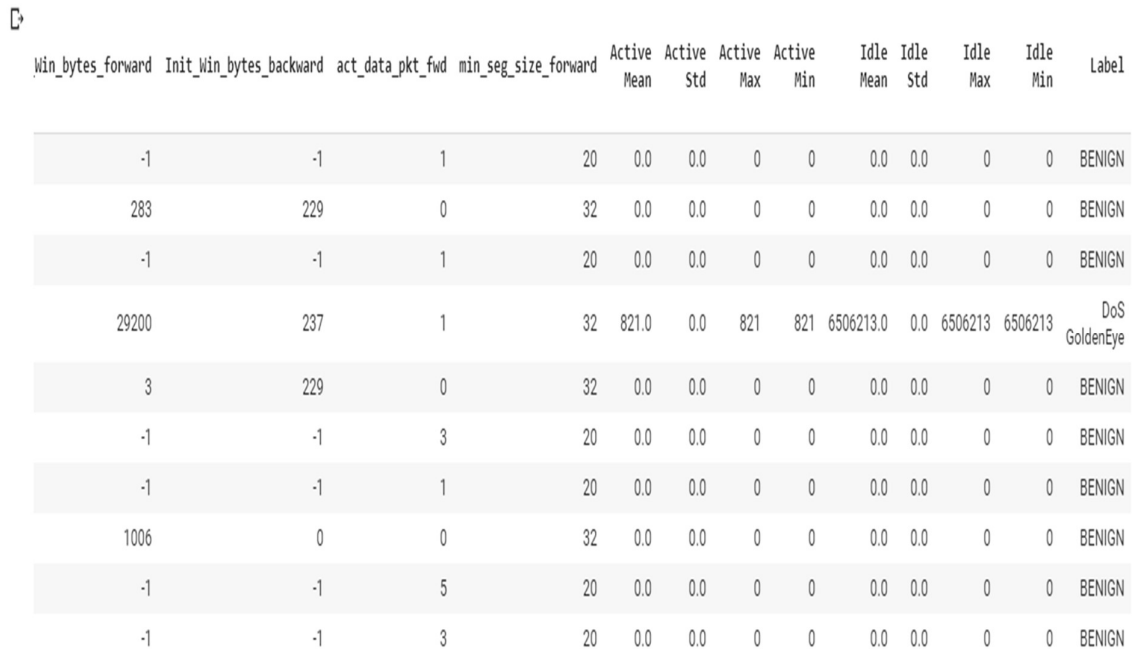
CHAPTER 3

EXPERIMENTAL APPROACH

3.1 DATASET

We get our dataset online from [26]. The dataset consists of benign and the most up-to-date common attacks, which resembles the true real-world data (PCAPs). It also includes the results of the network traffic analysis using CICFlowMeter with labelled flows based on the time stamp, source and destination IPs, source and destination ports, protocols and attack (CSV files).

The datasets used so far are out of date and unreliable as the traffic diversity is deficient in them. Those dataset also do not cover the variety of different known attacks. But our dataset covers the common attacks which are similar to the real world attacks. A sample of dataset is shown in figure 3.1.



win_bytes_forward	init_win_bytes_backward	act_data_pkt_fwd	min_seg_size_forward	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
-1	-1	1	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
283	229	0	32	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
-1	-1	1	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
29200	237	1	32	821.0	0.0	821	821	6506213.0	0.0	6506213	6506213	DoS GoldenEye
3	229	0	32	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
-1	-1	3	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
-1	-1	1	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
1006	0	0	32	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
-1	-1	5	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
-1	-1	3	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN

Fig.3.1 Raw Dataset

Initially the dimensions of our dataset are [692703, 79] i.e. we have 692703 rows and 79 columns in our dataset. Now from our dataset, we collect the set of attacks i.e. different types of attacks present in our dataset.

```
{'BENIGN',  
  'DoS GoldenEye',  
  'DoS Hulk',  
  'DoS Slowhttpstest',  
  'DoS slowloris',  
  'Heartbleed'}
```

Fig.3.2 Types of Attacks

3.1.2 Types of Attacks in Dataset

- Benign
The normal behaviour of data is called Benign here, also the literal meaning of benign is unharmed in effect.
- DoS GoldenEye
Denial of service attack using Goldeneye tool, Goldeneye is a python implemented tool used for the purpose of attack. It persists socket connection via caching until it consumes all available sockets on the HTTP/S server.
- DoS Hulk
Hulk is a tool used to attack the web servers by creating unique and obscure traffic. It is also python implemented tool to generate denial of service attack.
- DoS Slowhttpstest
Slowhttpstest is used to generate the denial of service attack at application layer. It prolongs the HTTP connection. It is used to test the web server's vulnerabilities towards DoS attack.
- DoS slowloris
Slowloris is a tool to stimulate the denial of service attack. It enables a single machine to take down the whole web server. It works on very low bandwidth and it is also simple in nature. The way of attack is by sending HTTP requests and keep them incomplete

always. If slowloris attacks go undetected then they can last for a long period of time and can harm the server for a long.

- Heartbleed

A block of memory of server up to the size of 64 kb is retrieved by the attacker, this allows the attacker to send malicious information to the server during the connection and there is no limit on how much malicious data can be sent over there. It thus allows the attacker to steal the sensitive data from server's memory.

3.2 DATA PREPROCESSING

Data preprocessing is the first step towards the designing of an IDS. The raw data can't be used directly in the model for training and testing, first we need to preprocess the data. Most of the time, the real world data or the raw data is not complete, not consistent, or is full of so many errors. To transform the data into an understandable form, data pre-processing is done. Data pre-processing is the method to resolve the unwanted issues before applying algorithms. Some of the data pre-processing steps involves cleaning, integration, transformation, reduction, etc.

Further from 'Label' column we take the count of different types of attacks, just to take the knowledge of dataset and attacks present in it.

```
There are 439683 BENIGN attack.  
There are 10293 DoS GoldenEye attack.  
There are 230124 DoS Hulk attack.  
There are 5499 DoS Slowhttptest attack.  
There are 5796 DoS slowloris attack.  
There are 11 Heartbleed attack.
```

Fig.3.3 Number of Attacks

- In order to take care of any infinite values in dataset, those values are replaced by 'Nan' i.e. not a number and further null values are dropped from data.
- Next step of data preprocessing is to convert the non-numeric values of features into the numeric values. This process is done so that the mathematical tools can be applied to the dataset. In our dataset, column describing the names of attacks is the only column

with non-numeric values. Therefore the attacks ‘Benign’, ‘DoS GoldenEye’, ‘DoS Hulk’, ‘DoS Slowhttpstest, DoS slowloris’ and ‘Heartbleed’ are given values 0, 1, 2, 3, 4, 5 respectively. This change in the dataset can be seen in the figure 3.4

Init_Win_bytes_backward	act_data_pkt_fwd	min_seg_size_forward	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
-1	1	20	0.0	0.0	0	0	0.0	0.0	0	0	0
229	0	32	0.0	0.0	0	0	0.0	0.0	0	0	0
-1	1	20	0.0	0.0	0	0	0.0	0.0	0	0	0
237	1	32	821.0	0.0	821	821	6506213.0	0.0	6506213	6506213	1
229	0	32	0.0	0.0	0	0	0.0	0.0	0	0	0
-1	3	20	0.0	0.0	0	0	0.0	0.0	0	0	0
-1	1	20	0.0	0.0	0	0	0.0	0.0	0	0	0
0	0	32	0.0	0.0	0	0	0.0	0.0	0	0	0
-1	5	20	0.0	0.0	0	0	0.0	0.0	0	0	0
-1	3	20	0.0	0.0	0	0	0.0	0.0	0	0	0

Fig.3.4 Data with all Numeric Values

- The next step in our approach is to divide the dataset into training, validation and testing data. Training set trains the model about the detection of attacks, validation set is there to validate the training and next we have test set in order to check the accuracy of our IDS model. We have distributed our data in train, valid and test set as 80%, 10% and 10% of the whole dataset respectively.

Win_bytes_forward	Init_Win_bytes_backward	act_data_pkt_fwd	min_seg_size_forward	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
-1	-1	1	20	0.0	0.0	0	0	0.0	0.0	0	0	0
294	71	0	32	0.0	0.0	0	0	0.0	0.0	0	0	0
283	229	0	32	0.0	0.0	0	0	0.0	0.0	0	0	0
-1	-1	1	32	0.0	0.0	0	0	0.0	0.0	0	0	0
554	243	0	32	0.0	0.0	0	0	0.0	0.0	0	0	0
791	253	0	20	0.0	0.0	0	0	0.0	0.0	0	0	0
-1	-1	3	20	31045.0	0.0	31045	31045	56200000.0	0.0	56200000	56200000	0
422	1311	0	32	0.0	0.0	0	0	0.0	0.0	0	0	0
-1	-1	1	20	0.0	0.0	0	0	0.0	0.0	0	0	0
1277	0	3	20	0.0	0.0	0	0	0.0	0.0	0	0	0

Fig.3.5 Training Set

Subflow Bwd Bytes	Init_win_bytes_forward	Init_win_bytes_backward	act_data_pkt_fwd	min_seg_size_forward	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
282	-1	-1	1	32	0.0	0.0	0	0	0.0	0.0	0	0	0
6	219	255	0	20	0.0	0.0	0	0	0.0	0.0	0	0	0
132	-1	-1	1	32	0.0	0.0	0	0	0.0	0.0	0	0	0
250	-1	-1	1	20	0.0	0.0	0	0	0.0	0.0	0	0	0
120	-1	-1	1	40	0.0	0.0	0	0	0.0	0.0	0	0	0
232	-1	-1	1	20	0.0	0.0	0	0	0.0	0.0	0	0	0
318	-1	-1	1	32	0.0	0.0	0	0	0.0	0.0	0	0	0
0	3	229	0	32	0.0	0.0	0	0	0.0	0.0	0	0	0
102	-1	-1	1	32	0.0	0.0	0	0	0.0	0.0	0	0	0
176	-1	-1	1	32	0.0	0.0	0	0	0.0	0.0	0	0	0

Fig.3.6 Validation Set

Init_win_bytes_forward	Init_win_bytes_backward	act_data_pkt_fwd	min_seg_size_forward	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
-1	-1	1	20	0.0	0.0	0	0	0.0	0.0	0	0	0
283	229	0	32	0.0	0.0	0	0	0.0	0.0	0	0	0
-1	-1	1	20	0.0	0.0	0	0	0.0	0.0	0	0	0
29200	237	1	32	821.0	0.0	821	821	6506213.0	0.0	6506213	6506213	1
3	229	0	32	0.0	0.0	0	0	0.0	0.0	0	0	0
-1	-1	3	20	0.0	0.0	0	0	0.0	0.0	0	0	0
-1	-1	1	20	0.0	0.0	0	0	0.0	0.0	0	0	0
1006	0	0	32	0.0	0.0	0	0	0.0	0.0	0	0	0
-1	-1	5	20	0.0	0.0	0	0	0.0	0.0	0	0	0
-1	-1	3	20	0.0	0.0	0	0	0.0	0.0	0	0	0

Fig.3.7 Test Set

After that we use normalization process in order to normalize the data, so that whole data can be made co-dimensional. As we have dataset in which the values are out of bound in comparison to other values. Therefore the data values of different range can be brought in optimal range by the normalization process so that processing can become easy. We stick to the commonly used min-max normalization method to normalize the data. The equation for min-max normalization can be given as

$$X_{sc} = \frac{X - Xmin}{Xmax - Xmin}$$

By using this equation all the values in the dataset can be brought in an optimal range of computing and hence results will be more accurate. After normalizing and preprocessing the dataset we calculate the dimensions of training data, validation data and test data.

- Training data have 553125 rows and 78 columns.
- Validation data have 69141 rows and 78 columns.
- Test data have 69140 rows and 78 columns.

3.3 FEATURE EXTRACTION

The next step for building an IDS is feature extraction from the dataset as dataset contains some irrelevant features that do not contribute in our concern or intrusion detection. In order to retrieve the important features from the dataset, we implement vanilla autoencoder. In our ‘Introduction’ chapter, we have discussed in detail how autoencoders work to extract the important features from the dataset.

3.3.1 Vanilla Autoencoder

Vanilla autoencoder is a simple 3 layered neural network with 1 hidden layer and that hidden layer is the bottleneck in vanilla autoencoder. Vanilla autoencoder implements dense neural network. As we have discussed, a neural network needs activation function for the purpose of learning and also to have non-linearity in the output.

- Activation function used for the encoding purpose is Relu function i.e. rectified linear unit activation function. Relu is a frequently used activation function. Relu function is half rectified function i.e. when input is negative the output will be 0. The equation of Relu can be given as

$$y = \max(0, x)$$

Relu function can be seen as in figure 3.8

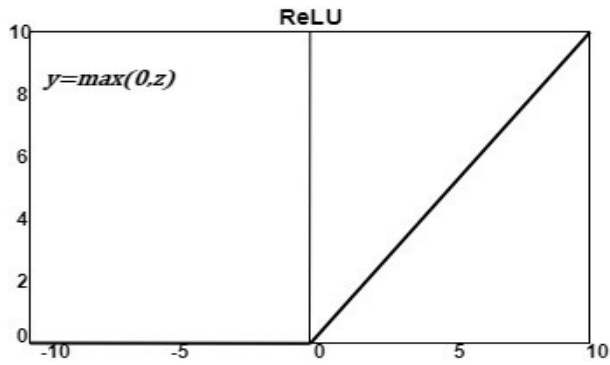


Fig.3.8 ReLU Function

After encoding for the decoding purpose also, we shall use Relu as our activation function. But at last while mapping the output of a hidden layer to the main output layer the activation function used is sigmoid function. We use sigmoid function so that after extraction of features the data values which were negative should not get affected. The equation of sigmoid function is given as follows

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

Sigmoid Function can be seen as in figure 3.9

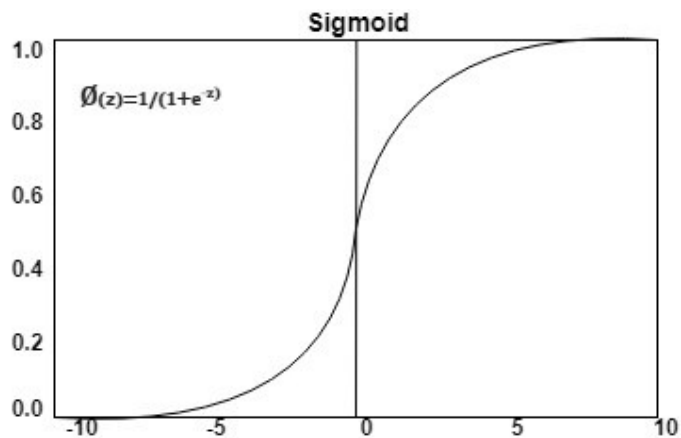


Fig.3.9 Sigmoid Function

The autoencoder constructed had 78 features as input, then input is passed to the autoencoder and at first in the encoder part 64 features were drawn from the input with relu as activation function, relu will also be the activation in the subsequent layers of autoencoder till the second last layer. In the next layer there will be 32 features and in the bottleneck layer, there will be 16 features. At bottleneck layer important and

relevant features are extracted, then decoder starts mapping the output from bottleneck back to original dimensions first to 32, then 64. From here in order to map the features to the final output, the activation function used is sigmoid in order to take care of negative values in dataset. The summary of autoencoder can be seen in the figure 3.10.

```
Model: "model"
```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 78)]	0
dense (Dense)	(None, 64)	5056
dense_1 (Dense)	(None, 32)	2080
dense_2 (Dense)	(None, 16)	528
dense_3 (Dense)	(None, 32)	544
dense_4 (Dense)	(None, 64)	2112
dense_5 (Dense)	(None, 78)	5070
=====		
Total params: 15,390		
Trainable params: 15,390		
Non-trainable params: 0		

Fig.3.10 Autoencoder Summary

It has a total of 15390 parameters and all parameters are trainable.

- Loss function

Loss function can be stated as the difference between the expected output and the original output. Loss is calculated so that learning process and back propagation can be carried out accordingly. We use Binary Cross-entropy as our loss function in the autoencoder. Binary cross-entropy is a log loss function. The equation of binary cross-entropy loss function can be given as

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^n y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))$$

P_y is the probability of classifying in one class and $1-P_y$ is the probability of classifying in other class. The log loss function is shown in figure 3.11.

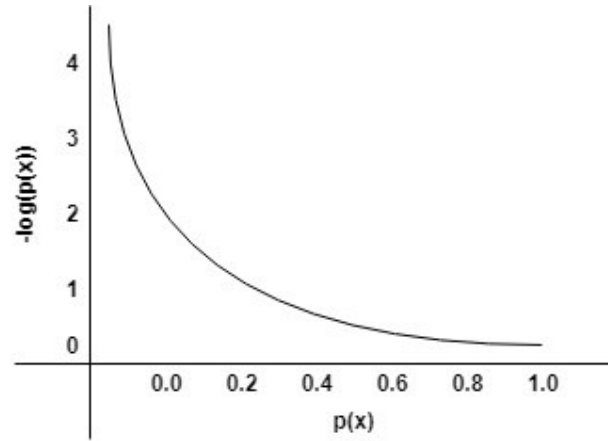


Fig.3.11 Binary Cross Entropy Function

- Optimizer

Optimizer is the function used to minimize the calculated error, the error is minimized by adjusting the weights and bias in a neural network. Weights are adjusted by back propagation and back propagation is controlled by optimizer function. The optimizer used by us is Adadelata. Adadelata means adaptive delta where delta means the difference between the current weight and the newly updated weight. Adadelata focuses on learning rate component. It is an update to Adagrad optimizer as it controls the rate of adagrad by using a window of size w . The equation of Adadelata is given as

$$\theta_{t+1} = \theta_t + \Delta\theta_t$$

$$\Delta\theta = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g_t]} \cdot g_t$$

The architecture of our autoencoder is shown in the figure 3.12

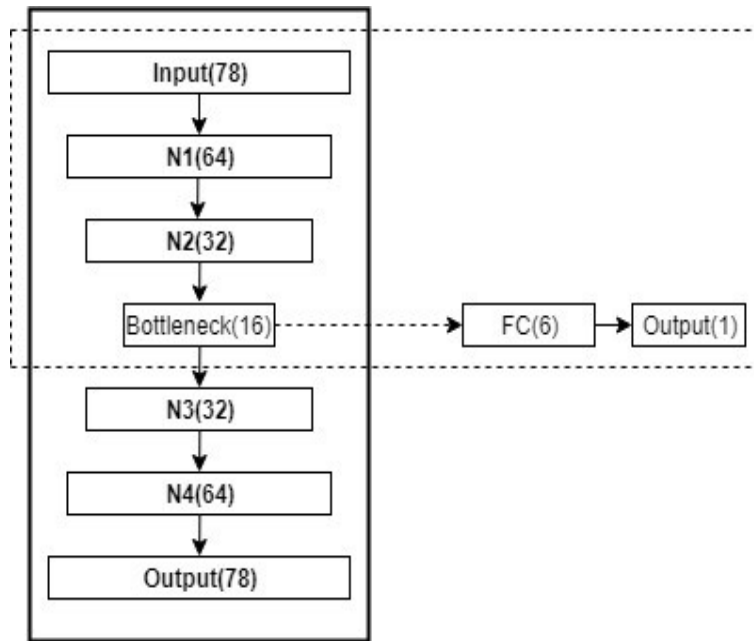


Fig.3.12 Architecture of Autoencoder

3.4 MODEL GENERATION

For the purpose of classification, we implement a dense neural network of 1 layer. Dense neural network will classify the attacks into their respective categories. A fully connected artificial neural network is known as dense neural network. All the nodes of a layer are connected to all the nodes of subsequent layer.

The summary of dense neural network model is shown in the figure 3.13

```

Model: "sequential_1"
-----
Layer (type)                Output Shape                Param #
-----
dense_7 (Dense)             (None, 6)                   102
-----
Total params: 102
Trainable params: 102
Non-trainable params: 0
  
```

Fig.3.13 Summary of Dense Neural Network

It has a total of 102 parameters and all are trainable.

- Activation Function

Softmax function is used as activation function. Softmax function converts the numbers into the probabilities values ranging from 0 to 1. The equation of softmax function can be given as

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

The values become in optimal range and thus makes the data co-dimensional. Softmax function can be shown in figure 3.14

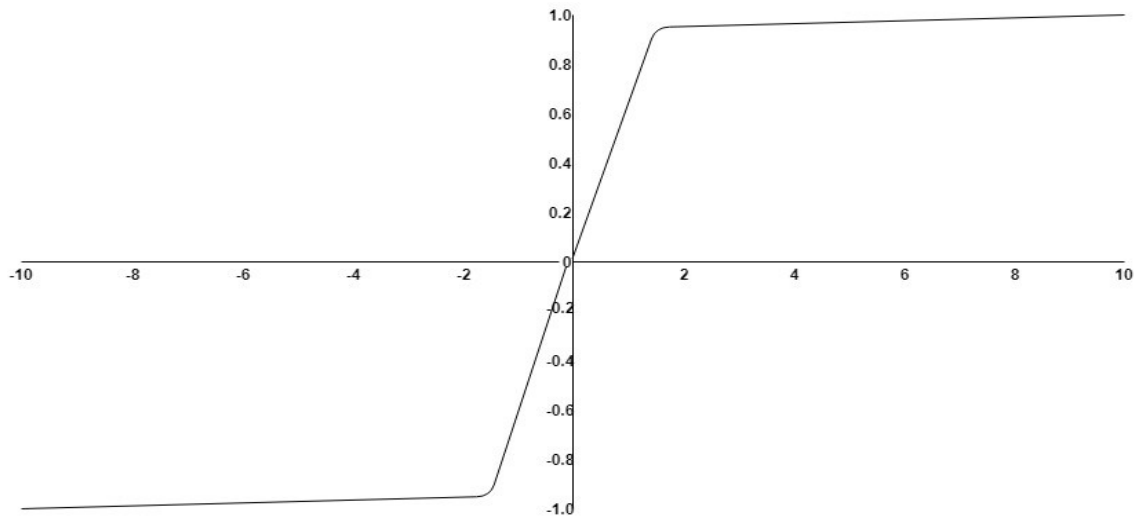


Fig.3.14 Softmax Function

- Loss Function

Loss function used is sparse categorical cross entropy function. This function does not require the whole vector as input rather it works with only the number or digits. This is also a log loss function.

- Optimizer

For the backpropagation and learning purpose, the optimizer used is RMSPROP. Rmsprop increases the learning rate of a neural network. The equation of Rmsprop can be given as

$$V_t = \rho v_{t-1} + (1 - \rho) * g_t^2$$

$$\Delta w_t = -\frac{\eta}{\sqrt{v_t + \epsilon}} * g_t$$

$$w_{t+1} = w_t + \Delta w_t$$

Thus with the implementation of dense neural network our IDS model is also ready and the model creation can be seen in the flowchart depicted in figure 3.15.

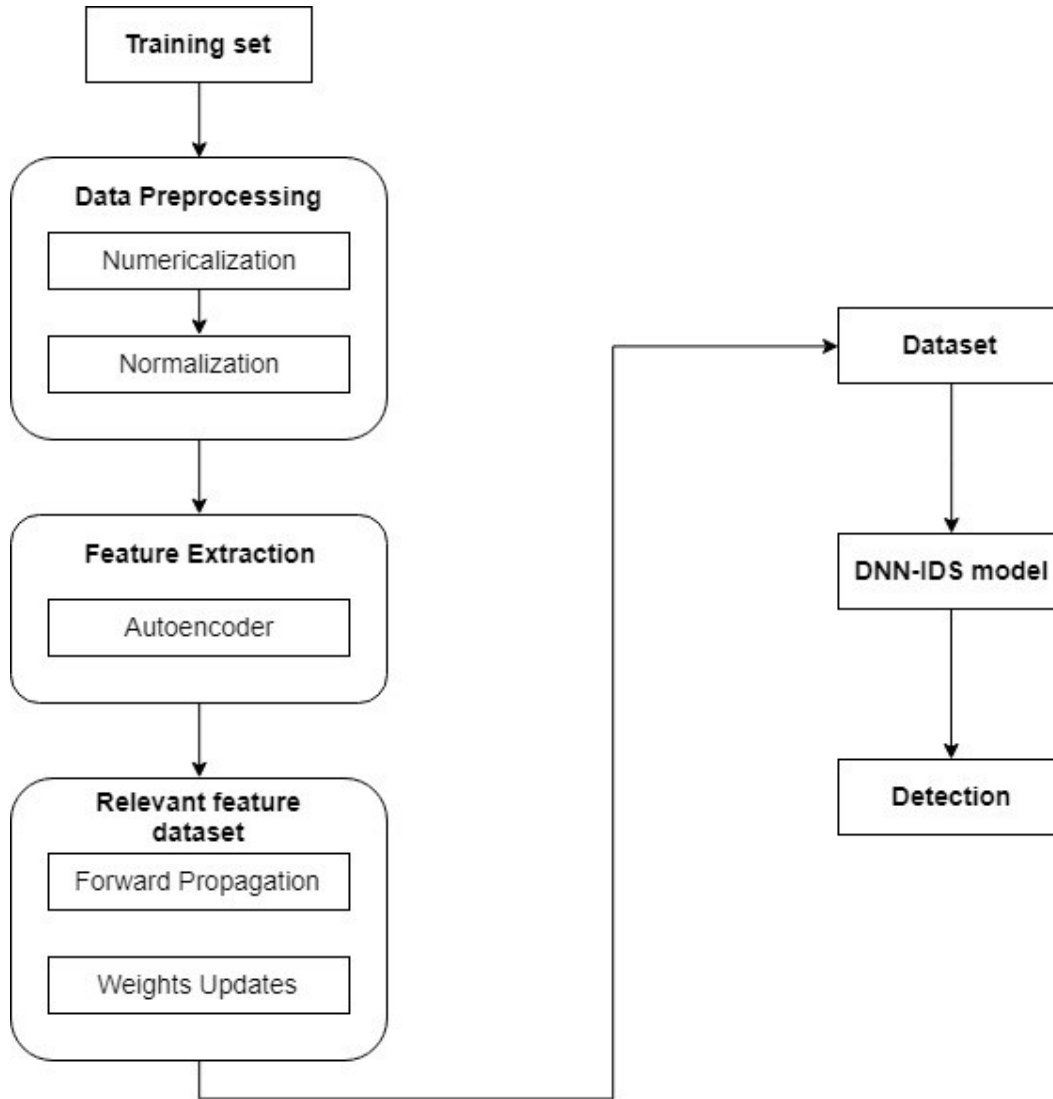


Fig.3.15 Block Diagram of Model

3.5 CNN MODEL

We also applied convolutional neural network for the classification purpose of attacks. We build a CNN model which can be used as IDS on our dataset. We have discussed about CNN model in our Artificial Neural Network. The working process of CNN model can be seen from there. The summary of CNN model can be seen in figure 3.16.

```

Model: "sequential_9"
-----
Layer (type)                Output Shape                Param #
-----
conv2d_2 (Conv2D)           (None, 4, 4, 32)           160
-----
conv2d_3 (Conv2D)           (None, 3, 3, 16)           2064
-----
flatten_1 (Flatten)         (None, 144)                 0
-----
dense_125 (Dense)           (None, 6)                   870
-----
Total params: 3,094
Trainable params: 3,094
Non-trainable params: 0

```

Fig.3.16 Summary of CNN Model

It has a total of 3094 parameters and all are trainable.

- The activation function used is Softmax.
- The optimizer used is Rmsprop.
- The loss function used is Sparse Categorical Entropy.

We have already discussed about these functions in autoencoder.

CHAPTER 4

RESULTS

The dataset has been applied to the models and we have calculated the results of both the models on the basis of following equations.

Accuracy can be calculated by the given equation

$$AC = \frac{TP + TN}{TP + FN + FP + TN}$$

Detection rate can be given by the equation:

$$DR = \frac{TP}{TP + FN}$$

False alarm rate can be given by:

$$FAR = \frac{FP}{FP + TN}$$

We applied the DNN-Autoencoder and CNN models to predict the attack in the given dataset. The values of 3 parameters for both the models are given in table 1.

Table1. Accuracy of DNN-Autoencoder and CNN

	ACCURACY	DETECTION RATE	FALSE ALARM RATE
DNN-Autoencoder	94.83%	94.83%	5.17%
CNN	67.38%	71.05%	6.32%

The accuracy shown by DNN-Autoencoder is higher than the normal Convolutional neural network. We can see the comparison of accuracies, detection rate and false alarm rate of both the methods in the charts in following figures.

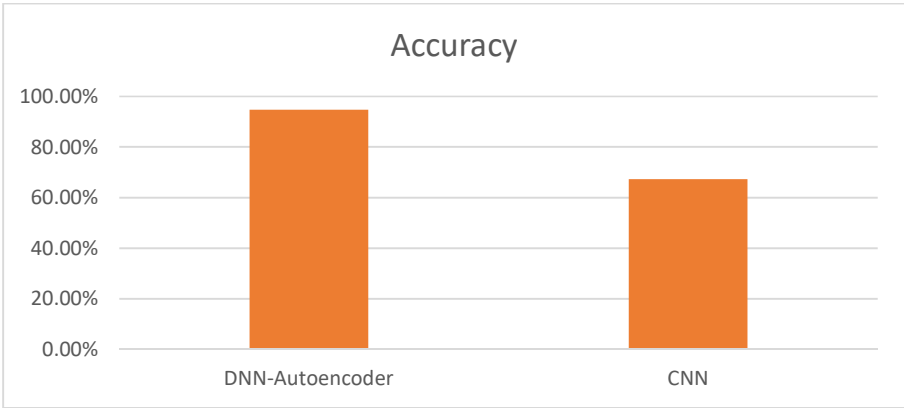


Fig.4.1 Accuracy Comparison

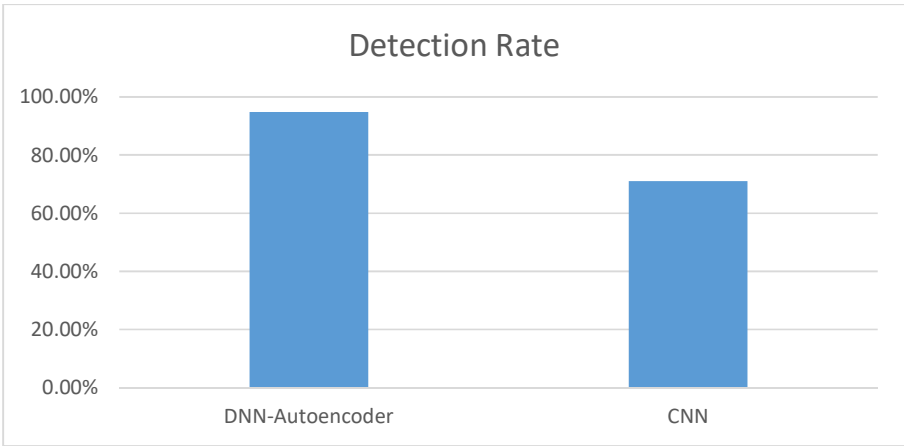


Fig.4.2 Detection Rate Comparison

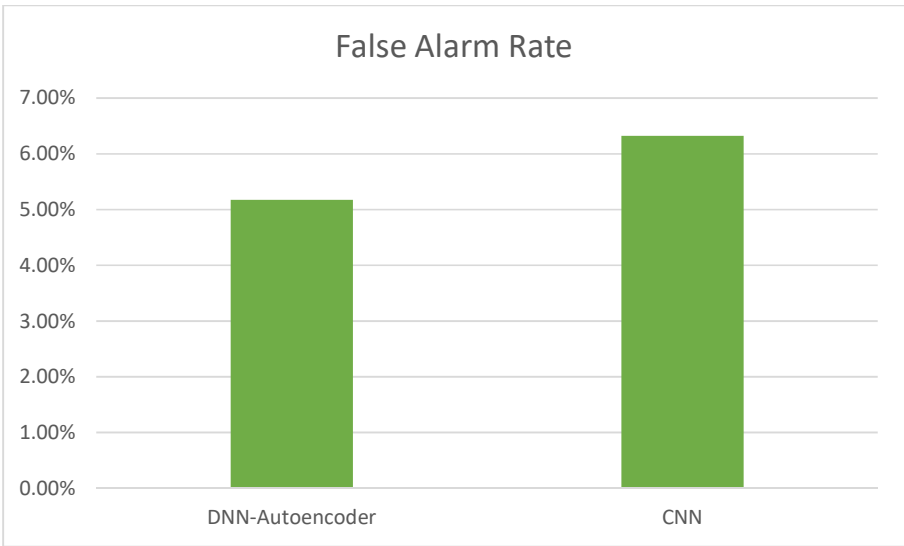


Fig.4.3 False Alarm Rate Comparison

From results it is clear that dense neural network with autoencoder gives better accuracy and detection rate. False alarm rate is low in case of dense neural network.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Intrusion detection system implemented in this work shows a great accuracy towards the detection of malicious data or attack data. It has high modelling ability towards intrusion detection. The model worked very well with the new dataset Intrusion Detection Evaluation Dataset (CICIDS2017). For the purpose of feature extraction we implement autoencoder which itself extracts the important features from the dataset. The experimental results shows that the model works better with new data which matches with the attacks of current generation. Different IDS shows different results. All have their own intrusion responses under different scenarios. Our model is quite comprehensive in terms of detection rate, false alarm rate and accuracy.

The accuracy is good but still there is scope of improvement in accuracy, we shall try to improve the accuracy by improving our dense neural network model. We shall try to implement the different neural network models with autoencoders in order to check their accuracy. We shall also try to improve the existing model by using different activation function, loss function and optimizer. Detection time must also be taken care of, so that when implemented in real network the model should work in real time. CNN model can be improved further by using different features extraction techniques.

Comparison of both the models show that our model is better in terms of all the aspects i.e. in terms of accuracy, detection rate and false alarm rate. CNN can also work better if feature extraction methods are improved i.e. better feature extraction can be implemented.

REFERENCES

- [1] K. Wu, Z. Chen and W. Li, "A Novel Intrusion Detection Model for a Massive Network Using Convolutional Neural Networks," in *IEEE Access*, vol. 6, pp. 50850-50859, 2018.
- [2] S. M. Kasongo and Y. Sun, "A Deep Learning Method With Filter Based Feature Engineering for Wireless Intrusion Detection System," in *IEEE Access*, vol. 7, pp. 38597-38607, 2019.
- [3] Y. Xiao, C. Xing, T. Zhang and Z. Zhao, "An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks," in *IEEE Access*, vol. 7, pp. 42210-42219, 2019.
- [4] C. Xu, J. Shen, X. Du and F. Zhang, "An Intrusion Detection System Using a Deep Neural Network With Gated Recurrent Units," in *IEEE Access*, vol. 6, pp. 48697-48707, 2018.
- [5] M. H. Ali, B. A. D. Al Mohammed, A. Ismail and M. F. Zolkipli, "A New Intrusion Detection System Based on Fast Learning Network and Particle Swarm Optimization," in *IEEE Access*, vol. 6, pp. 20255-20261, 2018.
- [6] X. Cheng, B. Liu, K. Li and J. Yan, "Intrusion Detection System Based on KNN-MARS," *2009 WRI World Congress on Software Engineering*, Xiamen, 2009, pp. 392-396.
- [7] P. Tao, Z. Sun and Z. Sun, "An Improved Intrusion Detection Algorithm Based on GA and SVM," in *IEEE Access*, vol. 6, pp. 13624-13631, 2018.
- [8] W. Li and Q. Li, "Using Naive Bayes with AdaBoost to Enhance Network Anomaly Intrusion Detection," *2010 Third International Conference on Intelligent Networks and Intelligent Systems*, Shenyang, 2010, pp. 486-489.
- [9] Lin, Z., Shi, Y., & Xue, Z.L. (2018). IDSGAN: Generative Adversarial Networks for Attack Generation against Intrusion Detection. *ArXiv*, [abs/1809.02077](https://arxiv.org/abs/1809.02077).
- [10] A. Borkar, A. Donode and A. Kumari, "A survey on Intrusion Detection System (IDS) and Internal Intrusion Detection and protection system (IIDPS)," *2017 International Conference on Inventive Computing and Informatics (ICICI)*, Coimbatore, 2017, pp. 949-953.
- [11] F. Sabahi and A. Movaghar, "Intrusion Detection: A Survey," *2008 Third International Conference on Systems and Networks Communications*, Sliema, 2008, pp. 23-26.

- [12] S. Naseer *et al.*, "Enhanced Network Anomaly Detection Based on Deep Neural Networks," in *IEEE Access*, vol. 6, pp. 48231-48246, 2018.
- [13] S. Teng, N. Wu, H. Zhu, L. Teng and W. Zhang, "SVM-DT-based adaptive and collaborative intrusion detection," in *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 108-118, Jan. 2018.
- [14] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," in *IEEE Access*, vol. 7, pp. 41525-41550, 2019.
- [15] S. M. Almansob and S. S. Lomte, "Addressing challenges for intrusion detection system using naive Bayes and PCA algorithm," *2017 2nd International Conference for Convergence in Technology (I2CT)*, Mumbai, 2017, pp. 565-568.
- [16] C. Yin, Y. Zhu, J. Fei and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," in *IEEE Access*, vol. 5, pp. 21954-21961, 2017.
- [17] F. A. Khan, A. Gumaei, A. Derhab and A. Hussain, "A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection," in *IEEE Access*, vol. 7, pp. 30373-30385, 2019.
- [18] A. S. Sadiq *et al.*, "An Efficient IDS Using Hybrid Magnetic Swarm Optimization in WANETs," in *IEEE Access*, vol. 6, pp. 29041-29053, 2018.
- [19] W. Wang *et al.*, "HAST-IDS: Learning Hierarchical Spatial-Temporal Features Using Deep Neural Networks to Improve Intrusion Detection," in *IEEE Access*, vol. 6, pp. 1792-1806, 2018.
- [20] M. Latah and L. Toker, "Towards an efficient anomaly-based intrusion detection for software-defined networks," in *IET Networks*, vol. 7, no. 6, pp. 453-459, 11 2018.
- [21] A. K. Saxena, S. Sinha and P. Shukla, "General study of intrusion detection system and survey of agent based intrusion detection system," *2017 International Conference on Computing, Communication and Automation (ICCCA)*, Greater Noida, 2017, pp. 471-421.
- [22] C. C. Tan and C. Eswaran, "Performance Comparison of Three Types of Autoencoder Neural Networks," *2008 Second Asia International Conference on Modelling & Simulation (AMS)*, Kuala Lumpur, 2008, pp. 213-218.
- [23] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," in *Proc. AAAI Fall Symp. Relevance*, 1994, pp. 140_144.

[24] A. Sharma, I. Manzoor, and N. Kumar, "A feature reduced intrusion detection system using ANN classifier," *Expert Syst. Appl.*, vol. 88, pp. 249_257, Dec. 2017.

[25]

https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks

[26] <https://www.unb.ca/cic/datasets/ids-2017.html>