ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FLOW-BASED ANOMALY INTRUSION DETECTION SYSTEM USING MACHINE LEARNING TECHNIQUES

BY:

TEKESTE EPHRAME HORA

JANUARY, 2020 G.C.

ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A

THESIS ON

FLOW-BASED ANOMALY INTRUSION DETECTION SYSTEM USING MACHINE LEARNING TECHNIQUES

BY:

TEKESTE EPHRAME HORA (ID: A/PE16383/10)

ADVISOR:

TILAHUN MELAK(PHD)

A Thesis Submitted to the School of Electrical Engineering and Computing of Adama Science and Technology University in Partial Fulfillment of the Requirements for the Degree of Masters of Science in Computer Science and Engineering (CSE)

JANUARY 2020 G.C

ADAMA, ETHIOPIA

ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FLOW-BASED ANOMALY INTRUSION DETECTION SYSTEM USING MACHINE LEARNING TECHNIQUES

BY:

TEKESTE EPHRAME HORA

Name and Signature of Members of the Examining Board

_____	_____	_____
Name of Chairperson	Signature	Date
_____	_____	_____
Name of Advisor	Signature	Date
_____	_____	_____
Name of Internal Examiner	Signature	Date
_____	_____	_____
Name of External Examiner	Signature	Date

Declaration

I declare that this piece of work is my own and all sources of materials used for this thesis work have been accordingly acknowledged. The thesis has been submitted in partial fulfillment of the requirements for the degree of Master of Science at Adama Science and Technology University. I truly declare that this thesis work is not submitted to any other institution anywhere for the award of any academic degree, diploma, or certificate. With accurate acknowledgment of the source, users are free to use this thesis without special permission.

Name: Tekeste Ephrame Signature: _____ Date: _____

As a thesis research advisor, I hereby approve that I have read and evaluated this thesis prepared, under my guidance, by Tekeste Ephrame, entitled "Flow-Based Anomaly Intrusion Detection System". I recommend it be submitted as fulfilling the thesis requirement.

Dr. Tilahun Melak _____ _____

Name of advisor Signature Date

Acknowledgments

I would like to thank my advisor Dr.Tilahun Melak for his continuous supervision, encouragement and patience. I would like to give my acknowledgement to my Family who was been with me supporting in any way they can during this research work, also this research wouldn't be where it is without the help of my colleagues, Specially Fakiha Husean, Zewdu Tiumay ,Muluken Yohannies, Tadele Elsai, Biruk Fikadu ,Yishak Aman and Abduraman Abdella

Thank you all!

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Adam	Adaptive Moment Estimation
AE	Autoencoder
AIDS	Anomaly Intrusion Detection System
AUC	Area Under Cover
CIC-IDS	Canadian Institute for Cybersecurity Intrusion Detection System
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSV	Comma-Separated Values
DAE	Denoising Autoencoder
DAG	Direct Acyclic Graph
DBNs	Deep Belief Nets
DNN	Deep Neural Network
DoS	Denial of Services
DPI	Deep Packet Inspection
ELU	Exponential Linear Unit
FS	Feature Selection
FTP	File Transfer Protocol
GDP	Gross Domestic Products
GUP	Graphical Processing Unit
HCNA	Huawei Certified Network Associate
HIDS	Host-Based Intrusion Detection
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IUT	International Telecommunication Union

MAC	Media Access Control
ML	Machine Learning
MSE	Mean Square Error
NIDS	Network Intrusion Detection System
OOPs	Object-Oriented Programming
PCA	Principal Component Analysis
R2L	Remote to user Attack
ROC	Receiver Operating Characteristics
SGD	Stochastic Gradient Descent
SIDS	Signature Based Intrusion Detection Systems
SQL	Sequential Query Language
TF	TensorFlow
U2R	User to Root Atta

Abstract

Cyber-attack increases from time to time. Personal and organization data and various resources found on the network are vulnerable to malicious attacks. An intrusion detection system plays a very important role in protecting computer network security. Various machine learning algorithms have been implemented in network intrusion detection systems. However, these machine algorithms require a long training time, have high computational complexity, have a high false-positive rate, and low accuracy. Also, the current intrusion detection system suffers from the rate of new cyber-attack exponentially increased, the speed of network lines growth, high dimension network traffic flow which is difficult to monitor, analyze, identify and detect the attack.

To overcome the problem mentioned above flow-based anomaly intrusion detection proposed. Flow-based anomaly intrusion detection system uses the information of network interaction, not payload to detect known, unforeseen and unpredictable cyber-attacks. The unsupervised deep learning algorithm, Denoising Autoencoder used to develop flexible and effective intrusion detection systems. CICIDS2017 data set used. CICIDS2017 dataset consists of 85 features, over 3 million records and all features tell the information of network interaction. Dataset is divided into three, that is training dataset, validation dataset, and test dataset. Denoising Autoencoder train on training dataset and develop model, tune parameters to find optimal hyperparameters on validation dataset and performance of the model was evaluated using the test dataset. Any deviation from the built model is considered an attack.

The model learned the abstract and high dimensional feature representation of the CICIDS2017 by passing them into hidden layers. The experimental results show that the performance of the proposed flow-based anomaly intrusion detection system model is significantly better than other network intrusion detection models, which achieved the best detection accuracy 99.98%, ROC 99.98%, short training time, low model loss, low reconstruction error, low false-positive rate, and low overfitting. Through rigorous experimental testing, it is confirmed that the flow-based anomaly intrusion detection system performs well in comparison to the previous studies.

Keywords: Autoencoder, flow-based anomaly, Denoising Autoencoder, Anomaly detection system (ADS), CICIDS2017, machine learning, Deep learning, Neural Network.

CHAPTER ONE
INTRODUCTION

1.1. Background

In the 21st century organizations are interconnected with networks and put a huge amount of data online to stay functional and profitable. Cyber-attack presents a massive threat to the safety and privacy of society [1]. Such a computer system breach can cause a loss of data integrity, access denied for online resources, the leak of confidential data, and take advantage of private resources [2]. According to McAfee and the center for strategic and international studies, nearly one percent of global GDP is lost to cybercrime each year, and its impacting job creation, innovation, and economic growth. Cybercrime drains $600 billion a year from the global economy [3].

Intrusion Detection System (IDS) is an essential method to protect network security from incoming on-line threats [4]. Also securing computer systems from unauthorized use, such as hackers, and protecting them against any kind of abuse from any legitimate access, such as insider threats.

One of the possible solutions for cybersecurity is Flow-Based Anomaly Intrusion Detection. This system monitors and examines network flow instead of packet load [1]. A network flow is a packet from source to destination at a certain time interval with the common property of the packets [1]. Examining network flows rather than the payload solves the curse of dimensionality of data by minimizing network data [5], except some attacks that hide in packet payload may not be detected [6]. Each network flow has a set of features with values, such features are connection duration, amount of packets transfer, flow-id, destination IP, source IP, etc. since the data is gathered from packet header and network flow behavior, privacy is maintained, data encryption is possible and less data is needed to be examined in comparison with the packet payload based intrusion detection technique [1].

Machine learning techniques are essential in building models and detecting intrusions in the anomaly detection system. Machine learning can handle uncertain data. This algorithm is used in both Misuse and Anomaly Intrusion Detection systems. In the Misuse detection system, classify the attack by learning hardcoded signatures stored in the signature list [6]. In an Anomaly detection system build a normal network traffic profile and identify abnormal data from normal one instead of classifying each attack [1,6].

This paper aims to develop a Flow-based Anomaly Intrusion Detection System with Denoising Autoencoder learning approaches dimension reduction. Experiment was performed using the python programming language. The result was compared with an Anomaly Intrusion Detection System that has been developed with the same dataset. System performance will be evaluated by measuring the accuracy of the model, loss of the model, and reconstruction error.

1.2. Motivation

It is difficult or almost impossible to develop an intrusion detection system with 100 percent success rate. Most systems today have a lot of security flaws. Not all kinds of intrusions are known. Also, hackers are figuring out new ways into the networks using various techniques. Quick detection of these attacks will help to identify possible intruders. and limit damage effected. So, developing an efficient and accurate intrusion detection system will help to reduce network security threats

1.3. Statement of the Problem

Intrusion Detection Systems (IDS) are considered one of the basic building blocks of the protection wall against intrusive activities by detecting intrusions before the entire network is affected [7]. An intrusion detection system (IDS) monitors the network traffic looking for suspicious or malicious activities or policy violations that could represent an attack or unauthorized access [8, 9].

Anomaly-Based Network Intrusion Detection System is a type of intrusion detection system (IDS), which builds a model for 'normal' network traffic and uses the model to detect anomalies in the network traffic by detecting deviations from the model [6, 10, 11, 12]. When the Model for the system is available, it is used to classify new events or traffic as anomalous or not [1,12,13]. Anomaly-based approaches succeed in detecting unknown and zero-day attacks which is an advantage over the signature-based approaches [6,10,11]. It aims to protect the confidentiality, integrity, availability of critical networked information systems, and the Network itself [4, 7, 14, 15, and 16].

However Intelligent IDS collects and analyzes information from different areas and deals with a large amount of data, which contains various redundant and irrelevant features and results in

increased overfitting, processing (training) time, and low detection rate [8]. Many problems need to be considered in anomaly-based network intrusion detection system (NIDS), such as the ability to adapt to dynamic network environments, unavailability of labeled data (Labels for normal behavior are usually available, while labels for intrusions are not), false-positive rate [4, 11].

Anomaly-based IDS suffer from a high false-positive rate in such cases of dealing with high dimensional datasets in the training process [6, 10, 12, 17, 18, 19]. The main challenge in anomaly-based systems is how to get high accuracy with a low false-positive rate [17, 20]. Also, this approach suffers from a low detection rate and a high false alarm rate [12, 13].

In the case of intrusion detection, false positives or false negatives lead to bad, sometimes, disastrous results. High false negatives mean high attacking possibility whereas high false positives can take two routes: (1) high false positives run fake alarms, preventing many network services and resources and wasting time and money. High false positives lead to high frequent fake alarms, which in turn, yields an untrustworthy behavior against intrusion alarms leading to an "all-pass" intrusion detection system through frequent alarms cutoff, which opens the possibility for real attacking activity [6].

1.4. Objectives

1.4.1. General Objective

The general objective of this study is to develop a Flow-Based Anomaly Intrusion Detection with machine learning techniques..

1.4.2 Specific objective

The specific objectives of this thesis work are listed as follows:

- ✓ To identify flow based real environment dataset
- ✓ To carry out dataset preprocessing
- ✓ To develop deep neural network architecture
- ✓ To carry out implementation
- ✓ To evaluate the performance of model

- ✓ To carry out concluding the study work and putting recommendation

1.5. Significance of the study

The result of this research can be applied to any network-based systems which their data and service are prone to intrusion, hackers such as online shopping system, commercial Banks, cloud infrastructure, Military, different online service providers, etc. The proposed system will stay on the main network line, learn normal network flow behavior, developed a model, examine incoming network traffic then identify the attack or anomaly that has been known or zero-day attack which is unknown to the model, and finally, the alarm will be generated.

1.6. Scope and Limitations of the study

1.6.1 Scope of the study

This study will mainly focus on developing a flow-based anomaly intrusion detection system. Flow-based anomaly intrusion detection system learns normal network behavior and develop model to identify outliers that is known and have not been seen by the model. The system uses network traffic flow features for analysis and recognition of outliers, but doesn't use a payload of the packet.

1.6.2 Limitation of the study

Even though flow-based anomaly intrusion detection system has a capacity of identifying zero-day and know attack it doesn't identify outliers with in payload packet.

1.7. Organization of the Thesis Work

This section of the document summarizes all the number of chapter used and the overall contents of the thesis work as follows:

Chapter one- Introduction*:* this chapter covers background of the study, statement of the problem, objective of the study, methodology (which is discussed under chapter three), Significance of the study, scope and limitation of the study.

Chapter Two- Literature Review**:** This chapter covers intrusion detection system, intrusion detection system category, feature selection, feature extraction, types of network attacks, application areas of intrusion detection system, deep and neural learning, Autoencoder, Network intrusion detection system, flow-based and packet-based intrusion detection system will be discussed.

Chapter Three- Methodology of the research**:** This chapter includes Dataset, data integration, encoding categorical data, data transformation, data dimensional reduction, missing and duplicate value handling, split dataset, Model development, model evaluation, materials will be discussed.

Chapter Four- Implementation**:** Under this chapter preprocessing and modeling concept will be discussed in detail.

Chapter Five- Discussion**:** under this chapter the result of the experiment will be discussed and compared to the other research work**.**

Chapter six- Conclusion and Recommendation**:** this is the last chapter of study, it contain conclusion of the study, recommendations forwarded by the study.

CHAPTER TWO
REVIEW OF LITERATURES

2.1 Intrusion Detection System

In 21first century organizations interconnected with networks and put a huge amount of data online to stay functional, accessible, and profitable. Cyber-attack presents a massive threat to the safety and privacy of society [1]. Such a computer system breach can cause a loss of data integrity, access denied for online resources, the leak of confidential data, and take advantage of private resources [2].

According to the international telecommunication union (ITU) report of 2019 around 4 billion people using the internet throughout the globe. This report also shows the growth of people using the internet every year is high [3]. Growth of individual internet users from the 2005- 2019 year.

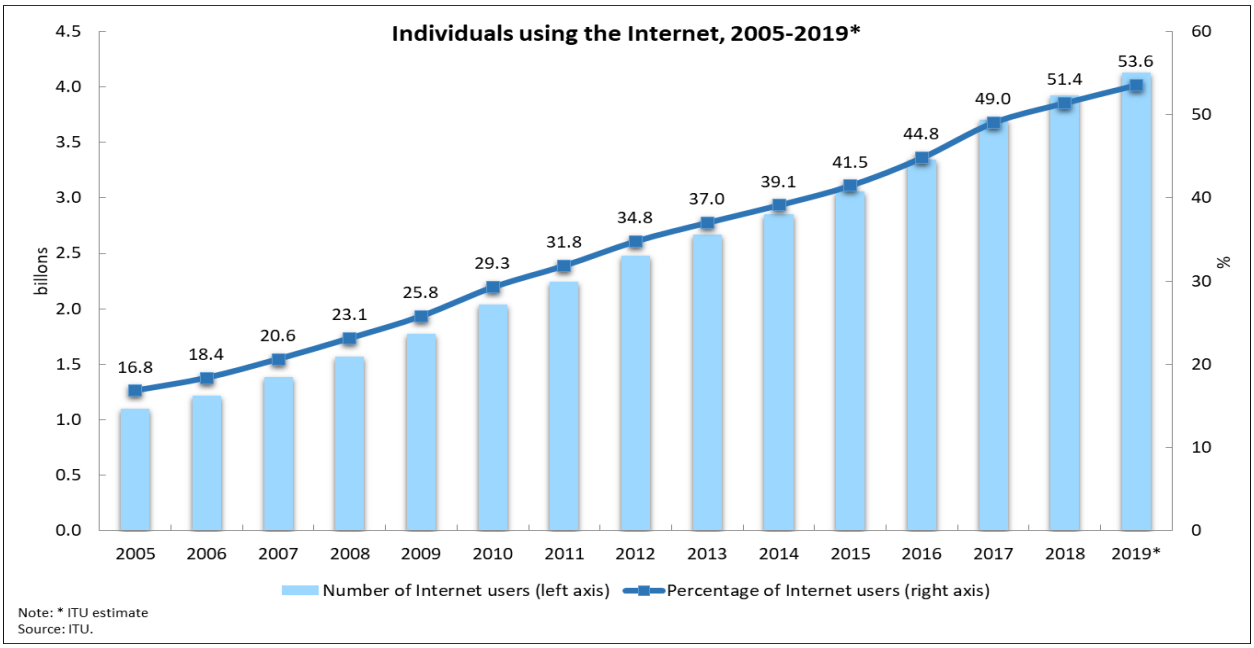


Figure 2.1 Individual Using The Internet: Adapted from [1]

According to an August 2019 report by McAfee [2], cyber-attack increases every year and 2019 is the highest one. Personal and organization data and various resources found with the network are vulnerable to malicious attacks that increase from time to time [2]. Macafee report also identifies the top target sector that has been attacked and the types of breaches that have been reported. Malware, account hijacking, unkonk, vulnerability, unauthorized, targeted attack, code injection

malicious script, denial of service, defacement, and theft is put according to their proportion within the report [1]. The figure shows clearly the growth of threats, attack type, and finally targeted organization. Based on the report we need to develop an intelligent intrusion detection system with high performance, that can keep the cyber-security attack from damaging resources.

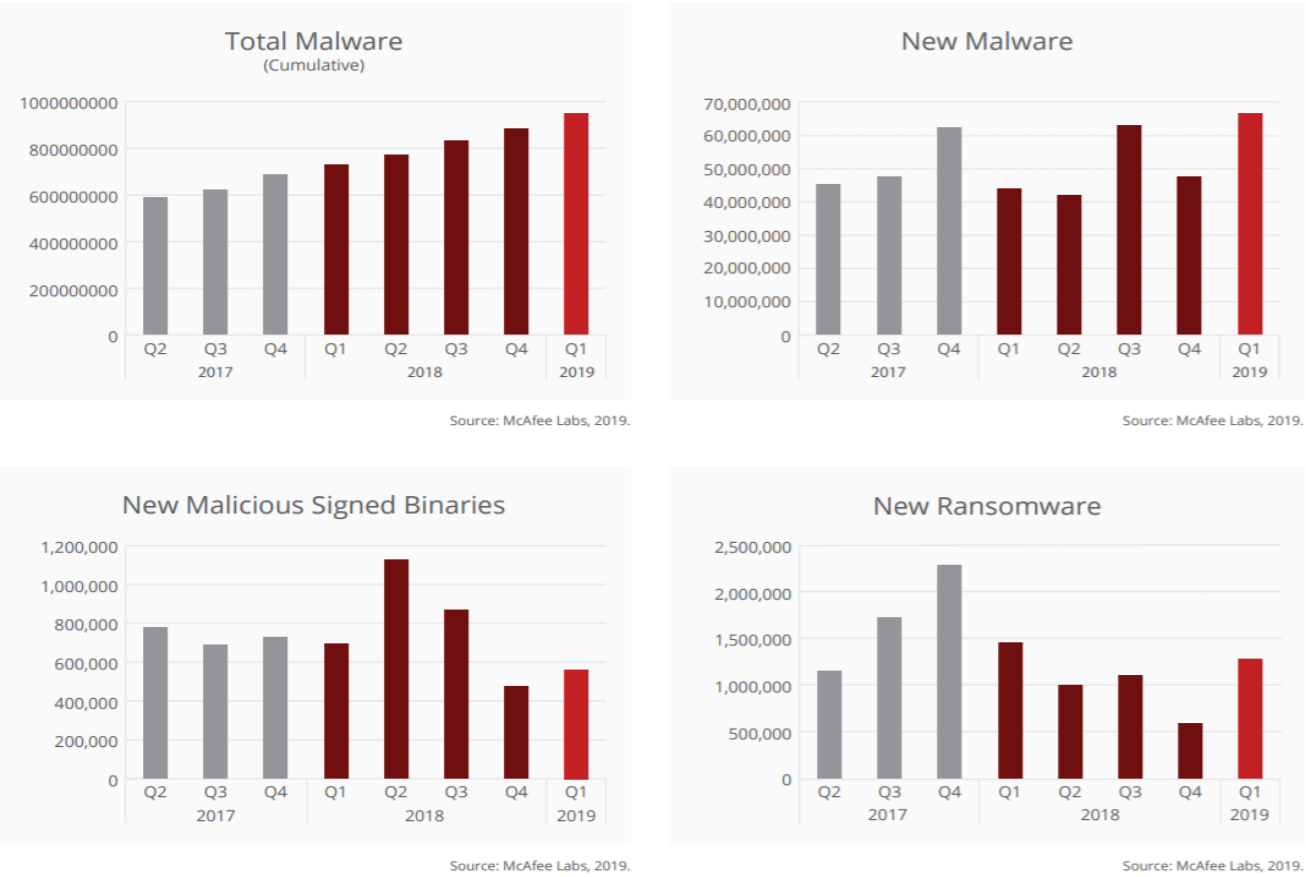


Figure 2.2 Threats growth, Adapted from [2]

According to McAfee and the center for strategic and international studies, nearly one percent of global GDP is lost to cybercrime each year, and its impacting job creation, innovation, and economic growth. Cybercrime drains $600 billion a year from the global economy [3].

Securing computer systems from unauthorized use, such as hackers, and protecting them against any kind of abuse from any legitimate access, such as insider threats. Intrusion Detection System (IDS) is an essential method to protect network security from incoming on-line threats [4]. Intrusion Detection Systems (IDS) are considered one of the basic building blocks of the protection

wall against intrusive activities by detecting it before it hits the network systems [7]. An intrusion detection system (IDS) monitors the network traffic looking for suspicious or malicious activities or policy violations that could represent an attack or unauthorized access [8, 9].

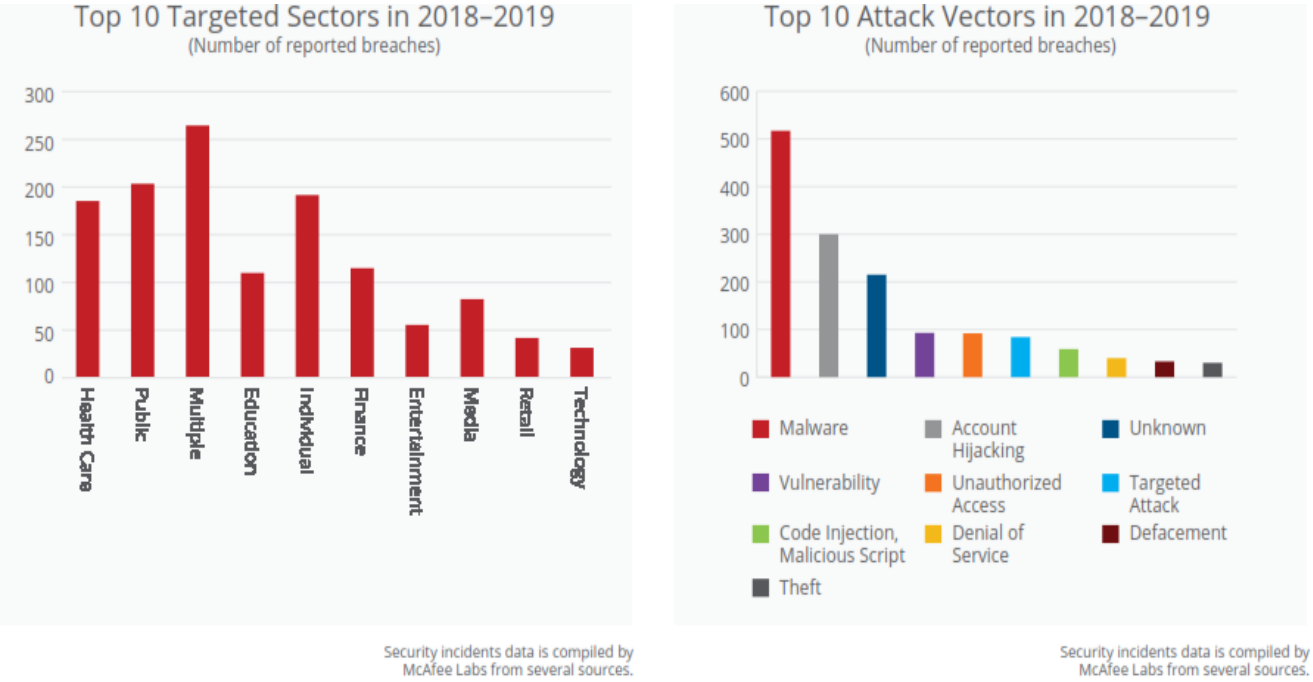


Figure 2.3 Top target sector attacked and top attack vectors, Adapted from [48]

2.2 Intrusion Detection Categories

There are various ways to categorize the intrusion detection system. Based on target intrusion detection system can be categorized as Host-Based Intrusion Detection System (HIDS) and Network-Based Intrusion Detection System (NIDS) and Signature-based and Anomaly-based intrusion detection Using Feature selection for intrusion detection system.

2.2.1 Host-Based Vs Network-Based Intrusion Detection System

The host-based intrusion detection system used data within-host to detect suspicious activities, this data comes from different host activates such as system logs, data recorded by an operating system called audit [30], different application programs, System configuration, application activity, a system command, running processes, file access and modification Security Logs [31]. This type of intrusion detection system is hosted in each host in the network.

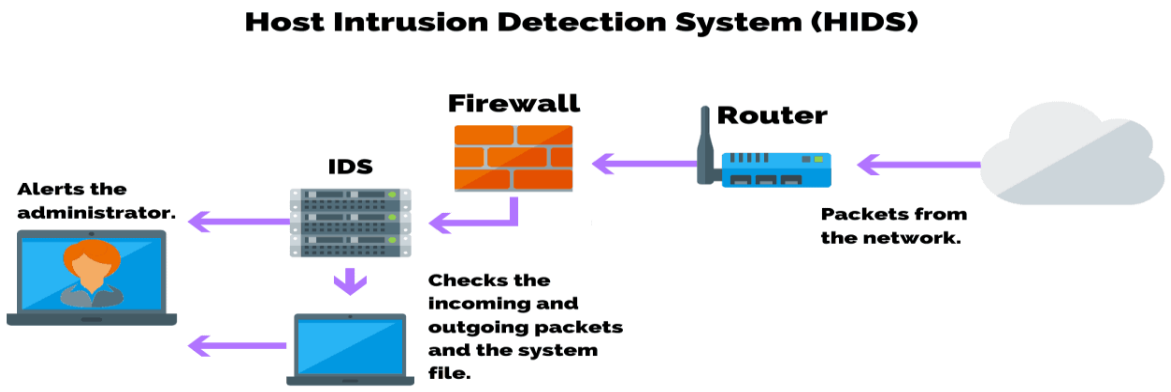


Figure 2.4 Host Intrusion Detection System

Network-Based Intrusion detection system placed on tactical position on the network and try to access the stream of network traffic pass by. NIDS is accessing larger data compared to HIDS and controls the incoming and outgoing flow of network traffic [30, 32].

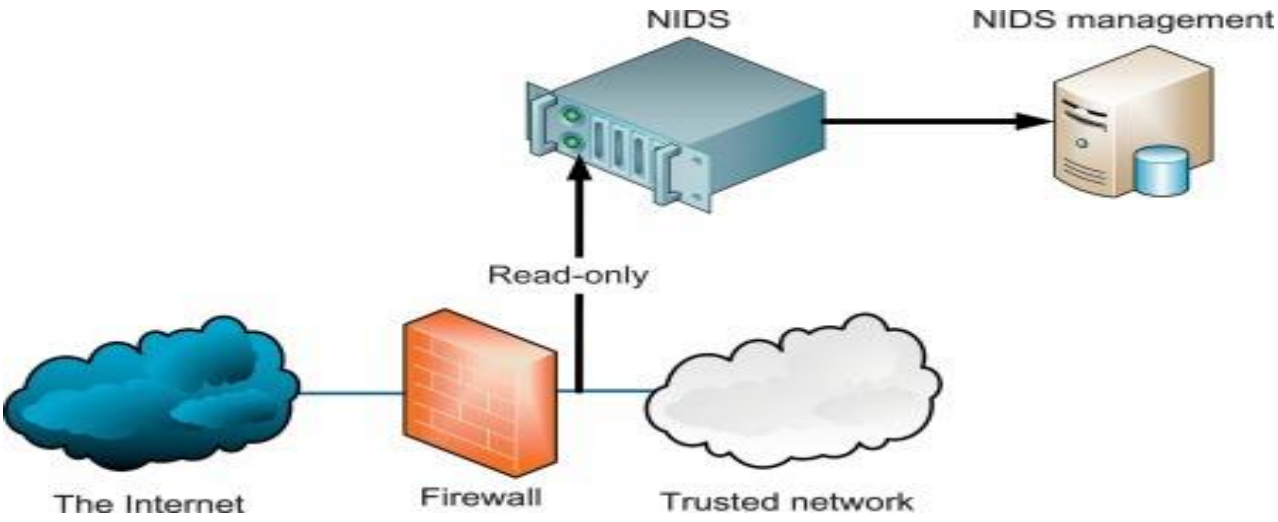


Figure 2.5 Network Intrusion Detection System

2.2.2 Signature-Based (SIDS) and Anomaly Based Intrusion Detection System (AIDS)

Both systems control and examine attacks, suspicious activities, and abnormal behaviors and try to generate an alarm to the administrator when they find threats they are looking for. Signature-Based or Misuse Intrusion Detection System (SIDS): to detected intrusive activities uses a

previously developed database that contains the list of signatures of known intrusion or attack that has been deeply studied and recognized [32]. SIDS can detect known attack efficiently and effectively. SIDS has a low false-positive rate but suffers from high false negatives. Even though the technique is a good reputation in detecting a known attack it fails to detect a zero-day attack or any attack that has a simple variation from normal behavior which results in a high false-negative rate. SIDS IS sensitive to a tiny simple variation and is also known as Misuse intrusion detection [29, 30,]. Every incoming data is matched against the database to detect an attack and an alarm is raised when the match is found [27].

Anomaly-Based Network Intrusion Detection system: detect the unknown attack and identifies abnormal behavior [27,30]. Learn normal network behavior and develop a model, this model is used to detect anomalies within network traffic. Anomaly-Based Network Intrusion Detection System is a type of Intrusion Detection System (IDS), which builds a model for 'normal' network traffic and uses the model to detect anomalies/outliers in the network traffic by detecting deviations from the model [29, 10, 30, 32]. When the Model for the system is available, it is used to classify new events or objects or traffic as anomalous or not [1,12,32]. Anomaly-based approaches succeed in detecting unknown and zero-day attacks which is an advantage over the signature-based approaches [29,30]. It aims to protect the confidentiality, integrity, availability of critical networked information systems, and the Network itself [4, 7, 14, 15, and 16]. AIDS deals with high dimension data which results in long training time, overfitting, processing (training) time, and low detection rate [29].

2.2.3 Advantage and disadvantage of Misuse and Anomaly intrusion detection system

Table 2.1 Advantage and Disadvantage of Signature-based intrusion detection [30, 32]

Advantage	Disadvantage
The high detection rate of know attacks	Fails to detect zero-attack
Low false positive rate.	Database which stored signature of known attack needs a frequent update
Less computing learning time	High false negative

Table 2.2: Advantage and Disadvantage of Anomaly-based intrusion detection [1, 4, 7, 15, 29].

Advantage	Disadvantage
Detect zero-day attack	High false positive rate
Low false negative rate	Long training time
No need to update database	Low detection rate
Detect both known and unknown	Require high computational resource

2.3 Feature selection (FS)

Feature Selection methods will have applied to the dataset to remove redundant and irrelevant features, create a subset of features which represent the dataset which results in short training time and low overfitting [7, 29,33]. During the detection of attacks, feature selection is vital because of the dimensional curse of network traffic. This huge amount of data requires a large amount of storage, computational processing capacity, large training time which reduces intrusion detection performance greatly [29]. Feature selection is data preprocessing tasks and well known and implemented in statistics, pattern recognition, and machine learning [7,29, 33]. FS removes redundant and irrelevant features using feature selection methods but it doesn't produce new features [33], because feature selection efficiency and clarity of model developed increase, decreasing processing and training time which increase in detection performance [33]. FS is also known as data mining [14].

2.3.1 Feature Selection Methods

Feature selection method can be classified into three categories

2.3.1.1 Filter methods

“Filter methods use a statistical model to score and rank the features depending on the intrinsic properties of the data. The features are sorted from the highest to the lowest-ranked. These methods have the advantage of being fast as they are independent of the classifier and scalable with high dimensional data. However, they ignore the correlation between features and the interaction with the classifier. The filter needs to be applied once on the dataset where different classifiers can be evaluated later. A threshold point can be found for cutting down the number of features which should have the lowest ranks [29]. There are different ranking evaluators for feature selection. Some of the evaluators, as referenced in [29] are Info Gain (IG), Gain Ratio (GR), Symmetrical Uncertainty (SU), Relief F (RF), One R (OR), and Chi-Squared (CS)” [29].

2.3.1.2 Wrapper methods

“Wrapper method has been applied in many research areas for features selection by evaluating a subset of features obtained from training and testing a classification model [29]. Many combinations of feature subset created and each subset evaluated finally their performance is compared among themselves then a good subset chosen and used for machine learning algorithms [33]. Whenever a new classifier used this process repeatedly this can be considered as a drawback [29].

2.3.1.3 Embedded or Hybrid methods

Is a combination of the wrapper and filter methods by using their best property combined to reduce high feature dimension space a filter method is used to prepare several candidate subsets then to find the best feature subset a wrapper method is applied. This method provides higher accuracy [35]. Any combination of the two models crates the hybrid method. Such as fuzzy random forest-based feature selection [36], hybrid genetic algorithms [37], and hybrid ant colony optimization [38].

2.4 Feature Extraction Vs Feature Selection Methods

High dimension data is a curse for intrusion detection. It is also problematic for different algorithms because of its high computation cost and memory usage. [39]. to solve the aforementioned problem “there are two-dimensionality reduction techniques know as feature extraction (also known as dimensionality reduction explicitly or feature transformation) and feature selection (FS)” [40]. Feature extraction performs some transformation on the original input data features to generate other features to prepare a suitable representation of high dimensional data.it reduces complexity and create simple representation of data. Feature extraction is widely used in the neural network, deep learning, Autoencoder, and principal component analysis (PCA). It also helps to extract the most relevant features from a set of noisy or redundant data [41]. Feature selection removes irrelevant, redundant and noise data provide viral features only but it doesn’t perform the transformation on input data. Both feature selection and extractions methods are also called dimensional reduction techniques [40].

2.4.1 Advantage of Feature Selection and Extraction

Advantage of both Feature Selection and Feature Extraction Methods listed below [29, 35, 36, 37, 38, 309 and 40].

- ✓ Reduce the dimension of large data
- ✓ Remove redundant, irrelevant, and noise data
- ✓ Reduce training time of learning algorithm
- ✓ Reduce computational resource (CPU, Memory, and Storage)
- ✓ Increase the accuracy and performance of the learning algorithm

2.5 Types of Network Attacks

A computer network is a connection of computers to share resources, such as printers, Applications, storage, processing units. Network attacks can be initialized from an external environment or from inside of the network [42]. The attack in the Wide Area Network is the most challenging one. The intrusion detection system is one of the tools which try to make safe network communication from different network attack types [1, 13,42].

Four major network attack is identified [23,42, and 43]

Denial of Service (DoS): Hacker makes the target machine unavailable or busy by sending many packets which makes any communication difficulties with other machines. Example: Apache, Smurf, Neptune, Ping of death, UDP storm, etc. [23,42, 43].

Probing: Hackers scan a networking device to find out user information, type of service, vulnerability, and weaknesses of the system with hacking tools. Example: Saint, Portsweep, Mscan, Nmap, and Ipsweep [23,42, 43].

Remote to User Attacks(R2L): Intrudes sends some network packets to target machine within the network to exploit privilege of user and damage system of the machine Examples: Dictionary attack, and Password attack [23,42, 43].

User to Root Attacks (U2R): An attacker appeared as a normal user after reaching some safe zone by exploiting vulnerability it makes itself a super user. Such type of intruder identification is most challenging. Example: Buffer overflow [23,42, 43].

Table 2.3: Categories of different types of attacks [44]

Category	Attack Type
U2R	Buffer_overflow, HTTP runnel, loadmodule, Perl, ps, rootkit, SQL attack, xterm
DoS	Aapacha2, back, land, mailbomb, netpune, pod, processtable, smurf, teardrop, udpstorm
R2L	ftp-write, guess password, IMAP, multihop, named, phf, Sendmail, Snmpgetattack, Snmpguess, spy, warezclient, warezmaster, worm, Xlock, Xsnoop
Probing	ipsweep, Mscan, Namp, portsweep, saint, satan

2.6 Application Areas of Intrusion Detection System

Everywhere internet technology exists, the safety of cyber is needed is the corresponding issues. The intrusion detection system is applied to banking operations, online auctions, electronic commerce applications, social networking, and online application/ registration, etc. [30,45].

2.7 Deep learning/ Deep Neural Network

Internet is continuously changing how people live, study, and work [37, 45, 46], in contrary security threats, becoming more serious than ever because hackers continuously changing inventing new techniques to exploit the vulnerability and to make themselves undetectable [37, 45, 47]. Identifying various network attacks is especially the zero-day attack is a big issue. Intrusion Detection System a vital for defending network infrastructure and information security which helps to detect attacks that already accorded or the ongoing one [37, 45]. Intrusion detection thought of as a classification problem weather it is multi-classification (identifying as normal traffic or any one of four attacks categories: Denial of Service (DOS), User to Root (U2R), Probe (Probing) and Root to Local (R2L)) or binary classification (identifying network traffic as normal or anomalous) [37]. Intrusion detection is expected to adapt itself to the dynamic changing of network behavior and attacks [45]. To detect anomalies or threats many supervised and unsupervised techniques have been studied and advised from the discipline of machine learning and data mining to make intrusion detection more reliable [37, 45, 46,47]. "However, most of the traditional machine learning methodologies belong to shallow learning and often emphasize feature engineering and selection" [37]. Machine learning methodologies exhibit some drawbacks: In the real network environment, the presence of high dimensional data make ML weak to solve intrusion classification problem [37]. With the dynamic growth of data sets, multiple classification tasks will lead to decreased accuracy. Shallow learning is couldn't handle intelligent analysis and high dimensional data.

Deep learning is a branch of Machine Learning, can create a better representation of large data which helps to build a better model compared to ML [37]. Deep learning not only creates a better model but also exhibits high accuracy of classification and extraction of features from incomplete data [46].

Deep-learning is a type of machine learning and artificial neural network with a multilayer-structure made up of an input-output layer, which mimics human brain nerves systems and functions like one. Deep means having more than one hidden layer. The deep learning algorithm is not linear like ML as a result can model non-linear and complex relationships. Each layer applies a nonlinear transformation onto its input and creates a statistical model as output from what it learns. The output layer returns the output data. Until the output has reached an acceptable level of accuracy, epochs are continued [45, 46]. Each layer of DNN extracts features and from low-level features, the output layer develops high-level features. DNN has a better ability of modeling complex mapping and feature compression [46].

"Deep learning models are increasingly used to solve complicated big data problems which are often not well addressed by the conventional machine learning algorithm. Deep learning requires less hand-engineered features and expert knowledge. Driven by the emergence of big data and hardware acceleration, the intricacy of data can be extracted with higher and more abstract level representation from raw input features. Some recent work showed that deep learning-based IDS has striking learning capability or outperforms traditional ML-based counterparts" [34].

Deep learning architecture has three parts such as input, hidden, and output layers. Each neuron is connected to the neurons of adjacent layers. Each connection has a weight value. The inputs are multiplied by the respective weights and summed at each unit. The sum then undergoes a transformation based on the activation function [41].

Deep Neural Networks (DNNs) have attracted the attention of the research community and multiple DNN structures including Convolutional neural networks (CNNs) [7], Recurrent Neural networks (LSTM) [9], Deep belief nets (DBNs) and different types of Autoencoder including Sparse, Denoising [30], Convolutional [31], Contractive [32] and variation Autoencoder have been proposed. These DNN structures have been successfully applied to devise state of the art solutions in multiple disciplines. Neural Networks have been around for many decades [29] and have been gaining and losing the favor of the research community. The latest rise of this technology is attributed to Alexnet [29], a Deep Neural Network, which won the ImageNet classification challenge. Alexie achieved top-1 and top-5 error rates of 37.5 % and 17.0% on ImageNet Dataset [7] which were considerably better than the previous state-of-the [45].

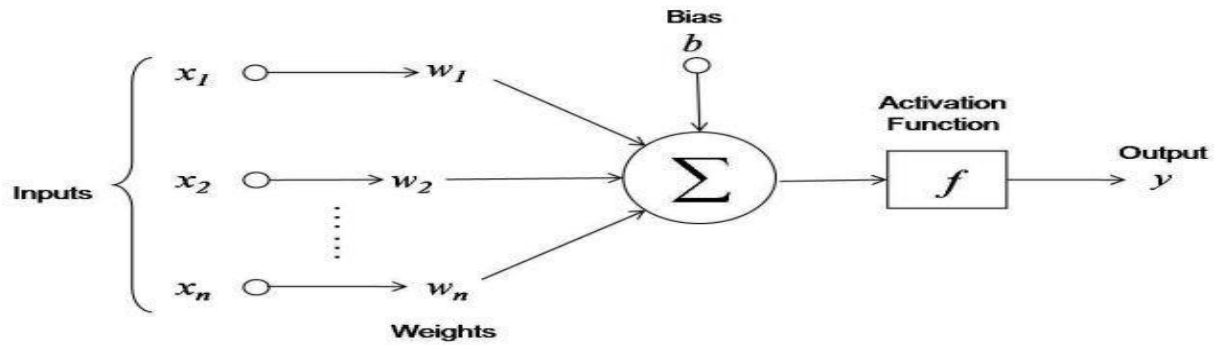


Figure 2.6 Deep Learning

2.8 Autoencoder (AE)

It is a branch of artificial [42] and unsupervised neural networks [50]. AE regenerates output which closes to its original input by studding and tuning parameters [50]. AE is containing three layers so-called input, hidden and output layers [41, 50, 51, and 52]. AE is based on the encoder and decoder principle. Encode is transforming input into lower-dimensional space, but decoder regenerates original data by expanding input pass from the encoder. The hidden layer helps to reduce the dimensionality of input data known as encoding. AE is reducing the dimensionality of the input data to capture the most important features [50]. Input and output layers of AE is always having the same nodes and hidden layers' size is less than the input and output layers' node size. AE is used for the detection of anomalies or feature extraction (unsupervised pre-training) of deep neural networks. In the encoding phase, the input dimension is reduced by using a hidden layer, and the decoding phase regenerates the original input by using encoded information from the hidden layer. Based on the behavior of problem different activation function is used such as sigmoid, than, ELU function [50].

According to [32, 50] there are several Autoencoder types such as Denoising Autoencoder, Sparse Autoencoder, Deep Autoencoder, Contractive Autoencoder, Convolutional Autoencoder, and Variation Autoencoder. Autoencoder has four hyper parameters those are, Code Size: hidden layers' number of neurons, the few neurons in the hidden layer, the high compression. Several layers, number of neurons per layer, and loss function.

Loss function Mean Square Error or binary cross-entropy used to depend on the input value range. Cross-entropy applied for value of input between zero and one otherwise MSE [48,50].

2.9 Related work

2.9.1 Network Intrusion Detection System (NIDS)

Network intrusion detection system monitor, analyze and detect intrusive activates and report to the administrator. According to [55,56] current network intrusion detection system suffers from the following challenges. The rate of new cyber-attack exponentially increased every year, the speed of network lines growth, a large amount of network traffic flow over the network which is difficult to monitor and analyze, identify and detect the attack. Besides storage and computation capacity required is high [55, 56].

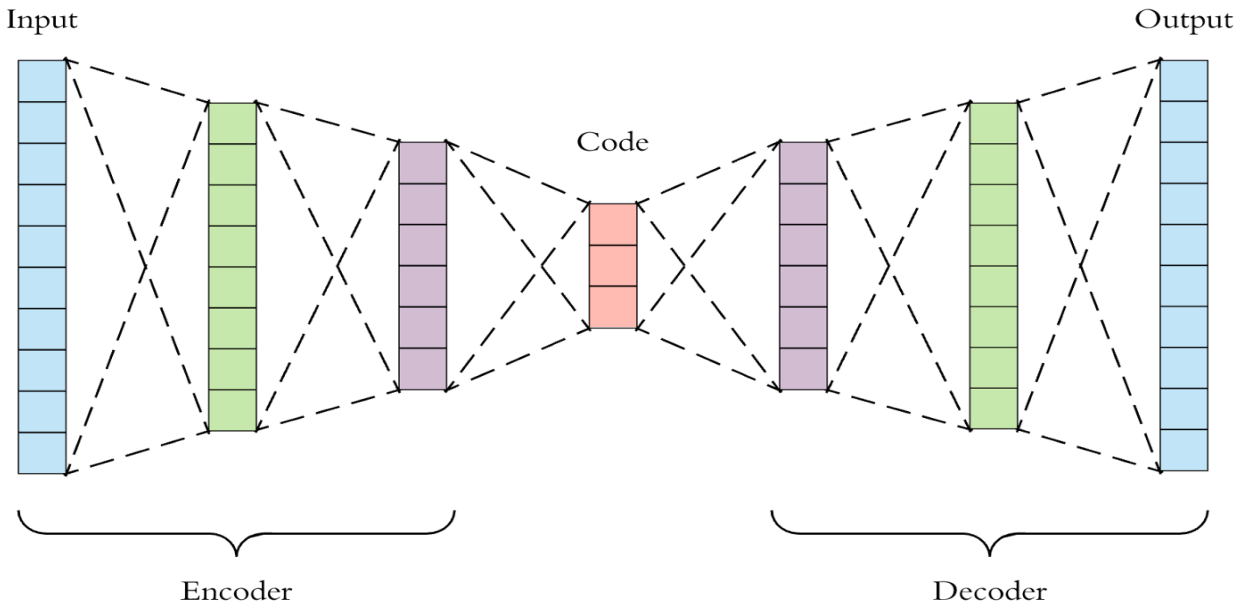


Figure 2.7 Autoencoder

2.10 Flow Definition

Flow is a collection of IP packets that pass through an observation point and have some common properties or flow keys within a certain period [55, 56]. These features are Source and destination IP, source and destination port number and protocols, Number of packets, and bytes transfer within the flow, start and end time of flow within milliseconds [55]. Flow provides information about the behavior of connection but not packet payload. Based on the source of data to be analyzed network intrusion detection system is categorized as packet-based and flow-based intrusion detection systems [55, 56].

2.11 Packet-Based Intrusion Detection

All network traffic passing a certain observation point is monitored (payload and header). Packet capturing and analysis performed at observation points such as routers, switches, and network monitors finally the packet transported to a remote analysis system. In packet-based detection well know attacks are detected, especially any attack that has been hidden within the payload. It is a very expensive, time-consuming, and computation-intensive system, especially as data grows. Signature matching is impossible especially when the payload is decrypted [55]. The hard coded signature database needs a rapid update and large storage, and encryption protocol also increases [55, 56].

2.12 Flow-based Intrusion Detection System

Based on the drawback of Packet-based intrusion detection mentioned above flow-based intrusion detection systems come up with a counter solution. Flow-based intrusion detection is based on information on network interaction rather than payload and it is also aggregated information. For many attack flow information is enough to identify them [55]. According to [55, 56] flow-based intrusion detection uses only information of network interaction, it's enough to identify many attacks [55]. Also, flow-based intrusion detection can detect an anomaly-based attack. There is an attack that can be detected based on flow, such as Denial of Service (DoS), computer worms, network probing, and flooding [56].

Flow-based anomaly intrusion detections have been proposed by researchers who have a great grasp of the domain, existing issues with current networks, and packet-based intrusion detection systems. Flow-based anomaly detection depends on the modeling of the normal behavior of flows and detecting deviations from normal behavior [57].

According to Sperotto et.al [56] flow-based intrusion detection system has different components and processes.

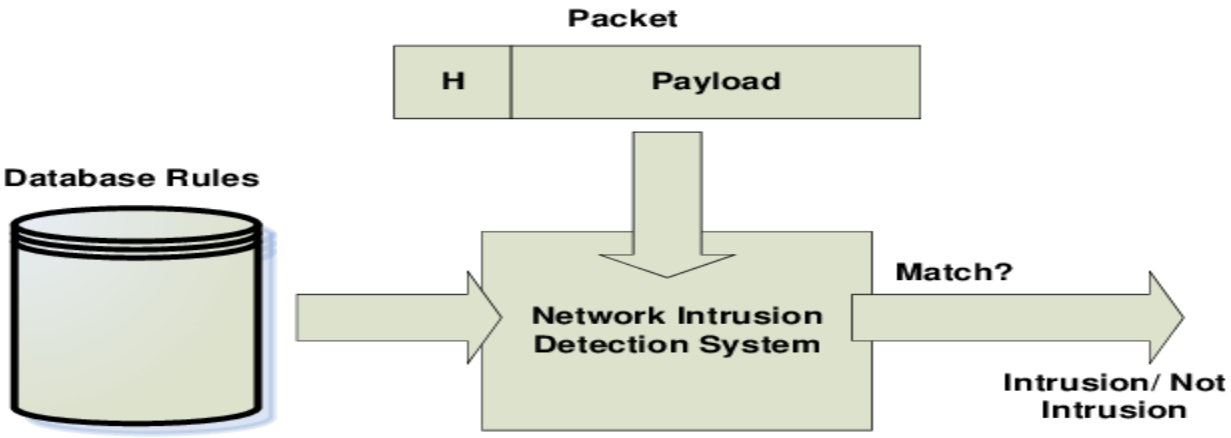


Figure 2.8 Packet-Based Intrusion detection

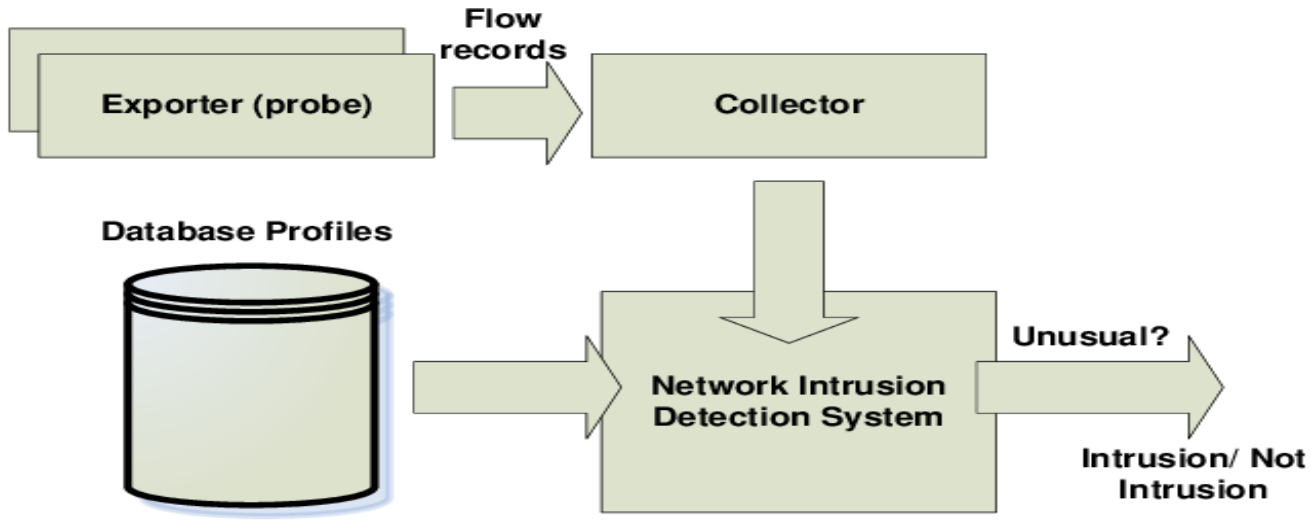


Figure 2.9 Flow-Based Intrusion Detection

Comparison of Packet-based and Flow-based Network Intrusion Detection System.

Table 2.4 comparison of packet-based and flow-based intrusion detection system

Packet-Based NIDS	**Flow-Based NIDS**
Inspect the whole payload besides headers	Inspect information of networks flow (information about connection)
Mostly signature-based NIDS	Mostly anomaly-based NIDS
Known as Deep Packet Inspection" (DPI) or knowledge base	Known as "Network Behavior Analysis".

Flow exporter (observation point): its job is metering the process or creation of records from network traffic. Each packet header is extracted and stamped with a time stamp. Then the sampling and filtering model is used to sample and filter packet header. To check if there is the same flow matching the packet header each packet header triggers an update, if not a new flow record will be created. If the flow record expires it will be sent to the flow collector.

Flow collector will receive a flow record and store it [56]. According to [58, 59, and 60] flow record is expired when four conditions are fulfilled. First, the flow is idle: form a longer given threshold time of interval no packets belonging to the flow. Example for Cisco Netflow is 15 seconds, Second Active timeout: flow reaches maximum time allowed lifetime, example for Cisco Netflow timeout is 30 minutes. Third, the FIN or RST Fags have been seen in a TCP Flow and the Fourth flow-cache memory gets full. In this case, certain Flow records are marked as expired and exported to the.

Flow Export Protocol: define the way flow records are transmitted between flow exporter and flow collector [56]. According to "Cisco Netflow version 9 and IPFIX [56] propose flexible protocols in which flow record formats can be defined by using templates. These protocols allow also a larger set of parameters to be exported, such as, for example, sampling rate and algorithm, source and destination VLAN identifiers, MAC addresses, and autonomous system numbers. An IPFIX packet is logically divided into sections known assets. A message can normally consist of three kinds of sets, namely template sets (format template exchange), data sets (flow records) and options template sets (necessary for the correct interpretation of a template set)".

Sampling: in a high-speed network millions of packets and hundreds of thousands of flow records per second could exist. If there is a flow lookup and state information is saved for each packet it requires high computing time and storage to resolve this sampling method created. Sampling has an impact on intrusion detection and flows accounting [56]. This sampling method can be categorized as packet sampling and flow sampling [56].

Flow-based network intrusion detection system research has been a focus to tackle some attacks since it depends on header information especially Denial of Services, Scan, Worms, and Botnet [56, 57, 59, 63,]. Current intrusion detection should be able to handle the following issues, growth in network line speed, increasing in internet bandwidth or Network traffic delivered per second, and dynamic network attacks [1, 48, 55, 56, 57, 59, 61, 62, and 63].

Flow-based anomaly intrusion detection is the logical choice for high-speed networks based on the following properties [55, 56, 57, 58, 59, 60, 63, 64,65, and 66]: Very small amount of data fractions needed to monitor and analyzed during detection, Net-flow has very small overhead compared to payload because Net flow record generated in exporter while in payload routers or switches did the job which consumes resource on the router and negative impact on routing, Storage is very small compared to the payload, Encrypted payload or packet is not influenced flow-based intrusion detection performance and Privacy issue is not violated. Even though flow-based intrusion detection outperforms packet-based intrusion detection it has its own disadvantage such as cannot detect attacks that hide within payload and suffer from high false positive rate [56, 57, 58, 59, 64, 70,71,72,73,74].

Why Flow-Based Intrusion Detection: Even though known attacks detected well in packet-based intrusion detection especially, attack that has been hidden within the payload it has some disadvantages such as, it is very expensive, time-consuming, and computation-intensive system, as data grows. Signature matching is impossible when the payload is decrypted [55]. The hard coded signature database needs a rapid update, large storage, and encryption protocol also increases [55].

According to [55, 56] flow-based intrusion detection uses only information of network interaction, it’s enough to identify many attacks [55]. Also, flow-based intrusion detection can detect an anomaly-based attack. Flow-based anomaly intrusion detection is the logical choice for high-speed

networks based on the following properties [55, 56, 57, 58, 59, 60, 63, 64,65, and 66]: Very small amount of data fractions needed to monitor and analyzed during detection at observation point such as Router or Switches, compared to payload-based and signature-based intrusion detection consumption of resource on the router and impact on routing is less, Storage is very small, Encrypted payload or packet is not influenced by flow-based intrusion detection and Privacy issue is not violated [56, 57, 58, 59, 64, 70,71,72,73,74].

In the last 20 years, many research paper has been done on Anomaly IDS using feature selection, feature reduction methods and data mining techniques to select important information to reduce irrelevant and redundant features from large network traffic, and reduce time complexity. Different machine learning and data mining techniques have also been used to precisely detect the anomaly attacks from normal network traffic based on the subset of features identified by feature selection, data mining techniques. Detection Rate, False Alarm Rate, and Accuracy of Anomaly-Based intrusion detection system have been improved by studding flow of network connection and by examining the payload of network sometimes by using both methods together.

Table 2.5 Table literature review

Reference paper	Purpose of IDS	Test Data used	Algorithm technique used	Advantages	Limitations
Apichit Pattawaro et.al [10]	Provide Anomaly-Based Network Intrusion Detection System through Feature Selection and Hybrid Machine Learning Technique	NSL-KDD dataset, KDDTest+ dataset.	K-Means clustering and XGBoost classification Model	false alarm rate equal to 18.20% and Use features (61.47%) to achieve this level of performance	accuracy equal to 84.41% with detection rate equal to 86.36%
Amreen Sultana et.al [14]	Propose intelligent NIDS using AODE algorithm for detection of different types of attacks.	NSL-KDD Dataset	AODE	Comply with dynamic network environment	Low false alarm rate and high detection rate compared to naïve Bayes.
Masaaki Sato et.al[22]	Develop method that automatically detects unknown attacks from an anomaly IDS alerts.	Kyoto 2006+	SVN	Can detect attack raises signature-based IDS alerts.	Low detection rate of unknown attacks that do not trigger any signature-based IDS alerts.

Reference paper	Purpose of IDS	Test Data used	Algorithm technique used	Advantages	Limitations
Monther Aldwairi et.al [24]	Provide Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model	NSL-KDD Dataset	Vote algorithm with Information Gain	accuracy of the proposed model was measured as 99.81% and 98.56% for the binary class and multiclass NSL-KDD data sets	High False alarm rate and complying with dynamin network environment
Yanjie Zhao [25].	Network Intrusion Detection System Model Based on Data Mining	Network data	C4.5, K-means clustering	High Known attack detection	Unknown data detection is low, long training time

Li Tian1 et.al[26]	propose a new model of anomaly IDS based on clustering algorithm.	KDD Cup 1999	Improved k-means clustering algorithm	High detection rate compared to old k-means clustering	Computational expense, influence of cluster number to the algorithm
Reference paper	**Purpose of IDS**	**Test Data used**	**Algorithm technique used**	**Advantages**	**Limitations**
Akash Garg et.al [27]	propose hybrid intrusion detection system using snort+PHAD detect both misuse and anomaly based attack	IDEVAL dataset	snort+PHAD	low computation time detect both anomaly and misuse attack	Low detection rate in zero day attack

How do improve the downside of flow-based IDS which is high false alarms: Even though flow-based intrusion detection outperforms packet-based intrusion detection has disadvantage, such as cannot detect attacks that hide within payload and suffer from high false positive rate [56, 57, 58, 59, 64, 70,71,72,73,74]. Counter solution used in this study to solve the drawback of high false alarm rate for flow-based IDS is performing dataset preprocessing. First encoding categorical data that is each flow is categorized as attack encoded to one and normal encoded to zero, dataset normalization was applied to rescale to range values between 0 and 1 from value ranges single digit to 10 digits in dataset, a large number of features increase computation complexity, increase resource usage, false-positive rate and time consumption is high [80]. To solve the mentioned drawback finding relevant and significant features of the dataset CICIDS2017 increases the accuracy of the neural network [80] besides Autoencoder is a deep learning algorithm that is known for its capability of dimensional reduction and outperform other dimensional reduction algorithms [14]. Building Denoising Autoencoder which has a capacity or reducing large dimensional dataset, free from identity function, outlier detections, avoid overfitting, train the learning algorithm on training dataset, tune it parameters to find optimum value on validation dataset, using an adaptive optimization method Adaptive Moment Estimation (Adam) to find optimum value for parameters with minimum loss function. Adam is efficient computationally, for a problem with large data and parameter is well suited, also suited with a problem with very noisy data and spare gradients, responds fast, work with squares of gradients and the average of the second-order moment [89,91,90].

CHAPTER THREE
METHODOLOGY

This chapter will discuss the proposed research methodology, the experimental setups, and related issues.

3.1 Research Methodology

The research methodology followed in this study is shown in figure 3.1below.

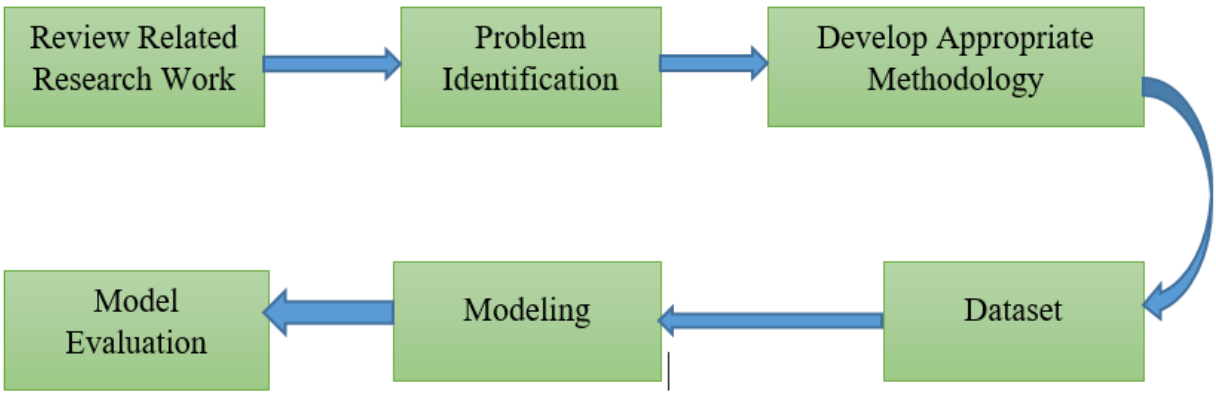


Figure 3.1 Methodology

The first job of this research was to review related research work to grasp knowledge about the domain. In this study, better results has been achieved in termers of accuracy, error reduction, less computation expense. These results were achieved by utilizing machine learning with a different approach. But those work has drawback, as a result cybersecurity is still a concern. According to Danny plammer and Abdulla et al., current network intrusion detection system suffers from the following challenges [3, 4] . The rate of new cyber-attack exponentially increased every year, the speed of network lines growth, a large amount of network traffic flow over the network which is difficult to monitor and analyze, identify, and detect the attack. Besides storage and computation capacity required is high [3, 4]. According to A.Midzic et al. some of the drawbacks of current network intrusion detection systems are listed below [43].

- ✓ The performance of intrusion detection system is poor in real environment

- ✓ Learning capacity of intrusion detection system is poor when dataset size increases
- ✓ Accuracy of intrusion detection system decrease while dataset size increase
- ✓ Intrusion detection system works well for specific dataset

3.2 Dataset

To solve cybersecurity challenges of our time, Canadian Institute for Cybersecurity(CIC) brings together researchers and practitioners from a different academic discipline to create technology and carry out groundbreaking research [75]. CICIDS2017 is a dataset generated by Canadian Institute for cybersecurity. CICIDS2017 includes normal network traffic flow and current known common attacks. CICIFlowMeter is used to produce network traffic analysis with labeled flows [75]. According to Sharafaldin et al., CICIDS2017 tries to generate natural network traffic by modeling the behavior of human interaction [75] . Abstract behavior of 25 users based on the HTTP, HTTPS, FTP, SSH, and email protocols. The data capturing period was five days from Monday, July 3, to Friday, July 7, 2017. Known attack at the given year was implemented to simulation environment to make the dataset realistic. Those most common attacks were chosen based on the 2016 McAfee report attack includes Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS.

According to Sharafaldin et al, eleven criteria were used to build the naturalistic dataset, all those eleven criteria were not fully considered by the previous dataset, such as complete network configuration, complete traffic, labeled dataset, complete interaction, complete capture, available protocols, attack diversity, heterogeneity, feature set, and metadata [75]. Even though Canadian Institute for cybersecurity generate the latest dataset called CSE-CIC-IDS2018 on AWS but it's not publically available to use yet. Finally, the CICIDS2017 was chosen by this study because it is more realistic dataset compared to other publically available dataset according to Sharafaldin et al. [75] and latest one during this study performed.

Dataset preprocessing includes several processes. Integration of dataset is one of the activities of dataset preprocessing which allows different data frames is integrated into a single data frame. Single dataset is easy to be managed. This study used CICIDS2017 dataset contains a categorical value which needs to be converted to the numeric value. Each feature in the dataset contains a different range of values, magnitude, datatypes, and scale. To make all features contribute equally

to the result, data transformation is applied besides normalization. The curse of dimensionality is one of the biggest challenges to handle. Network traffic flow increase from time to time with the bandwidth of the network backbone as increased. It is difficult to process every network traffic flow in the dataset. Developing compressed representation is one of the solutions to reduce the dataset dimension. The famous deep learning algorithm Autoencoder takes input on its input layer and developed low level compressed representation of dataset in its bottleneck layer and build the original input from the low level compressed. Dimension reduction concerns with the most important feature this results in increasing the performance of deep learning algorithm and to overcome the curse of dimensionality. Dataset CICIDS2017 contains a missing value. This research chooses to apply ignoring techniques by simply deleting missing value rows. According to zhang et al. ignoring techniques is widely used and simple, especially when the dataset is big in size [76]. Duplicate values in the dataset result in accuracy decreasing and consumption of computational resources, so removing redundancy is vital in addition to make the dataset consistent [77,78]. To train and evaluate the model the dataset is divided into three samples, these are training, validation, and test dataset.

Encoding the Categorical Data: Encoding categorical data value is encoded to a numeric value. Machine learning requires all inputs values must be numeric before fit to the model and evaluation. The CICDS2017 dataset contains 15 categorical values, fourteen of them is attack type and one is normal network flow. The proposed model in this study identify whether the network flow is normal or attack, so encoding all attack category to number one and normal network flow to number zero was applied.

Data Transformation: Each feature in the dataset may contain a different range of values, magnitude, datatypes, and scale. To make all features contribute equally to the result data transformation is applied. Besides normalization was also used to rescale to range values between 0 and 1. Data transformation can have achieved through data normalization [78].

Normalization: Deep learning algorithm trains and develop a model then expected output compared with actual output in hopping the [78] difference is minimum (minimum loss function). Gradient Descent techniques scale the magnitude of parameters. During searching optimum values for hyperparameters small incremental update results in converging on the optimal value on the

other hand drastic update results in divergent behavior. The mentioned drawback was overcome by using learning rates. The loss function must be at the lowest value while finding the optimal value of parameters with the help of the learning rate. Min-Max normalization is a type of normalization it changes the range of data, scaled to a fixed range. The normalized dataset is proportionally distributed and slightly uniform. Normalized input helps the neural network layer to learn the parameters at the beginning of the layer. The relationship between the original value of dataset is preserved after normalization [78]. This study selected min-max normalization because it intended not to lose the relationship of original dataset values.

Missing Value and Duplicate Values Handling: According to R.B Kline there are three methods to handle missing values in data science [79]. Ignoring method [76], imputation method and last one is using modeling method [77]. According Luengo et al., ignoring techniques is widely used when the dataset is big in size [78]. Duplicate values in the dataset result in accuracy decreasing and consumption of computational resource so removing redundancy is vital beside it helps to make dataset consistent [77,78]. Ignoring techniques is choosing for this study because, CICIDS2017 dataset contains over 3.1 million instance of network flow which is large.

Data Dimensional Reduction: A large number of features increase computation complexity, increase resource usage, and time consumption is high [80]. To solve the mentioned drawback finding relevant and significant features of large network traffic is needed besides it increases the accuracy of the neural network [80]. Even though there are several feature reduction machine learning algorithms, Autoencoder is the ideal one. Autoencoder is a deep learning algorithm that is known for its capability of dimensional reduction and outperform other dimensional reduction algorithms [14]. This study uses big dataset size which needs to be reduced and Autoencoder is chosen for its capability to reduce large input to small representation.

Training/Validation/Test Dataset Split

Training Dataset: a subset of the normal network traffic flow dataset used to train deep learning networks. Autoencoder train only on the training dataset and try to create a model by studying it. Any deviation from the developed model is considered as an outlier or anomaly.

Validation Dataset: is a subset of the dataset which is attack free instance network flows. Validation dataset used to tune the parameter of the developed model, such as find an optimal number of hidden units or determine the stopping point.

Test Dataset: the subset of the dataset which includes an instance of Benning and attack network flows. Used to assess the performance of fully trained deep learning.

3.3 Model Development and Architectures

3.3.1 Autoencoder(AE)

The concept of Autoencoder was originally proposed by LeCun in 1987 [81]. Autoencoder is a type of feedforward neural network, that is trained to regenerate input dataset as an output. Autoencoder compresses the input to lower-dimension and reconstructs output from it. The architecture of Autoencoder has three modules, namely Encoder, Bottleneck, and Decoder [82].

Encoder: It's the input layer of Autoencoder, it takes the input then encodes it by compressing. The compressed representation of the input data dimension is reduced and it is not the same as the original dataset.

Bottleneck: Holds the compressed representation of the input. The hidden layer of AE is known by different names in literature including bottleneck, discriminative, coding, or abstraction layer.

Decoder: It is the output layer of the Autoencoder. responsible for the reconstruction of the original input data form compressed representation.

Both encoder and decoder layer consists of the same number of nodes in most cases. In practice, AEs are created from multiple layers where the outputs of the preceding layer are connected to inputs of the following layer. Learning task accomplished by capturing the most significant features of training data at the hidden layer [81,82].

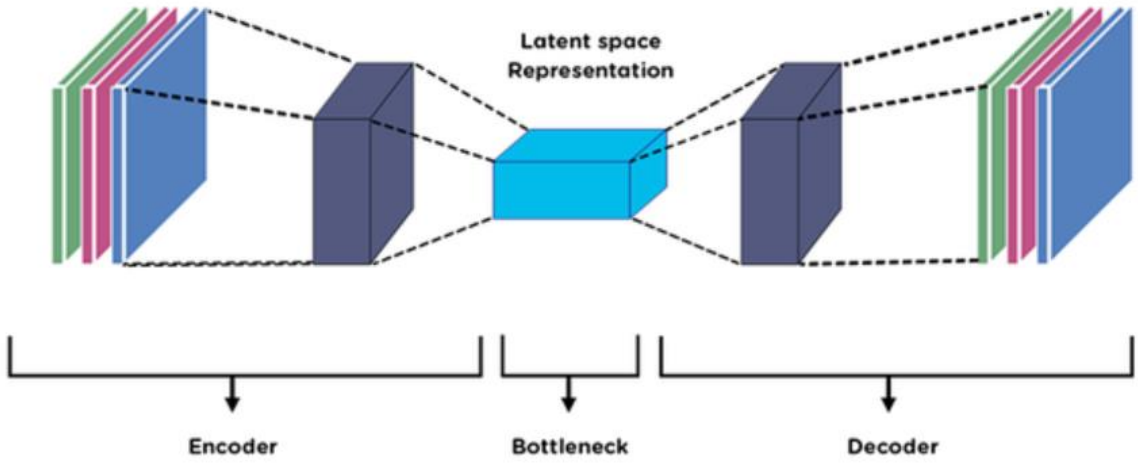


Figure 3.2 Autoencoder Architecture (Tara Larrue et al. 2018)

Deep AE contains more than one hidden layer which maps from input to bottleneck. Adding Depth has the effect of exponential reduction in computation cost and amount of training data needed for representing many functions [83]". Autoencoder used for dimensionality reduction, Denoising , and outlier detection [81,82,83,84]. There is some Autoencoder architecture such as Sparse Autoencoder, Contractive Autoencoder, Convolutional Autoencoder, and Denoising Autoencoder. Each of the architecture has pros and cons, this paper focuses use Denoising Autoencoder [15].

3.3.2 Model Building

Identity Function is one of the drawbacks of Autoencoder, it occurs when nodes of the hidden layer outnumbered the input. In simple terms when the input is copied to output [84]. To get rid of the identity function problem Denoising Autoencoder (DAE) was proposed by Vincent et al. 2008 [4].

Denoising Autoencoder (DAE) algorithm is used to develop the model. DAE is a type of Autoencoder that receives the corrupted output instance and is trained to recover the original uncorrupted input as its output. DAE does not replicate input at output rather it captures the most significant features. Inputs are randomly corrupted and DAE excepted to reconstruct pure original input as its output, aims to produce a robust representation [84]. Introducing noise or corrupted input not only a solution for solving identity function, but also the reduce overfitting of the model generated by DAE [84]. Advantage DAE is identity function free, low overfitting, high capability

of dimensional reduction. The output of each neuron is passed to the next layers of neurons as input, to activate this neuron activation function is used in this study. Activation function decides which neurons are activated or not [84]. The architecture of DAE is shown in figure 15 below. to overcome the current challenge and drawback back intrusion detection system as explained in section 3.1 this study chooses Autoencoder and its variation DAE, for their advantage such as high capability of dimensional reduction, outlier detections, identity function free, low overfitting, this study chooses to apply.

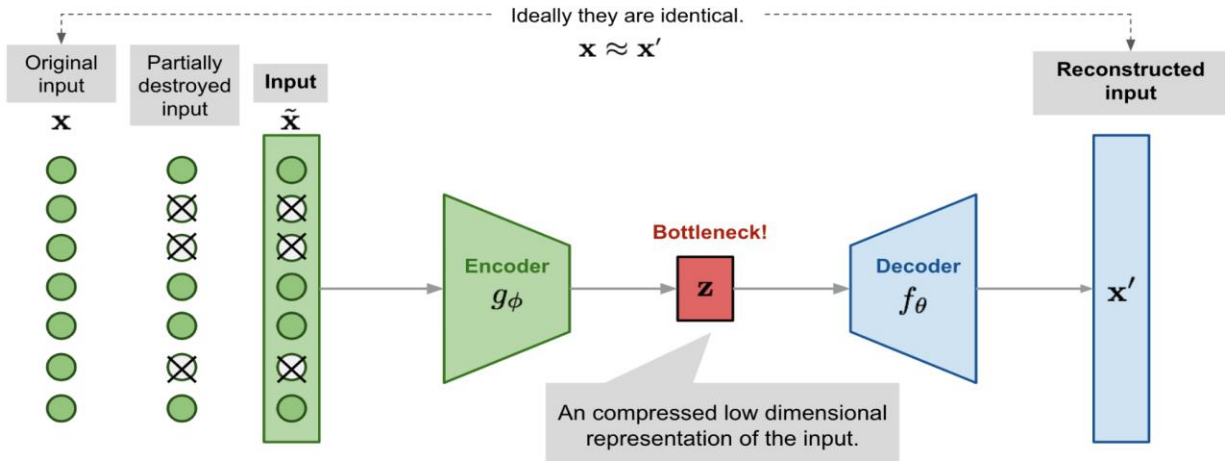


Figure 3.3 Denoising Autoencoder (Hinton, et al. 2012)

Gaussian Noise: Gaussian noise layer is applied as a regularization layer and only active at training time. This layer randomly corrupt input layers of the neural network. Gaussian noise layer is a technique used to avoid system overfitting and identity function [85].

Activation Function: A neural network consists of multiple neurons that are clustered into several layers, neurons in the previous layers connected to the neurons of the next layer, each connection have weight W, bias b, and activation function corresponding to it. To perform complex tasks neural network needs an activation function. A neural network without activation function is a linear regression model. The output of the single neuron is calculated and passed to the next layers of neutrons using activation function this process is continuously repeated. The activation function is used to introduce non-linearity into the output of a neuron besides decide whether the neurons need to be activated or not [86].

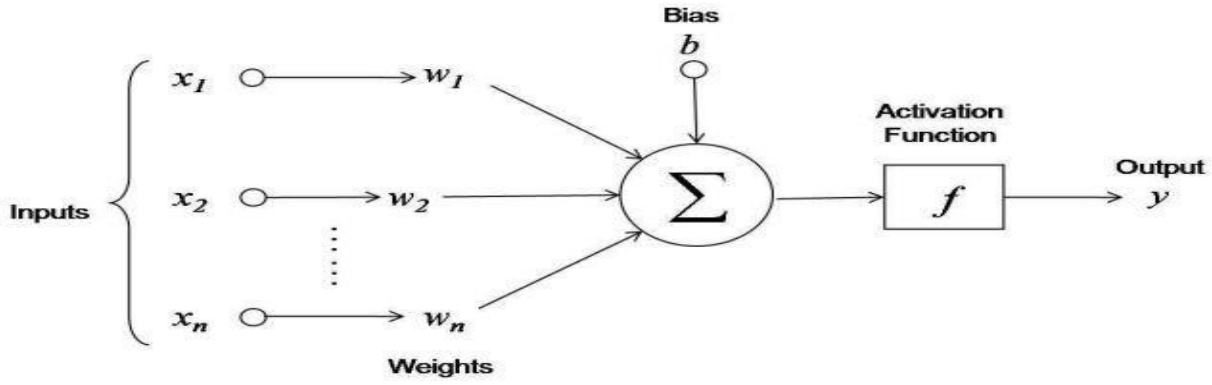


Figure 3.4 Activation Function In Deep Learning (Paul 2020)

Y=Activation$\sum$ $(weight * inputs) + bias)$

Exponential Linear Unit (ELU) Activation Function: It is a type of activation function which is continuous and differentiable at all points. If the values of input x are positives, the output of the function is x, while the value of input x is negative, the output is Expo(z)-1. The gradient descent value of the ELU function is 1 for all positive values and Expo(x) for negative values [86,87]. The equation of ELU is shown below.

y = ELU(x) = expo(x) − 1; if x<0 and y = ELU(x) = x; if x≥0

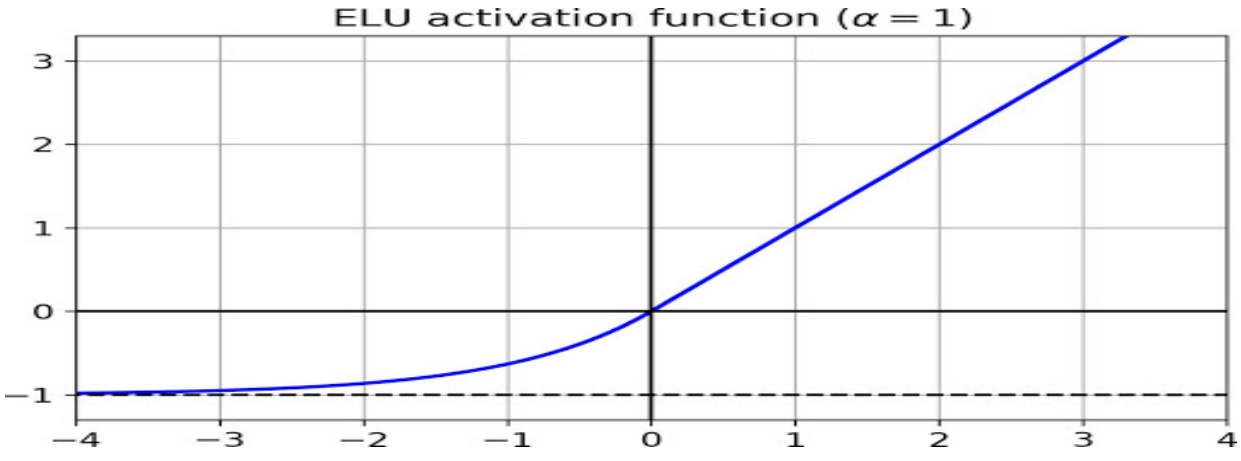


Figure 3.5 ELU Activation Function (datascience.stackexchange.com 2018)

ELU has some advantages compared to other activation functions such as its continuous and differentiable at all point, faster training time specially ReLU and its types, have no dying neuron

problems (it is non zero for all negative values), have no vanishing gradient problems, and finally achieves higher accuracy compared to Hyperbolic Tangent, Sigmoid and ReLU variants and significantly better generalization performance [86,87,88].

Optimizer: Good models have minimum loss function while finding the optimal value of parameters with the help of the learning rate [88,89]. The learning rate of deep learning must be perfect for finding the optimal value of parameters quickly. it should be between zero and one. Smaller learning rates changes the parameters slowly, requires more training epoch even the process may stack. in reverse, if the learning rate is too large models converge quickly to a suboptimal solution [88,89]. The optimization method helps the algorithm to achieve the optimum solution for nonlinear problems, better success rate, and frequently and quickly update parameter values [21]. Such as Stochastic Gradient Descent (SGD), Adadelta, Adam, RmsProp, Nadam optimization methods were used in the study on Keras.

Adaptive Moment Estimation (Adam): Is an adaptive optimization method, have advantages of popular methods AdaGrad and RMSProp AdaGrad works well with sparse gradients and RMSProp works in on-line and non-stationary setting [89,90] during the training of deep learning on large scale data Adaptive Moment Estimation helps to optimize parameters [91]. Adaptive Moment Estimation have many pros such as memory requiems is little, efficient computationally, for a problem with large data and parameter is well suited, also suited with a problem with very noisy and spare gradients, responds, fast, work with squares of gradients and the average of the second-order moment [89,91,90].

Application area of Flow-Based Intrusion Detection: This study assumes flow-based anomaly intrusion detection system is placed at the backbone of main network line observation point such as Firewall or Router to monitor incoming and outgoing network flow. Placing flow-based anomaly intrusion detection at such tactical position is help to analyze and identify network attack before it hits the entire network and damage happens both on network based resource and network itself. The intrusion detection system is applied to banking operations, online auctions, electronic commerce applications, social networking, and online application/ registration, etc.

3.3.3 Model Evaluation

After Denoising Autoencoder trained on training dataset, tune parameters of model on validation dataset and develop model. Finally, the developed model performance was evaluated on a test dataset. Model evaluation helps to understand how was the model performed. The terms below used to determine the performance of the model.

Accuracy: Total number of correct identification for attack/benign data concerning total made. If the accuracy is higher, the machine learning model is better [89,90].

Loss: Model loss of the proposed system evaluated. This model loss is expressed with two model loss variations that are a model loss against the test dataset and model loss on the validation dataset [89,90]. The goal of training a model is to find a set of weights and biases that have low loss [91]. Model loss measured using Mean Squared Error (**MSE).** MSE is useful when the dataset contains a lot of noise, dataset contains outliers, or unexpected values (too high or too low values) [91].

Reconstruction Error: DEA corrupts and encode the input into a compressed representation at the bottleneck layer then recover the original data at the decoding layer. The loss function explains the difference between the original input and the output at the decoded layer. If the loss function of a model is small it implies the Autoencoder is well performed because the original input and predicted output are almost similar [86,87,88,89,90].

Area Under Curve (AUC): indicates the performance of the system which fail to distinguish outliers or attack from normal network traffic. AUC tells how much model is capable of distinguishing between attack and normal network flows. Higher the AUC, better the model identifying network flows of attack and normal [86,87,88,89,90].

Receiver Operating Characteristic Curve (ROC): is a graph which tells performance of a model at all classification thresholds with a true positive rate against the false-positive rate at all possible threshold [86,87,88,89,90].

Training Time of Model: is a period of time that deep learning spent to learn dataset and identify attack network flow from normal network flow. The more the model take long time to train on

dataset, the fast the cyber-attack happen before it identified which makes the model less usable [3,5,86].

Latent Space: Latent space representation tells how well the model clusters the normal network flow from attack network flow in bottle neck layer of Denoising Autoencoder:

3.5 Materials

This section describes both hardware and software tools used in this research

3.5.1 Hardware Tools

The hardware tools used in this research are CPU: Intel ® Core ™ i5-4210 CPU, Core: Two Cores, Computer Speed 1.70 GHz, RAM size:12 GB, Graphics: AMD RADEON, Operating System: Windows 10 64bit.

3.5.2 Software Tools

3.5.2.1 Python programming language

Python is a pseudo-code programming language nature, Easy to Learn, a high-level programming language for general-purpose programming, Guido van Rossum in 1999 propose python to develop suitable computing for students, that is simple and effective for analysis and program development [92]. Now python is given at most USA university computer science departments [80]. Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. Python includes various data structures and comes up with a large and comprehensive standard library [92]. Python has many advantages compared to C, C++, Java, JavaScript which is few lines of coding and High Readability or highly visible code to the eyes, Portability and Flexibility to choose OOPs approach or perform scripting, platform-independent, ability to balance high-level programming with low-level, libraries for numerous need of an AI project such as SciPy for advanced computing, Pybrain for ML and NumPy for scientific computation. Python has popular libraries such as NumPy, TensorFlow, Pandas, Scikit-Learn, Matplotlib, etc. [79].

NumPy: Stands for Numerical Python. NumPy is the fundamental package needed for scientific computing with Python. This package contains an N-dimensional array object, basic linear algebra functions, basic Fourier transforms, random number capabilities, etc. [93].

Pandas: Pandas is the best library for data preparation. It is capable of various input/output data formats such as Excel, CSV, Python/NumPy, HTML, SQL, and more. handles tabular, multidimensional, heterogeneous, and time-series data. It can help to analyses real-world data in Python. pandas have powerful querying possibilities, statistic calculations, and basic visualizations [94].

Matplotlib: Matplotlib is a Python implementation of the MATLAB-like plots and most widely used for data visualization. Libraries like pandas and seaborn are built on the top of Matplotlib. It is easy to get a sense of data using Matplotlib and to draw any graph. Matplotlib can generate plots, bar charts, histograms, power spectra, scatterplots, error charts, pie charts, and many other types [94].

Seaborn: Seaborn library is Matplotlib based libraries. It helps to visualize and enhance understanding of data. By supporting NumPy and panda's data structure produce informative plots. internally perform the necessary semantic mapping and statistical aggregation [94].

SciPy: The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation. Used to manipulate numbers on publishing the result on the computer. The SciPy library is built to work with NumPy arrays. They are independent of operating systems, easy install, and open source. [95]

TensorFlow Framework: "TensorFlow (TF) is a popular library that can help to create Deep Neural Networks(DNNs). TF is based on a Directed Acyclic Graph (DAG) that contains nodes, which represent mathematical operations, and edges, which encapsulate tensors. This computational graph offers a high-level of abstraction to represent the algebraic computation of the constructed DNN models, independent of the programming language used, the execution environment, and hardware. Thus, it provides TF users with the flexibility to execute their computation on one or more CPUs/GPUs, or even mobile devices. TF uses a concept known as deferred execution or lazy evaluation. This means that there are two principal phases in a TF

program: (1) a construction phase, that assembles a graph, including variables and operations; (2) an execution phase that uses a session to execute operations and evaluate results in the graph. The constructed DAG allows the TF XLA compiler to generate an optimized and faster code for any targeted deployment environment. Indeed, the TF execution engine schedules the designed computations' tasks in a way that ensures the efficient use of resources" [96].

Keras: Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It is used to solve deep learning and machine learning problems. Keras is very helpful for engineers and researchers to fully access the platform of TensorFlow. After Keras run on TPU or a large cluster of GUPs, the Keras model can run on a browser even on a mobile device [97]

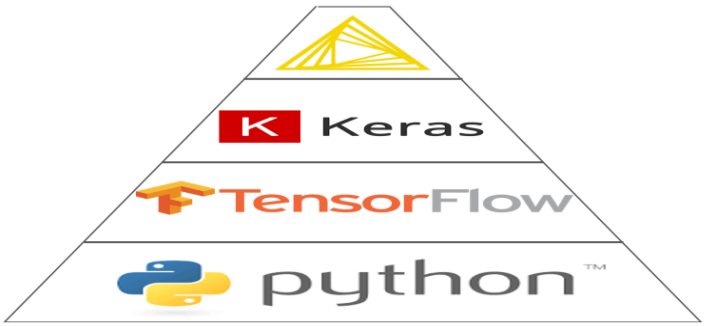


Figure 3.6 Python Modules Architecture(Rosaria Silipo 2019)

Why Python: For data analysis and interactive computing and data visualization, Python will inevitably draw comparisons with other open source and commercial programming languages and tools in wide use, such as R, MATLAB, SAS, Stata, and others. Advantage of python compared open source and commercial programming languages is, Python's improved support for libraries, support data analysis tasks, ease of use, huge ecosystem consisting of a number of libraries for every aspect of data science and its reliance via NumPy and SciPy wrappers on the fast implementations of a large number of scientific algorithms

Data preparation is a vital step in the process of data analysis, and it includes data preprocessing and data manipulation. Preprocessing aims at cleaning, integrating, transforming and reducing the original raw data so that it can become usable for data analysis, and transforms the preprocessed dataset into a data format that could easily manipulated by the data modeling algorithms. Pythons'

play a major role in data analysis. In this sturdy, there are many framework and libraries of python implemented which clearly shown in Appendix A. In this section, major advantages of using python other than open source and commercial programming languages described.

Data visualization

To understand more CICIDS2017, data visualization techniques needed. Matplotlib and Plotly is simple, customized and the most widely visualization tool of python. To visualize the size of normal and attack network flow in this study Matplotlib and Plotly applied, good example is figure 4.3, 4.4.4.7,4.8 and 4.9.

Data preparation and Manipulation

Data set CICDS201 out of 85 feature 84 contains value of numeric. For sorting and manipulating, such numeric data Panda is effective. Data normalization putting each value of features to some fixed range scale to avoid unbalanced feature impact to the model is a good example. Python helps to analyses tabular, column-oriented data structure with both row and column labels, which CICIDS2017 dataset is.

CICIDS2017 comes with eight different data frames, for manipulation purpose, aggregation is required to form single data frame CICIDS2017 sliced into three different frames known as training, validation, and test dataset. Cleaning CICIDS2017 from duplication, unwanted character's, and missing value for all above listed operation used panda module of python.

Deep learning and Data modeling

Tensorflow and Keras used applied in this study, to implement Denoising Autoencoder which is a type of deep learning and neural network with single input and output layer, 7 hidden layer, a Gaussian and encoding layer to train on CICIDS2017 and develop model to identify attack or anomaly network flow. To activate nodes in each layer, which performs transform on input activation function (ELU) applied. During evaluation of performance of the developed model one criteria was measuring how much was the system loss. To optimize the parameters of model while decreasing loss of the model Adam optimizer implemented.

CHAPTER FOUR
IMPLEMENTATION

4.1 Preprocessing

Dataset Integration: Dataset CICIDS2017 is coming with eight separate data frames. Each data frame has the same number of features, that is 85 features. Out of 85 features, 82 features data types are float data type and the rest is string datatypes.

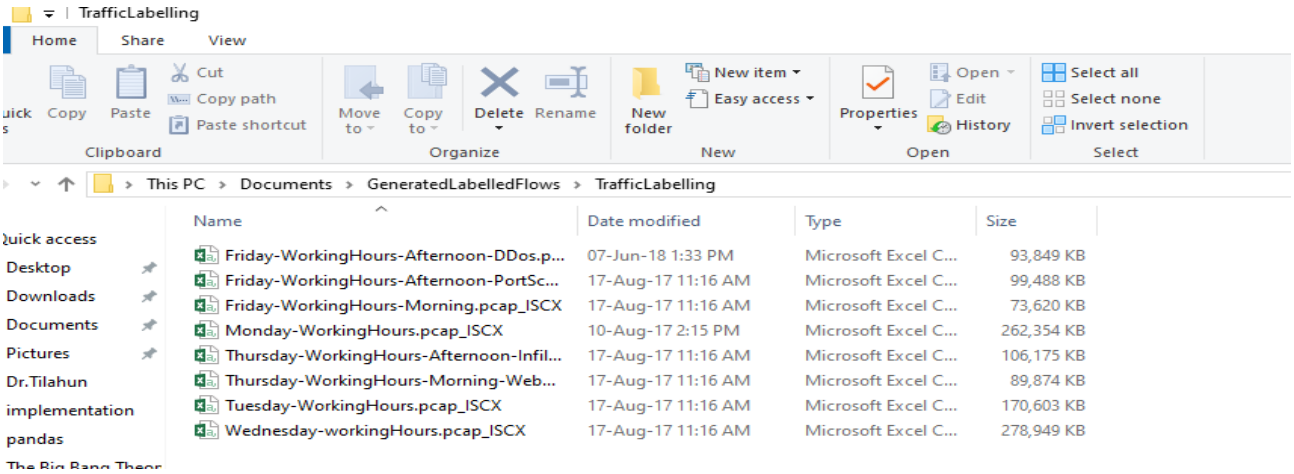


Figure 4.1 Dataset CICIDS2017 Before Integration

Using Python programming language by integrating the eight data frames, single data set is created, with a total of 3119345 instances. This dataset has 85 features and 15 distinct labeled classes. Each instance labels benign for normal network flow and fourteen attack labels as shown in figure 4.1.

Encoding the Categorical Data: The label feature or column contains 15 categorical values, that are benign and 14 different attack types. Fourteen attack categorical value is, Web Attack XSS, Web Attack Sql Injection, Web Attack Brute Force, SSH-Patator, PortScan, Infiltration, Heartbleed, FTP-Patator, DoS slowloris, DoS Slowhttptest, DoS Hulk, DoS GoldenEye, DDoS, Bot. To make the dataset suitable for modeling and binary classification, benign has been encoded to zero (0) and all attack values are encoded to 1 as shown in figure 4.3 and 4.4.

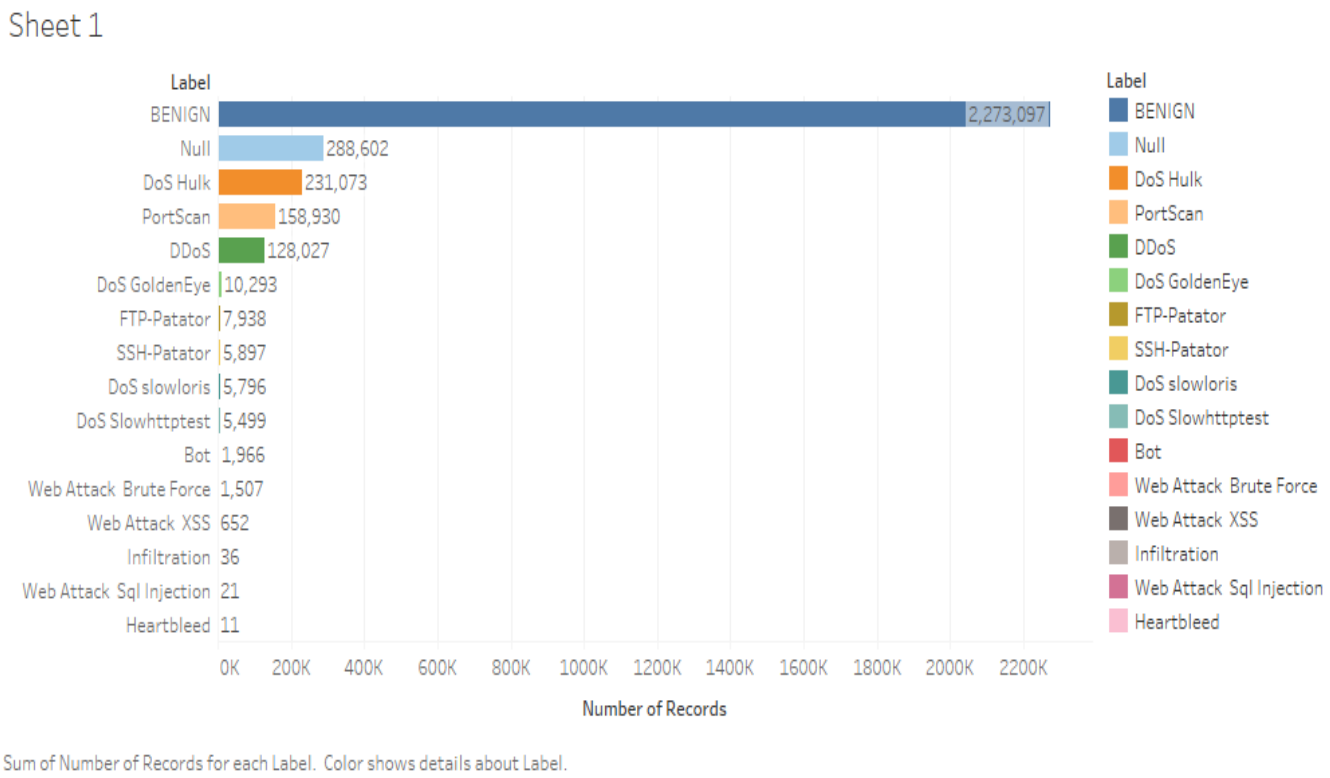


Figure 4.2 Sum of Number of Records for each Label

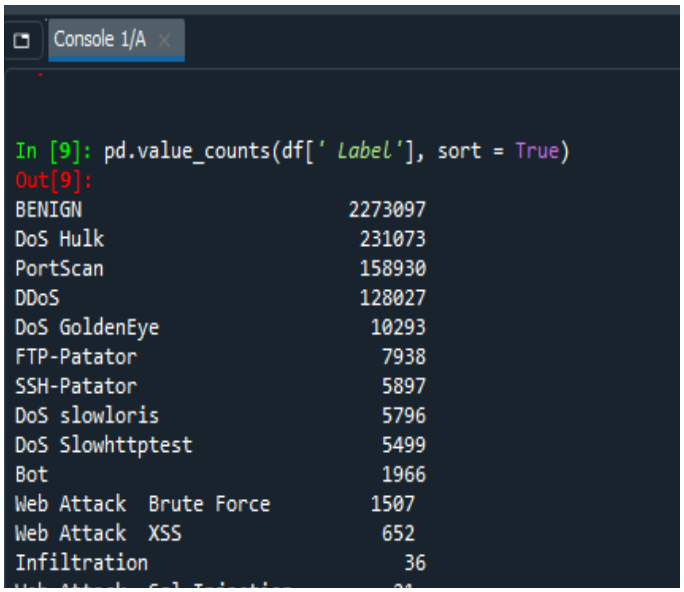


Figure 4.3 Label feature categorical values before Encoding

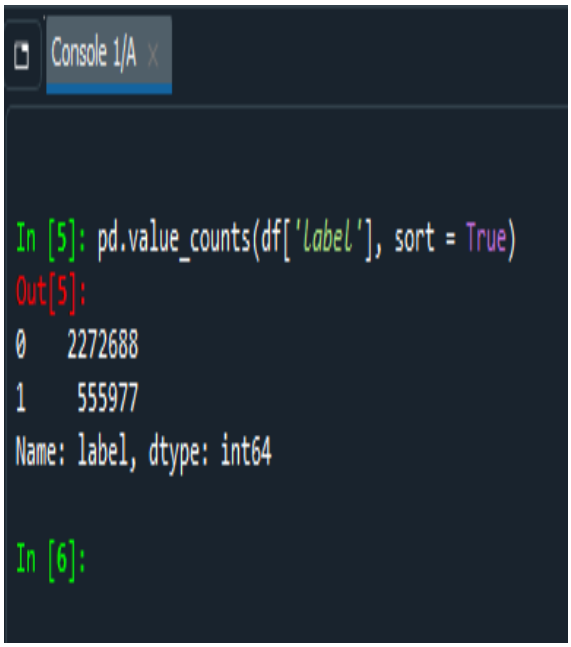


Figure 4.4 Label feature categorical values After Encoding

Python Programming Language not processing the dotted-decimal IP address such as 192.168.20.2. To overcome such drawback each IP address converted to the corresponding decimal. As a result, 17007 source IP and 19114 destinations IP address converted to the corresponding decimal number.

Table 4.1 IP Address Encoding

Source IP Address	**Source IP Address convert to decimal**	**Destination IP Address**	**Destination IP Address convert to decimal**
104.16.207.165	1745932197	192.168.10.5	3232238085
104.16.28.216	1745886424	192.168.10.16	3232238096
104.17.241.25	1746006297	192.168.10.0	3232238080
104.20.10.120	1746143864	91.86.253.245	1532427765
73.112.226.1	1232134657	76.176.23.2	9876543234

Special Character Handling, Datatypes, and Feature Removal: Dataset CICIDS 2017 contains special characters such as #, @, &, - and space. The special character needed to be removed from the entire dataset because of this character has no functional usage for modeling. Python programming language needs to have every dataset feature in integer or float datatypes to manipulate dataset. All columns 81 columns datatype is changed to integer or float.

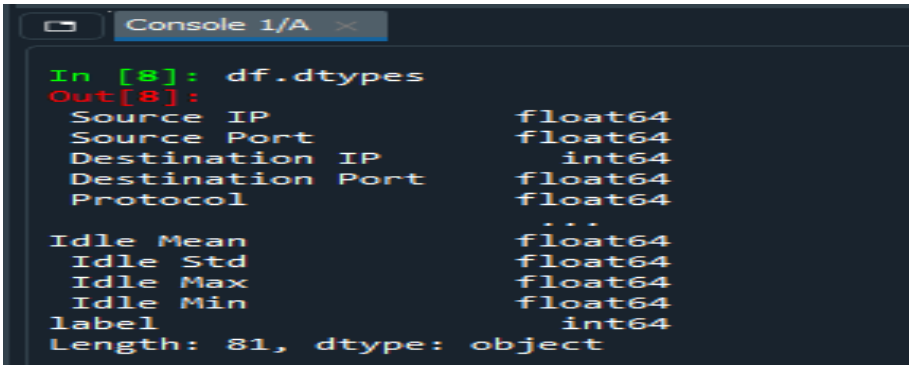


Figure 4.5 Datatype of all Features

Missing Value Handling: In CICIDS2017 all instance was not complete, 9.25% (288602) record of the original dataset was null which is not important for detection of anomalies. As mentioned in section 3.1 ignoring techniques or simply deleting missing values records was chosen by this study. The total instance is reduced to 2,830,743 and the corresponding percent of each label also changed as shown in figure 4.6.

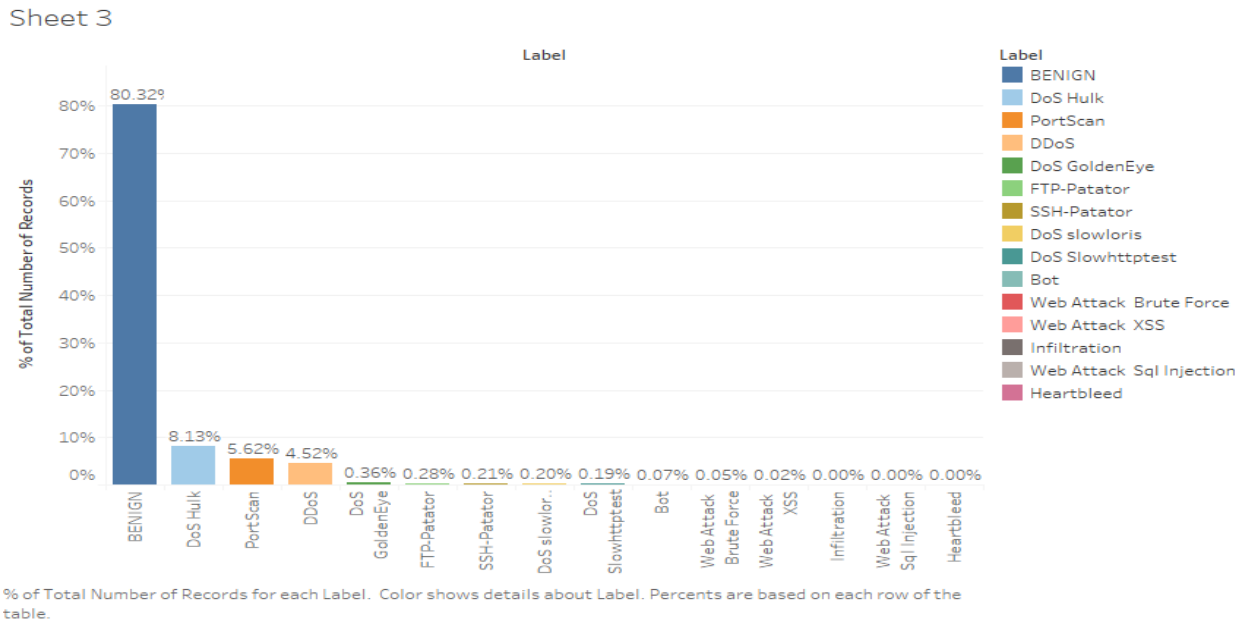


Figure 4.6 Total Number of Records for each Label in Percent After missing value handled

There are unbalanced network flows, as depicted below 80.32% of total instance is normal network flow while the rest 19.68% of network flow was attacks, after ignoring technique applied on dataset which removed all null records. The result is shown in table 4.2.

Table 4.2 Total Number of Records for each Label Before and After Null Value Removed .

Label(Before Ignoring Technique applied)	**Percent of each Label**	**Label(After Ignoring Technique applied)**	**Percent of each Label**
BENIGN	72.87%	BENIGN	80.32%
Null	9.25%		

Label(Before Ignoring Technique applied)	Percent of each Label	Label(After Ignoring Technique applied)	Percent of each Label
DoS Hulk	7.41%	DoS Hulk	8.13%
PortScan	5.09%	PortScan	5.62%
DDoS	4.10%	DDoS	4.52%
DoS GoldenEye	0.33%	DoS GoldenEye	0.36%
FTP-Patator	0.25%	FTP-Patator	0.28%
SSH-Patator	0.19%	SSH-Patator	0.21%
DoS slowloris	0.19%	DoS slowloris	0.20%
DoS Slowhttptest	0.18%	DoS Slowhttptest	0.19%
Bot	0.06%	Bot	0.07%
Web Attack Brute Force	0.05%	Web Attack Brute Force	0.05%
Web Attack XSS	0.02%	Web Attack XSS	0.02%
Infiltration	0.00%	Infiltration	0.00%
Web Attack Sql Injection	0.00%	Web Attack Sql Injection	0.00%
Heartbleed	0.00%	Heartbleed	0.00%

Dataset Normalization: Dataset CICIDS2017 has a value with a different scale as shown in the figure 25. To make all features contribute equally to the result data transformation is applied. Data transformation can have achieved through data normalization changing the range of data, scaled to a fixed range is necessary. Using min-max technique rescaled the range of values between 0 and 1. Dataset before and after normalization is shown in figure 4.7 and 4.8 respectively.

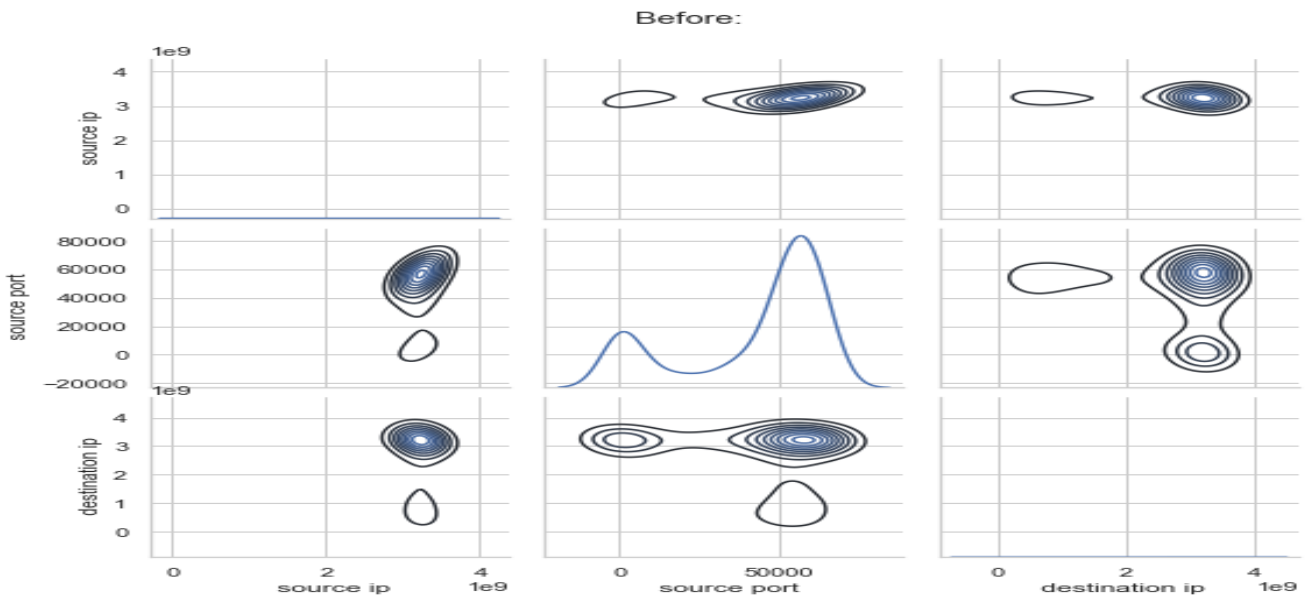


Figure 4.7 Dataset Before Normalization

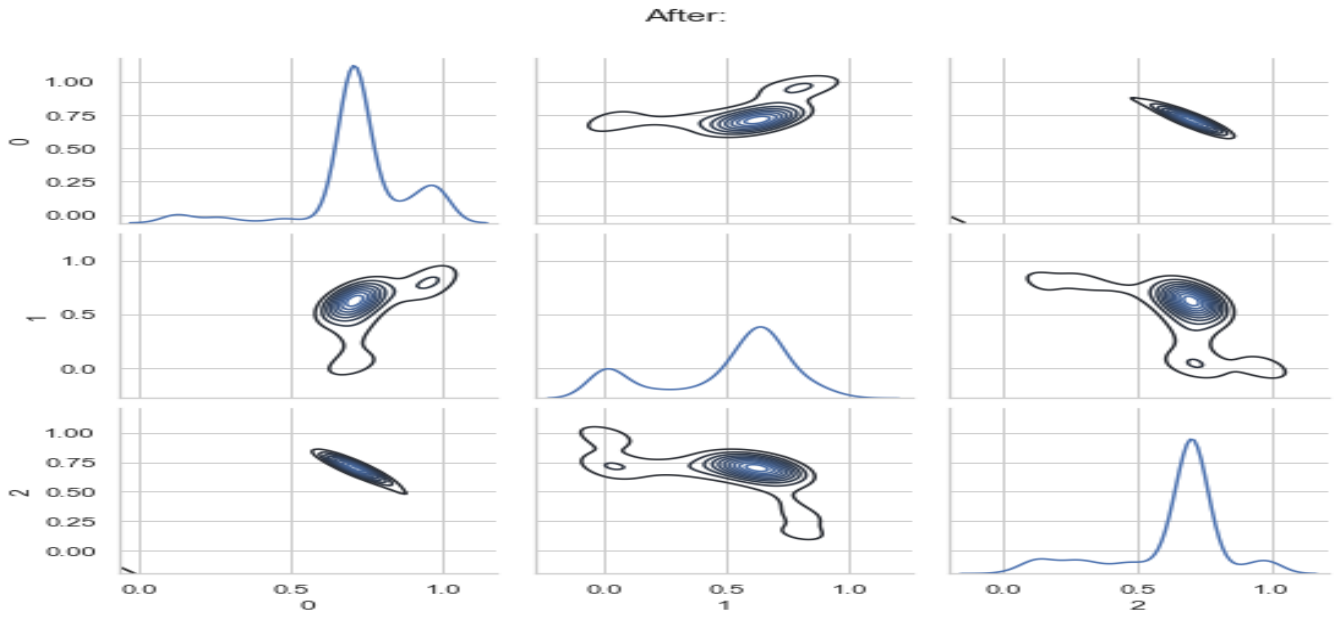


Figure 4.8 Dataset After Normalization

Duplicate Values and Final Dataset: CICIDS2017 dataset checked for inconsistency and duplicate value. CICIDS2017 dataset is free from duplication and final dataset is shown in figure 4.9 below.

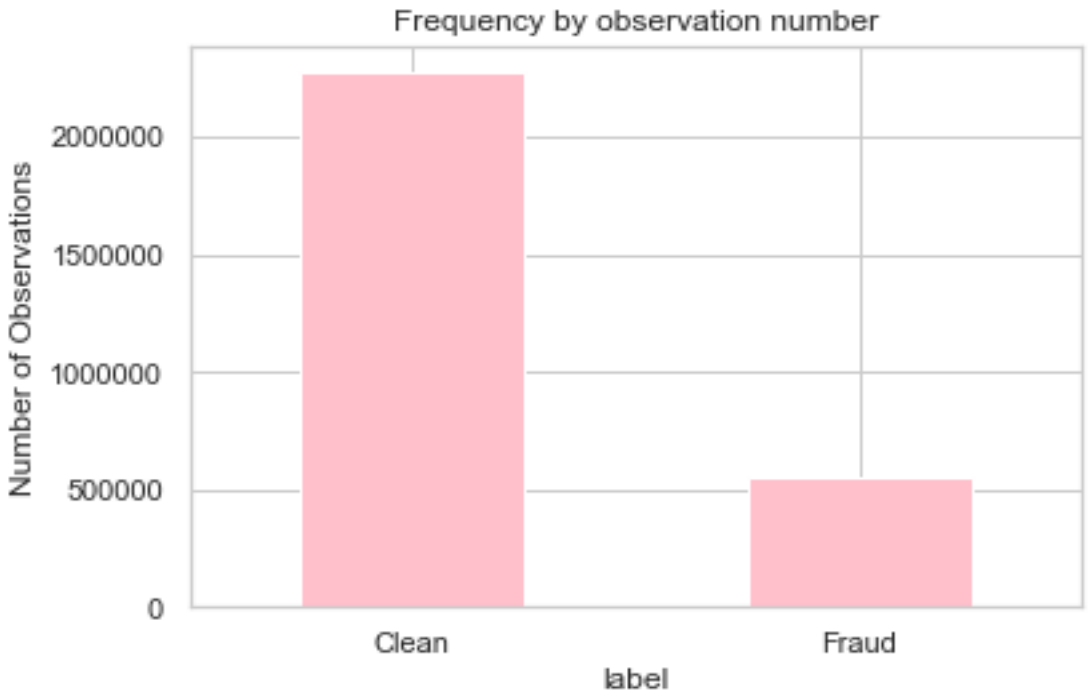


Figure 4.9 Final Dataset After preprocessing

Training/Validation/Test Dataset Split Result

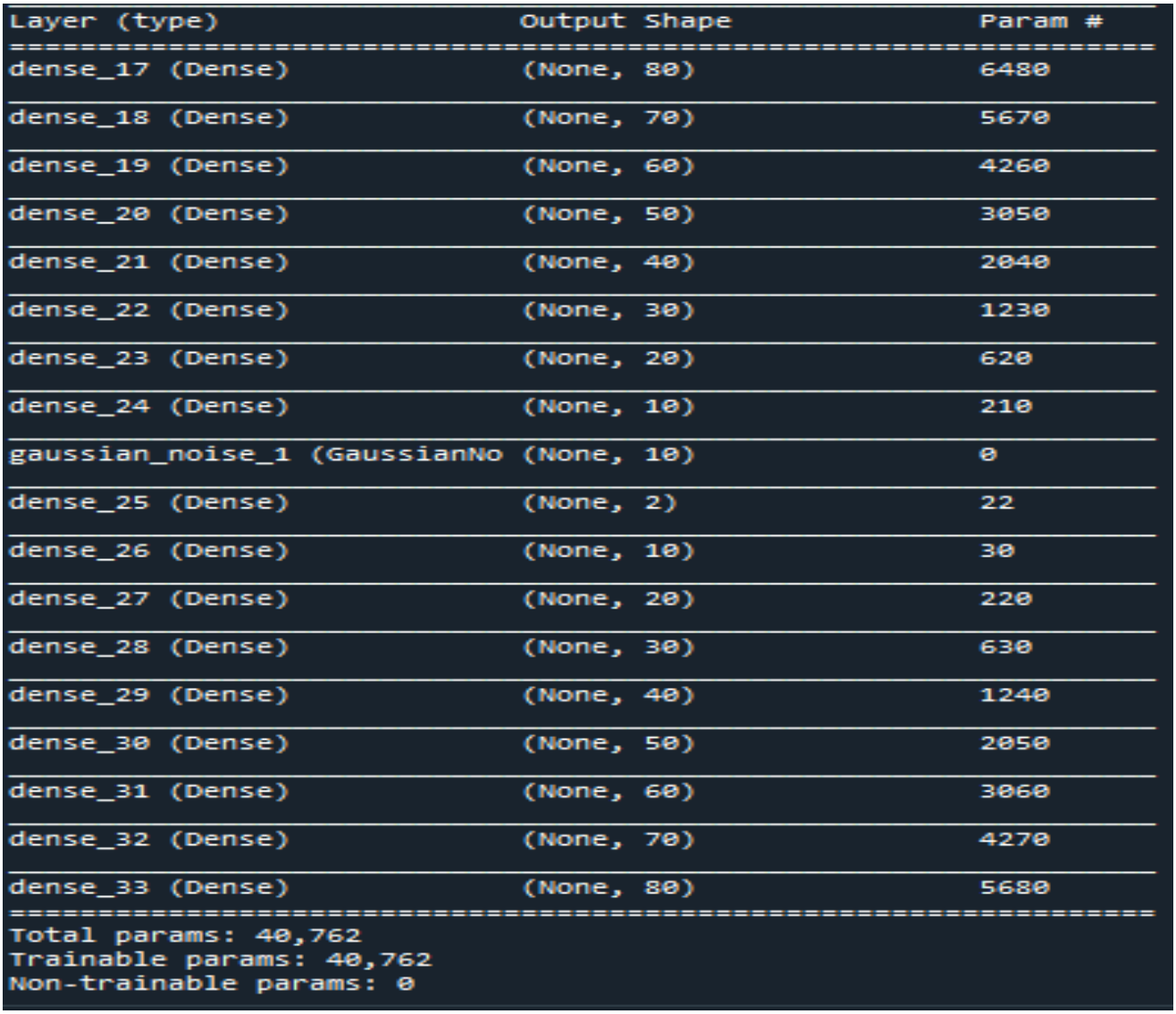
The original dataset CICIDS 2017 was split into three separate data frame, those are training datasets which contains only normal network traffic and it covers 60 percent of normal network flow. Learning algorithm train on training dataset and build a model. The developed model parameters need to be tuned to get optimum value, to do that validation dataset used which covers 20 percent of the total normal network flow. Finally testing dataset used to evaluate the performance of model. Testing dataset contains 20 percent normal network traffic which was not used for training and validation purpose plus the entire attack network flow which covers 20 percent of the entire dataset also not seen by the learning algorithm before.

```
In [21]: X_train, X_validate = train_test_split(X_train,
    ...:                                        test_size=VALIDATE_SIZE,
    ...:                                        random_state=RANDOM_SEED)

In [22]: X_train.shape
Out[22]: (1440000, 80)

In [23]: X_validate.shape
Out[23]: (360000, 80)

In [24]: X_test.shape
Out[24]: (1028665, 81)
```

Figure 4.10 Dataset Split

4.2 Model Development

Denoising Autoencoder used for this thesis is built from 80 neurons of an input and output layer respectively. Seven hidden layers include for both encoding and decoding end, each layer contains several neurons. Number of neuron decrease while the depth of the hidden layer increases because Denoising Autoencoder reduce the input to representative and significant data representation. A Gaussian layer is applied just right before the bottleneck layer which helps to randomly corrupt inputs. The bottleneck layer contains only two neurons which except compress down the input. Decoder end of Denoising Autoencoder hidden layer takes the compressed representation of data and recovers the original data with minimum reconstruction loss. Finally, the last layer of the decoder decodes the encoded data. Each layer uses Exponential Linear Unit (ELU) activation function and Adaptive Moment Estimation (Adam) optimizer. After trial and error learning rate of this research is 10e-7. DAE learn features relationship and develop model, any deviation from this model is considered as an outlier. For 100 epochs and a batch size of 256, the DAE is trained on the training dataset, tune the parameter of the model until optimal was found using validation dataset and the final performance of the developed model is tested against the test dataset. The system monitors loss with patience for 100 times at the end of every epoch. Training terminated if the loss is not reducing for 100 consecutive epochs. Figure 4.11 shows the Denoising Autoencoder built this study.



Layer (type)	Output Shape	Param #
dense_17 (Dense)	(None, 80)	6480
dense_18 (Dense)	(None, 70)	5670
dense_19 (Dense)	(None, 60)	4260
dense_20 (Dense)	(None, 50)	3050
dense_21 (Dense)	(None, 40)	2040
dense_22 (Dense)	(None, 30)	1230
dense_23 (Dense)	(None, 20)	620
dense_24 (Dense)	(None, 10)	210
gaussian_noise_1 (GaussianNo	(None, 10)	0
dense_25 (Dense)	(None, 2)	22
dense_26 (Dense)	(None, 10)	30
dense_27 (Dense)	(None, 20)	220
dense_28 (Dense)	(None, 30)	630
dense_29 (Dense)	(None, 40)	1240
dense_30 (Dense)	(None, 50)	2050
dense_31 (Dense)	(None, 60)	3060
dense_32 (Dense)	(None, 70)	4270
dense_33 (Dense)	(None, 80)	5680

Total params: 40,762
Trainable params: 40,762
Non-trainable params: 0

Figure 4.11 Denoising Autoencoder Developed in Python

Hyperparameters used in the Model: Hyperparameters are the variables that determines the network structure such as number of hidden units and the variables, which determine how the network trained such as learning rate. Hyperparameters are set before training or before optimizing the weights and bias.

Number of Hidden Layers and units: Hidden layers are the layers between input layer and output layer**.** In this study seven hidden layers which each layers have different number of neurons, each

neuron connected to the neuron of next layer neurons and perform some transformation on input**.** Them more the layer hidden layer increase while the neurons decrease the dataset compressed.

Activation function: ELU Activation functions used to introduce nonlinearity to models, which allows deep learning models to learn nonlinear prediction boundaries. ELU activation function is differentiable at all points, ideal for noise and large data and requires a little memory.

Learning Rate: The learning rate defines how quickly a network updates its parameters. Low learning rate slows down the learning process but converges smoothly. Larger learning rate speeds up the learning but may not converge. Based on this the optimal learning rate used by this study is 10Exp-7**.**

Number of epochs: The number of times to train on training data tune Hyperparametes and develop model. The study increases number of epochs until accuracy of the model stops decreasing and loss of the model stops decreasing. The epochs of this, study is 100.

Batch size: Batch size is the number of sub samples given to the network after which parameter update happens. A good default for batch size might be 32, 64, 128, 256, and so on. In this study batch, size is 256.

Accuracy: As shown in figure 4.12 the accuracy of the model with both the validation dataset and test dataset is shown below in the figure 30. Model accuracy for both is 99.98% which is close to 1. Total number of correct identification for attack and benign data 99.98% was right.

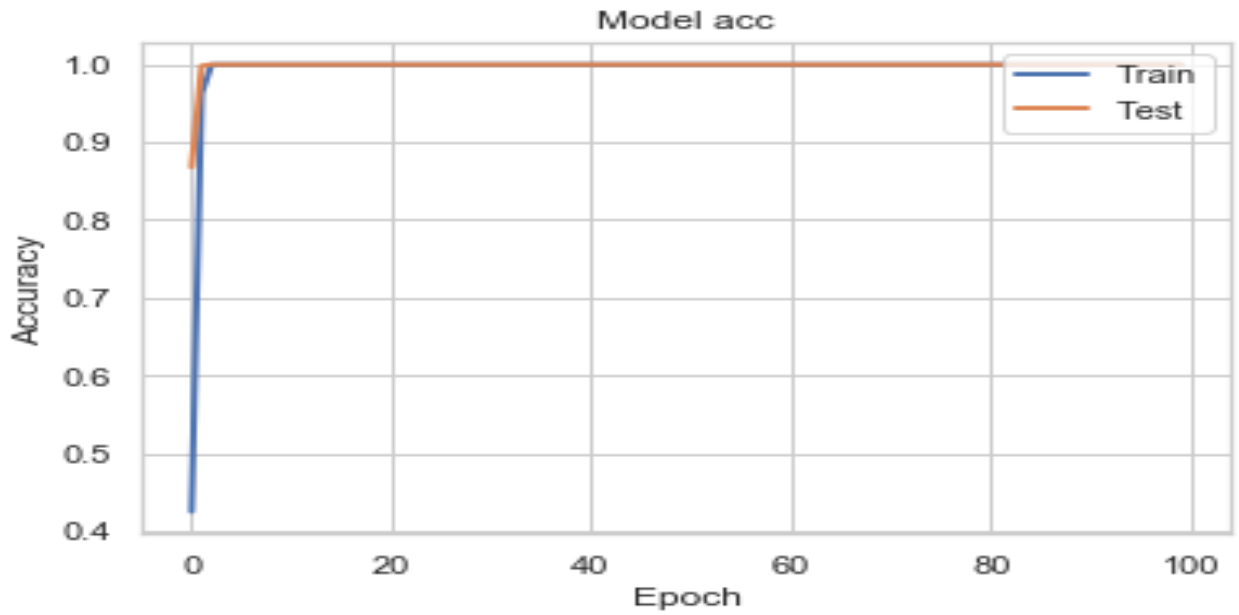


Figure 4.12 Model Accuracy

Loss: Loos of the developed model is measured on both validation dataset and test dataset. Loss of the model continuously decreases while the model train repeatedly as epoch increase. The loss of model shown below in figure 4.13.

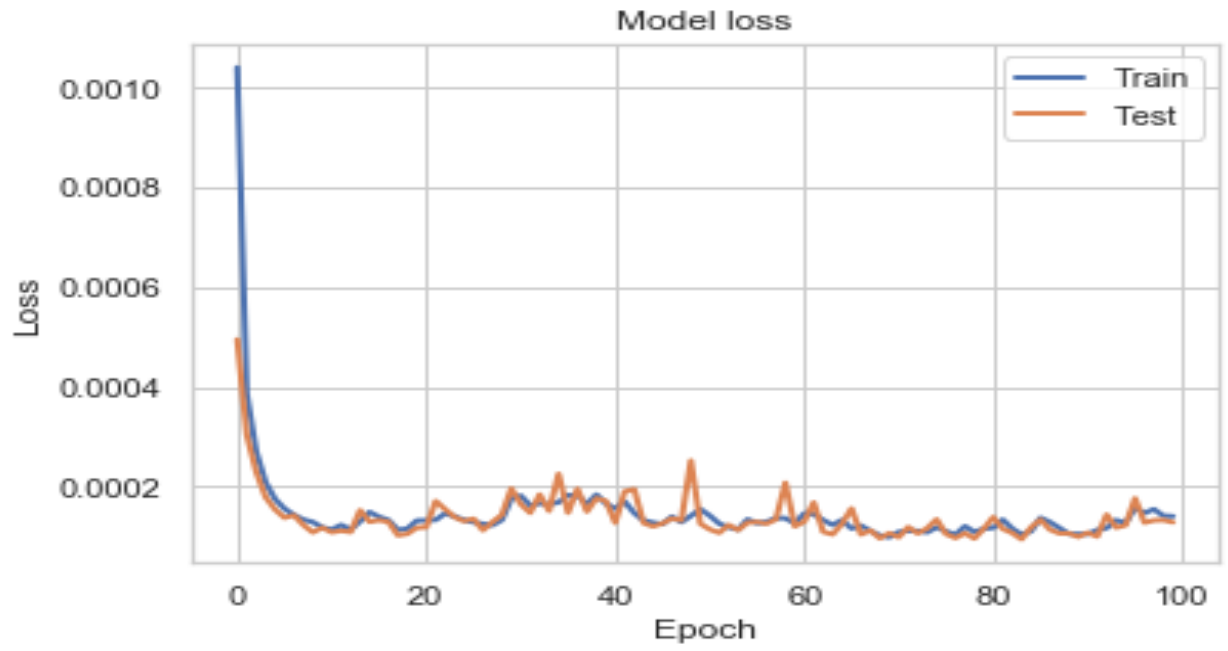


Figure 4.13 Model Loss

Reconstruction error: Reconstruction error or Loss function result is, shows how well the model recovers the original dataset from compressed representation and the error is very small. Reconstruction error is shown in figure 4.14, 4.15,4 and 416.

```
In [40]: X_test_transformed = pipeline.transform(X_test)

In [41]: reconstructions = autoencoder.predict(X_test_transformed)

In [42]: X_test_predictions = autoencoder.predict(X_test_transformed)
    ...: mse= np.mean(np.power(X_test_transformed - reconstructions,
2), axis=1)
    ...: error_df = pd.DataFrame({'recostruction_error': mse,
'True_class':y_test})
    ...: error_df.describe()
Out[42]:
       recostruction_error    True_class
count         1.028665e+06  1.028665e+06
mean          4.555693e-03  5.404840e-01
std           9.780317e-03  4.983586e-01
min           2.609015e-05  0.000000e+00
25%           1.048912e-04  0.000000e+00
50%           2.940152e-04  1.000000e+00
75%           2.105418e-03  1.000000e+00
max           1.143217e-01  1.000000e+00
```

Figure 4.14 Reconstruction Error of The Model

```
Mean Squared Error reconstruction losses for 100 clean transactions:
[3.33679038e-05 4.95636592e-05 4.40110210e-05 4.08718122e-04
 3.12512407e-05 1.53993792e-03 3.48139045e-05 3.14105547e-05
 3.12756268e-05 3.30346902e-05 3.49621721e-05 4.10158446e-05
 4.82950239e-05 1.53868405e-04 1.75924968e-04 5.16361859e-03
 4.93481885e-05 6.39464577e-04 3.36935434e-05 4.47510469e-05
 3.48749857e-05 3.47802951e-05 8.54134348e-05 2.97771783e-05
 3.01710265e-05 4.47574009e-05 4.73815214e-04 7.61980290e-05
 2.42862639e-03 7.42372141e-04 1.92845761e-03 2.65807394e-04
 8.26224059e-05 3.53677492e-05 2.02256463e-04 3.21418638e-05
 5.84528891e-05 1.49965362e-03 1.38726345e-04 3.38795122e-05
 6.00863692e-03 9.31407600e-05 7.84539735e-05 2.16666703e-04
 5.33310670e-04 1.07248148e-04 9.96632758e-04 3.60134881e-05
 1.17203880e-03 4.60199715e-04 6.38302499e-04 2.77527317e-04
 3.52878915e-05 2.84467975e-05 1.59708935e-03 1.70363258e-03
 1.20832065e-03 3.56119156e-04 3.44036475e-05 4.22531110e-05
 3.55665825e-05 3.43231754e-05 7.05700022e-04 5.94259213e-04
 2.75330339e-04 1.20030384e-04 4.19982351e-05 1.14488511e-04
 2.88016825e-05 9.86909963e-04 6.72323633e-05 2.37570330e-04
 6.59844460e-04 2.88412382e-04 3.62891698e-04 5.17841422e-04
 2.05282272e-04 2.18074148e-04 1.77981422e-03 3.36524619e-05
 3.25312868e-05 5.31455883e-04 1.06916791e-04 5.09498750e-04
 3.70021518e-05 4.70166929e-05 1.62097527e-04 3.06476060e-05
 3.81173170e-05 4.78413989e-03 1.23693920e-04 4.82419786e-04
 4.33219776e-04 3.16965624e-05 1.10214091e-03 2.17005688e-04
 3.06805336e-05 1.23056883e-03 2.82989321e-05 1.62460179e-04]
```

Figure 4.15 Loss function result for 100 random taken normal network flow.

```
Mean Squared Error reconstruction losses for 100 fraudulent transaction
[6.36947691e-04 1.08948509e-04 3.55643755e-03 1.24143240e-02
 3.87074571e-04 6.98719477e-04 1.29868572e-04 1.20817673e-03
 6.90875770e-04 9.04692988e-04 1.86695899e-04 1.83855010e-03
 9.80050608e-05 1.27747414e-03 3.08174813e-02 1.09788286e-02
 2.25606855e-03 3.09475058e-04 1.63802759e-04 1.21119777e-02
 9.90732082e-05 1.30117959e-02 4.56426731e-03 1.12186718e-02
 1.04477747e-04 1.70585202e-04 1.33743497e-02 1.19869221e-02
 1.50079200e-02 1.21186009e-03 1.33422190e-04 1.23175457e-04
 3.93001846e-02 1.05675201e-03 3.41319738e-04 1.11638556e-02
 3.09552476e-03 1.82179683e-04 1.07277474e-02 1.80674718e-03
 1.47302351e-02 1.28655615e-03 1.01774593e-03 2.18298717e-02
 3.55180062e-02 5.30279298e-03 1.37803354e-02 1.57863331e-04
 1.21002866e-03 1.41965527e-02 1.66062092e-04 2.66321157e-04
 8.29037747e-04 9.94635572e-05 1.22041540e-02 2.85892567e-04
 1.10971114e-04 8.45991516e-04 1.78645041e-02 1.84103795e-02
 3.56465658e-02 1.80671119e-04 1.04004942e-04 1.14626645e-02
 7.18393343e-04 9.98735150e-05 1.20281281e-04 7.44722128e-04
 1.70216348e-03 8.43683683e-04 3.04676903e-02 1.57905283e-04
 1.51681828e-04 1.15717593e-03 1.60761430e-02 2.17293195e-04
 1.54307372e-03 1.65773705e-03 9.44405757e-03 1.36148201e-02
 1.12958825e-02 1.06215572e-04 1.48014542e-04 2.20407576e-04
 2.03491779e-02 1.63729264e-04 1.13385361e-03 3.27864488e-02
 1.32818773e-02 2.12192659e-04 1.29959778e-02 1.15203606e-02
 1.57978362e-03 1.50888400e-02 8.13483503e-03 1.23850817e-02
 9.99359804e-05 1.17318069e-04 1.07902215e-04 3.53403526e-04]
```

Figure 4.16 Loss function result for 100 random taken attack network flow

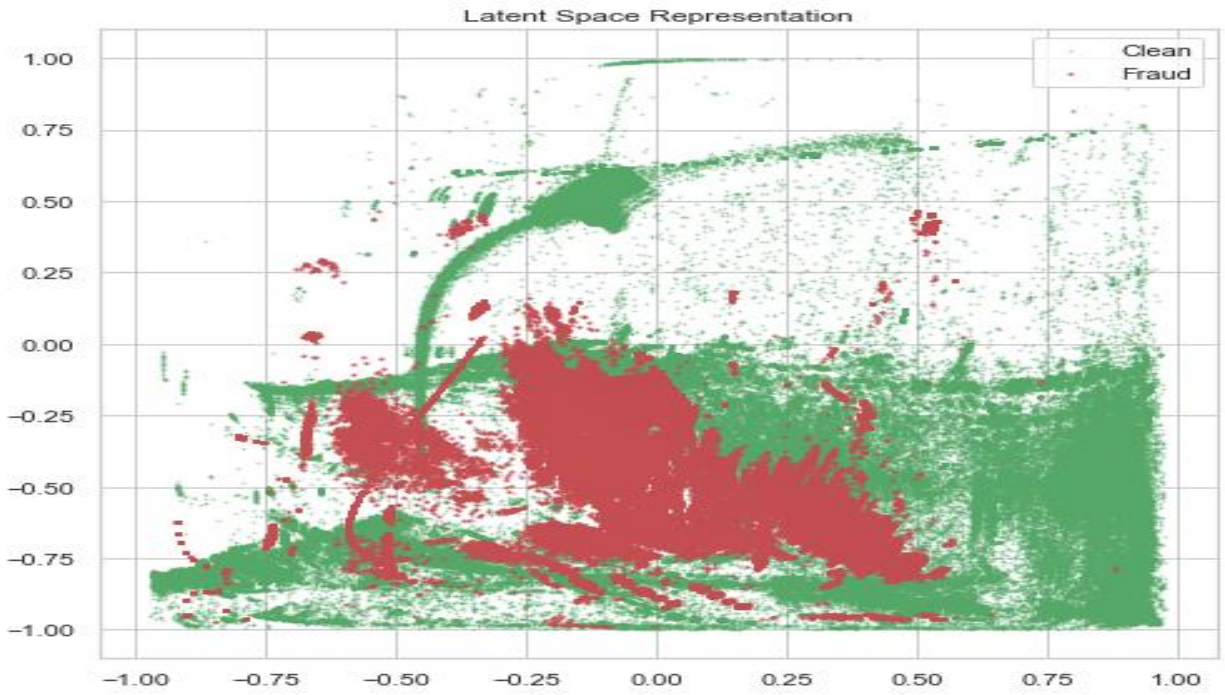


Figure 4.17 Visualization of latent or bottleneck layer with a cluster of the network flow

Area Under Curve-Receiver Operating Characteristic Curve: area under curve indicates the performance of the system which fail to distinguish attack and normal network flows. Area under curve is 86.1%. Receiver operating characteristic curve is 99.9% which shows the performance of the model at all classification threshold. Figure 4.18 shows AUC-ROC curve.

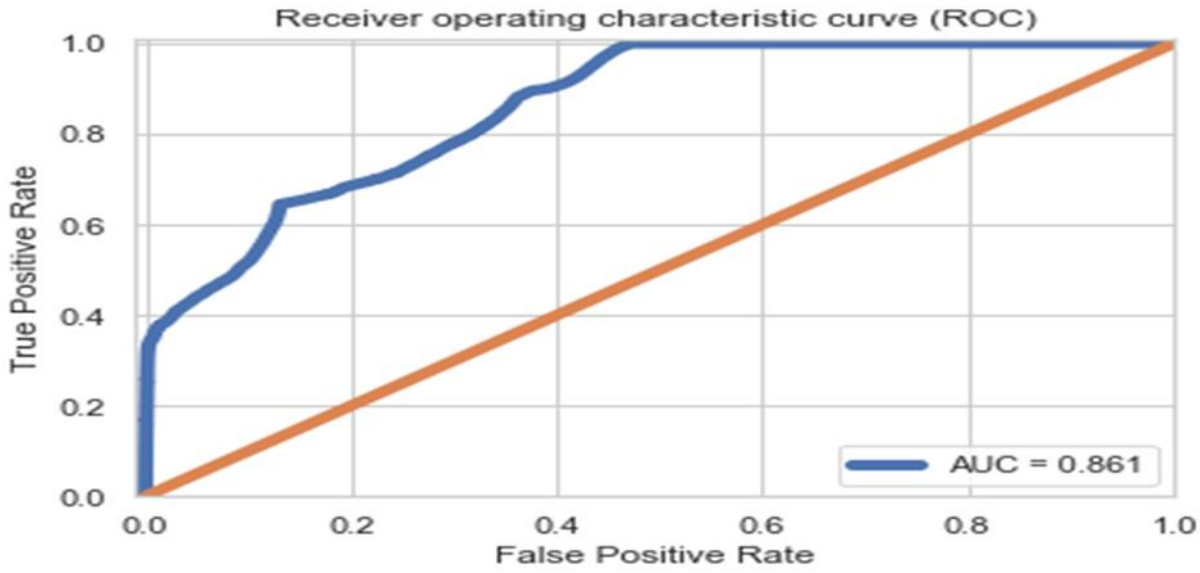


Figure 4.18 AUC-ROC of Model

CHAPTER FIVE
RESULTS AND DISCUSSIONs

There are numerous research methodologies has been proposed to solve the cyber security threats. In this chapter finding of some researches compared with the experiment conducted in this study. These research work which were done by other researchers uses the same dataset but apply different algorithm and methodologies. This study tried to provide a new approach to ward flow-based anomaly intrusion detection system. There is a number of metrics used for comparison of this study such as Dataset, Algorithm used, Accuracy, Model Loss, Reconstruction Error, AUC-ROC, and training time of the model. Table 5.1 summarizes different research work findings and approach being used with this study work.

In figure 4.11 Denoising Autoencoder consisted eight input and output layer, a hidden layer and a Gaussian layer. DAE train on the training dataset and build model. Any deviation from this model is considered as attack network flow. The developed DAE in this study has more than one hidden layers which makes it deep neural network. Gaussian layer used to intentionally corrupt input to avoid identity function and overfitting. From the nature of the DAE the proposed model was not vulnerable to overfitting.

The experiment result of this study is very promising. The mode accuracy acquired from other researcher is very high, the record shows from 86.1%-99.8% accuracy. The accuracy of the proposed model, flow-based anomaly intrusion detection shown in figure 4.12 evaluated for both on test dataset and validation dataset was 99.98% which is even better. The more accuracy high the better the model identifies the attack or outliers from normal network traffic.

Loss of developed model is tested against validation dataset and test dataset, as show in figure 4.13 while the model train on dataset for 100 epoch loss of the system getting minimized both on validation and test dataset, which is recorded bellow 0.0002. This shows the more the model train on dataset, flow-based anomaly intrusion detection finds an optimal value for parameters and minimize loss. Even though none of the mentioned research work didn’t present loss of model to compare with, the proposed model still shows very small system loss.

The proposed model used Denoising Autoencoder, which recovers the original input from intentionally corrupted input and compressed representation experiment result shows very small range from 1.05exp-4 - 1.14exp-1. The experiment also shows in figure 4.14,4.15, and 4.16 that reconstruction error for random one hundred normal network flow and attack network flow which is very small. Reconstruction error small means, even dataset reduction performed by Denoising Autoencoder still the relationship between feature is kept and the dataset is represented perfectly which makes increase the performance of the proposed model.

Area under curve indicates the performance of the system which fail to distinguish attack and normal network flows. Area under curve result find by other researcher was 75.96% to 96.12% high while the proposed model was 86.1%. Even the proposed model performance shows it failed to distinguish attack from normal network flows but it still better than some of the research result found. AUC shows in figure 4.18.

Receiver operating characteristics curve shows the performance of a model with a true positive rate against the false-positive rate at all possible threshold. The result of the experiment by other research work is range from 75.65-99.99%. The ROC of proposed model is 99.98 % which high and almost equal to finding of other research work. The high ROC imply that false positive rate is highly decreased even though it didn't show by how much. ROC shows in figure 4.18.

The experiment result of training time of the proposed model shown in appendix A, which ranges from 178 micro seconds per sample up to 231 micro seconds per sample. other research findings not present training time to compare with, but it can be said that the proposed model learns fast on training dataset which helps to identify attack before the damage happened.

Latent space representation tells how well the model clusters the normal network flow from attack network flow in bottle neck layer of Denoising Autoencoder. From figure 4.17 the result of experiment shows the model well the network traffic to their category.

Table 5.1: Literature Review Different IDS Use CICIDS2017 Dataset

Reference paper	**Purpose of IDS**	**Dataset**	**Algorithm**	**Result**	
V. Kumar t.al[98]	Detecting Intrusions and Attacks in the Network Traffic using Anomaly based Techniques	CICIDS2017	• Naïve Bayes • ID3 • Ensemble Learning • MLP	**ML Algorithm**	**Accuracy**
				Naïve Bayes	81.65%
				ID3	95.15%
				Ensemble Learning	89.1%
				MLP	83.87%
Y.Zhan.et.al[99]	Deep hierarchical network, which integrates the improved LeNet-5 and LSTM neural network structures	CICIDS2017	Improved LeNet-5 and LSTM neural network	Accuracy = 99.81%	
N.Marir.et.al[100]	Unsupervised Feature Learning with Distributed Stacked Denoising Sparse Autoencoder for Abnormal Behavior Detection Using Apache Spark	CICIDS2017	Distributed Stacked Denoising Sparse Autoencoder	AUR = 96.12%	
S. Zavrak et.al[101]	Anomaly-Based Intrusion Detection From Network Flow Features Using Vibrational Autoencoder	CICIDS2017	Vibrational Autoencoder	AUC=75.96%	

CHAPTER SIX
CONCLUSIONs AND RECOMMENDATIONs

6.1 Conclusions

In this study, deep neural network based flow-based anomaly intrusion detection system is proposed. The study aims to detecting an anomaly in networks flows by using Denoising Autoencoder algorithm to develop model. The proposed system used CICIDS2017 dataset to learn complex feature and their relationship's, reduced the high dimension dataset. CICIDS2017 dataset contain information about the behavior of connection rather than the payload of packet. CICIDS2017 contains over eighty features, up-to date divers attack network flow and normal network flow traffic. CCICDS2017 network flow dataset feed to Denoising Autoencoder. Denoising Autoencoder learn the normal network behavior and build model, uses this model to identify attack network traffic flow. To know how well the proposed model, perform different performance measurement metrics is used and finally the performance of the proposed model is compared to other research work which uses the same dataset.

The proposed flow-based anomaly detection system has high detection accuracy 99.98%, very small model loss and reconstruction error which is almost zero including both normal network flow and attack network flow, the performance of the model in distinguishing between normal and attack network traffic was measured using Area Under Curve (AUC) 86.1% and Receiver Operating Characteristic Curve (ROC)99.98%, which apply false positive rate was decreased. latent space representation was encouraging even though there is no clear cluster most of the attack and normal network flow appear to be neatly grouped together. Training time spent by model was very small, ranges from 178 micros second per sample up to 231 micros second per sample which helps the proposed model to build model quickly and identify the attack before damage happens.

Finally, the model performance was compared against other research work and better detection performance capability, high accuracy, ROC, model loss and reconstruction error, with good latent space representation, low training time and minimum false positive rates with medium AUC. Based on the performance of flow-based anomaly intrusion detection system it can be applied for online network security purpose to identify anomalies or attacks.

Drawback of the proposed mode: the proposed model didn't show how much the false positive rate decreased, even though DAE perform dimensional reduction it didn't show by how much and other performance measure were not used.

6.2 Recommendations

It is recommended that the proposed model can be further enhanced if the following upgrading can be achieved. First if hybrid deep learning model is applied on advanced hardware. Second more focus paid into decreasing false alarm rate and by how much finally identifying and optimizing different issues that affect its performance and use variety of performance measure.

REFERENCES

[1] ITU Publications, “Global Cybersecurity Index (GCI) 2018”, ITU Publications, pp 1-92, 2019.

[2] A. A. Aburomman and M. B. I. Reaz, “A novel svm-knn-pso ensemble method for intrusion detection system,” Applied Soft Computing, vol. 38, pp. 360–372, 2016.

[3] Danny Plammer,” Cybercrime drains $600 billion a year from the global economy”, February 21-2018, july 2 2019 ,https://www.zdnet.com/article/cybercrime-drains-600-billion-a-year-from-the-global-economy-says-report/ .

[4] Abdulla Amin Aburomman and Mamun Bin Ibne Reaz ,” Survey of learning methods in intrusion detection Systems”, International Conference on Advances in Electrical, Electronic and System Engineering, pp. 362-365, 2016.

[5] S.Visalakshi,v.Radha, “A Literature Review of Feature Selection Techniques and Applications” 2014.

[6] Hebatallah Mostafa Anwer, Mohamed Farouk ,Ayman Abdel-Hamid “A Framework for Efficient Network Anomaly Intrusion Detection with Features Selection” , 9th International Conference on Information and Communication Systems (ICICS), pp.157-162, 2018.

[7] Muder Almi'ani *, Member, IEEE, Alia Abu Ghazleh+, Member, IEEE, Amer Al-Rahayfeh†, Member,IEEE, Abdul Razaque‡, Member, IEEE,” Intelligent Intrusion Detection System Using Clustered Self Organized Map”, Fifth International Conference on Software Defined Systems (SDS), pp. 138-145, 2018.

[8] Aysel gul, Asref adal ,” A Feature Selection Algorithm For IDS ”, Second International Conference on Computer Science and Engineering , pp.816-920, 2017. (LIACS), pp.3-32, 2017.

[9] Sufyan T. Faraj Al-Janabi and Hadeel Amjed Saeed,” A Neural Network Based Anomaly Intrusion Detection System”, Developments in E-systems Engineering, pp. 221-225, 2011.

[10] Apichit Pattawaro ,Chantri Polprasert,” Anomaly-Based Network Intrusion Detection System through Feature Selection and Hybrid Machine Learning Technique”, Sixteenth International Conference on ICT and Knowledge Engineering, pp. ,2018.

[11] Nguyen Thanh Van, Tran Ngoc Thinh, Le Thanh Sach,” An anomaly-based Network Intrusion Detection System using Deep learning”, International Conference on System Science and Engineering (ICSSE), pp. 210-214, 2017.

[12] Kapil Wankhade, Sadia Patka,” An Efficient Approach for Intrusion Detection Using Data Mining Methods” Pp. 1615-1618, 2013.

[13] Luis Miguel Torres, Eduardo Maga˜na, Mikel Izal and Daniel Morat´o ,Guzm´an Santaf´e,” An anomaly-based intrusion detection system for IEEE 802.11 networks”, 2010.

[14] Amreen Sultana , M.A.Jabbar,” Intelligent Network Intrusion Detection System using Data Mining Techniques”, pp. 328-333, 2016.

[15] S. Alexander, "An anomaly intrusion detection system based on intelligent user recognition", Ph.D. Thesis, Faculty of Information Technology, University of Jyväskylä, Finland, 2002.

[16] S. Mansour and A. Sha'bani, "Fast neural intrusion detection System Based on Hidden Weight Optimization Algorithm and Feature Selection", World Applied Sciences Journal, No. 7 (Special Issue of Computer & IT), pp. 45-53, 2009.

[17] Guohang Yin ,Youran Zhang,Ziyi Zhao, “A novel computer network intrusion detection Algorithm based on OSVM and context validation”,pp. 591-595,2016.

[18] Barghi, M. N., Hosseinkhani, J., & Keikhaee, S., “An Effective Web Mining-based Approach to Improve the Detection of Alerts in Intrusion Detection Systems”, International Journal of Advanced Computer Science and Information Technology (IJACSIT),(ELVEDIT), Vol. 4, Iss. 1, Pp. 38-45, 2015.

[19] E. Biermann, E.Cloete , L.M.Venter A comparison of Intrusion Detection systems, Computers & Security, 20 , pp. 676-683, 2001.

[20] Keisuke Kato∗ and Vitaly Klyuev,” Development of a Network Intrusion Detection System Using Apache Hadoop and Spark”, pp. 416-423,2017.

[21] Raghavendra Chalapathy, Sanjay Chawla “Deep Learning for Anomaly Detection: a survey,” ResearchGate, pp. 1–47, 2019.

[22] Masaaki Sato, Hirofumi Yamaki and Hiroki Takakura,” Unknown Attacks Detection Using Feature Extraction from Anomaly-based IDS Alerts”, International symposium on Applications and Internet, pp. 263-267, 2012.

[23] Ammar Alazab , Michael Hobbs, Jemal Abawajy, Moutaz Alazab,” Using Feature selection for intrusion detection system ”, international symposium on communication and technologies , pp286-301. , 2012.

[24] Monther Aldwairi and Muneer Bani Yassein ,” Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model” , ResearchGate, pp 2-10 , 2017.

[25] Yanjie Zhao,” Network Intrusion Detection System Model Based on Data Mining ”, Weifan univesty , pp. , 2016.

[26] Li Tian1, Wang Jianwen1,” Research on Network Intrusion Detection System Based on Improved K-means Clustering Algorithm” International Conference on Advanced Computing and Communication Systems, pp 76-79. , 2009.

[27] Akash Garg and Prachi Maheshwari,” Identifying Anomalies in Network Traffic using Hybrid Intrusion Detection System”, International Conference on Advanced Computing and Communication Systems , pp. , 2016.

[28] JIA Zhanpei, SHEN Chao, YI Xiao, CHEN Yufei, YU Tianwen, GUAN Xiaohong, ”Big-Data Analysis of Multi-Source Logs for Anomaly Detection on Network-based System” , 13th IEEE Conference on Automation Science and Engineering, pp1136-1141. , 2017.

[29] Choi, Huy Anh Nguyen and Deokjai, Application of Data Mining to Network Intrusion Detection: Classifier Selection Model, Chonnam National University, Computer Science Department, 300 Yongbong-dong, Buk-ku, Gwangju 500-757, Korea.

[30] Mohammadreza Ektefa, Sara Memar, Fatimah Sidi, Department of IS,Lilly Suriani Affendey

” Intrusion Detection Using Data Mining Techniques”, IEEE, pp. 200-203, 2010.

[31] Zuech, R., T.M. Khoshgoftaar, and R. Wald,” Intrusion detection and big heterogeneous data: a survey. Journal of Big Data”, springer: p. 3, 2015.

[32] Rataha Samirn and D Vasynati” Review on Anomaly based Network Intrusion Detection System”, International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT): p.141-147, 2017.

[33] A. Sheik Abdullah, C. Ramya, V. Priyadharsini, C. Reshma,s.Selvakumar,” A Survey on Evolutionary Techniques for Feature Selection”, IEEE Conference on Emerging Devices and Smart Systems (ICEDSS):pp 58-62 , 2017.

[34] C. F. Tsai, et al., "Intrusion detection by machine learning: A review," Expert Systems with Applications, vol. 36, pp. 11994-12000, 2009.

[35] S. Das, "Filters, wrappers and a boosting-based hybrid for feature selection," in: Proc. 18th International Conference on Machine Learning (ICML-2001), San Francisco, CA, USA, Morgan Kaufmann, pp. 74–81, 2001.

[36] J. M. Cadenas, M. C. Garrido, and R. Martínez, "Feature subset selection Filter–Wrapper based on low quality data," Expert Systems with Applications, vol. 40, pp. 6241–6252, 2013.

[37] I. S. Oh, J. S. Lee, and B. R. Moon, "Hybrid genetic algorithms for feature selection," IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, no. 11, pp. 1424–1437, 2004.

[38] S. I. Ali and W. Shahzad, "A Feature Subset Selection Method based on Conditional Mutual Information and Ant Colony Optimization," International Journal of Computer Applications, vol. 60, no. 11, pp. 5–10, 2012

[39] A. G. K. Janecek and G. F. Gansterer et al, “On the Relationship between Feature Selection and Classification Accuracy”, In: Proceeding of New Challenges for Feature Selection, pp. 40-105, 2008.

[40] Samina Khalid, Tehmina Khalil, Shamila Nasreen, "A Survey of Feature Selection and Feature Extraction Techniques in Machine Learning", Science and Information Conference: pp 372-378, 2014.

[41] Sheraz, Yasir, shehzad Khalid, muhammad khawar bashir,jihun han, muhammad munwar iqbal , and kijun han, "Enhanced Network Anomaly Detection Based on Deep Neural Networks", Researchgate : pp -1-15, 2018.

[42] S. Latha Assistant, Dr. Sinthu Janita Prakash, "A Survey on Network Attacks and Intrusion Detection Systems", International Conference on Advanced Computing and Communication Systems (ICACCS), pp, 2017.

[43] A. Midzic, Z. Avdagic and S. Omanovic, "Intrusion Detection System Modeling Based on

Neural Networks and Fuzzy Logic", INES 2016 • 20th Jubilee IEEE International Conference on Intelligent Engineering Systems: pp 198-194, 2016

[44] Crothers M., Implementing Intrusion Detection Systems, a Hands-On Guide for Securing the Network, USA, 2002.

[45] Rahul Vigneswaran K∗, Vinayakumar R†, Soman KP† and Prabaharan Poornachandran, "Evaluating Shallow and Deep Neural Networks for Network Intrusion Detection Systems in Cyber Security" IEEE, pp , 2018.

[46] Huang Yi, Sun Shiyu, Duan Xiusheng, Chen Zhigang, "A Study on Deep Neural Networks Framework "IEEE,pp 1519-1522, 2016.

[47] Zheng Wang, "Deep Learning-Based Intrusion Detection with Adversaries", IEEE Access, pp 38367-38383, 2018.

[48] Ajay Shrestha and Susif Mahmood, "Review of Deep Learning Algorithms and Architectures ", IEEE Access, pp 53040-53065, 2019.

[49] Chuanlong Yin , Yuefei Zhu, Jinlong Fei, and Xinzheng He" A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks",IEEE Acess,pp 21954-21961,2017.

[50] Nathan Shone , Tran Nguyen Ngoc, Vu Dinh Phai , and Qi Shi, "A Deep Learning Approach to Network Intrusion Detection" , IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 1,PP 41-49, 2018.

[51] Ali Moradi Vartouni, Saeed Sedighian, Mohammad Teshnehlab, "An Anomaly Detection Method to Detect Web Attacks Using Stacked Auto-Encoder" IEEE, pp 131-133, 2018.

[52] McAfee Labs Threats Report, "McAfee Labs Threats Report 2019", McAfee Labs Threats Report, 99 1-44, 2019.

[53] Hashem Alaidaros ,Massudi Mahmuddin ,Ali Al-Mazari, "An Overview of Flow-Based and Packet- Based Intrusion Detection, Performance in High Speed Networks", semanticsholar.org, pp 324-331,2011.

[54] Anna Sperotto, Gregor Schaffrath, Ramin Sadre, Cristian Morariu, Aiko Pras and Burkhard Stiller, "An Overview of IP Flow-Based Intrusion Detection" IEEE communications surveys & tutorials, vol. 12, no. 3, pp 346-353, 2010.

[55] Rohini Sharma,Ajay Guleria,R.K. Singla," An overview of flow-based anomaly detection", Int. J. Communication Networks and Distributed Systems, Vol. 21, No. 2, pp 220-242, 2018.

[56] Cisco.com,"Cisco IOS NetFlow Configuration Guide", Release 12.4 available at http://www.cisco.com, Sept. 2010.

[57] J. Quittek, T. Zseby, B. Claise, and S. Zander," Requirements for IP Flow Information Export (IPFIX)", RFC 3917 (Informational).

[58] S. Leinen, "Evaluation of Candidate Protocols for IP Flow Information Export (IPFIX)," RFC 3955 (Informational), Jul 2008.

[59] Aiyung-Sup Kim, Hun-Jeong Kang, Seong-Cheol Hang, Seung-Hwa Chung. And James W Hang," A Flow-based Method for Abnormal Network Traffic Detection" IEEE, PP 559-612, 2004.

[60] Laurens Hellemons, Luuk Hendriks, Rick Hofstede, Anna Sperotto, Ramin Sadre, and Aiko Pras, "SSHCure: A Flow-Based SSH Intrusion Detection System", IFIP International Federation for Information Processing 2012, pp 86-97, 2012.

[61] Aiyung-Sup Kim, Hun-Jeong Kang, Seong-Cheol Hang, Seung-Hwa Chung. And James W Hang, "A Flow-based Method for Abnormal Network Traffic Detection" IEEE, pp 599-612, 2004.

[62] Anukool Lakhina, Mark Crovella, Christophe Diot, "Detecting Distributed Attacks using Network-Wide Flow Traffic" Researchgate, pp 1-3, 2014.

[63] Anna Sperotto, Ramin Sadre, Frank van Vliet, Aiko Pras, "A Labeled Data Set For Flow-based Intrusion Detection", Researchgate, pp 1-13, 2009.

[64] Tomas Jirsik, Milan Cermak, Daniel Tovarnak, and Pavel Celeda, "Toward Stream-Based IP Flow Analysis", IEEE, pp 70-76, 2017.

[65] Yeung, D., Jin, S. and Wang, X. 'Covariance-matrix modeling and detecting various flooding attacks', IEEE transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, Vol. 37, No. 2, pp.157–169, 2007.

[66] Winter, P., Hermann, E. and Zeilinger, M. 'Inductive intrusion detection in flowbased network data using one-class support vector machines', 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pp.1–5, IEEE, 2011.

[67] Bhuyan, M.H., Bhattacharyya, D.K. and Kalita, J.K. 'NADO: network anomaly detection using outlier approach', Proceedings of the 2011 International Conference on Communication, Computing & Security, pp.531–536, ACM, 2011.

[68] Paredes-Oliva, I., Castell-Uroz, I., Barlet-Ros, P., Dimitropoulos, X. and Sole-Pareta, J. 'Practical anomaly detection based on classifying frequent traffic patterns', Conference on Computer Communications Workshops (INFOCOMWKSHPS), pp.49–54, IEEE, 2012.

[69] Jadidi, Z., Muthukkumarasamy, V. and Sithirasenan, E. 'Metaheuristic algorithms based flow anomaly detector', 19th Asia-Pacific Conference on Communications (APCC), pp.717–722, IEEE, 2013.

[70] Bhuyan, M.H., Bhattacharyya, D.K. and Kalita, J.K. 'A multi-step outlier-based anomaly detection approach to network-wide traffic', Information Sciences, Vol. 348, No. C, pp.243–271, 2016a.

[71] Abuadlla, Y., Kvascev, G., Gajin, S. and Jovanovic, Z. 'Flow-based anomaly intrusion detection system using two neural network stages', Computer Science and Information Systems, Vol. 11, No. 2, pp.601–622, 2014.

[72] Fernandes, G., Rodrigues, J. and Proenca, M.L. 'Autonomous profile-based anomaly detection system using principal component analysis and flow analysis', Applied Soft Computing, Vol. 34, No. C, pp.513–525, 2015.

[73] Ahmim, A. and Zine, N.G. 'A new hierarchical intrusion detection system based on a binary tree of classifiers', International Journal of Information & Computer Security, Vol. 23, No. 1, pp.31–57, 2015.

[74] Fernandes, G., Carvalho, L.F., Rodrigues, J.J. and Proenca, M.L. 'Network anomaly detection using IP flows with principal component analysis and ant colony optimization', Journal of Network and Computer Applications, Vol. 64, No. C, pp.1–11, 2016.

[75] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018.

[76] C.Y. J.Peng, M.Harwell, S.M.Liou, L. H.Ehman, Advances in missing data methods and implications for educational research, In S. Sawilowsky (Ed.), Real data analysis, pp 31−78, Greenwich, 2006.

[77] Shichao Zhang , Xindong Wu , Manlong Zhu, “Efficient Missing Data Imputation for Supervised Learning, “ In proceeding of: Proceedings of the 9th IEEE International Conference on Cognitive Informatics, ICCI 2010, Beijing China, 2010

[78] Julian Luengo, Salvador Garcia, Francisco Herrera, “On the choice of the best imputation methods for missing values considering three groups of classification methods”, Knowl Inf Syst in Springer-Verlag, London , 2011.

[79] R.B Kline, Principles and Practice of Structural Equation Modelling, Guilford Press, New York, 1998.

[80] Kurniabudi, Deris Stiawan, Darmawijoyo, Mohd Yazid Bin Idris,, Alwi M. Bamhdi ,and Rahmat Budiarto, “CICIDS-2017 Dataset Feature Analysis With Information Gain for Anomaly Detection”, IEEE,Volume 8, July 30, 2020,

[81] J. Zhai, S. Zhang, J. Chen and Q. He, "Autoencoder and Its Various Variants," 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Miyazaki, Japan, 2018, pp. 415-419.

[82] S. Naseer et al., "Enhanced Network Anomaly Detection Based on Deep Neural Networks," in IEEE Access, vol. 6, pp. 48231-48246, 2018.

[83] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016. [85] J. Gabela et al., "The Effect of Linear Approximation and Gaussian Noise Assumption in Multi-Sensor Positioning Through Experimental Evaluation," in IEEE Sensors Journal, vol. 19, no. 22, pp. 10719-10727, 15 Nov.15, 2019.

[84] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, ‘‘Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,’’ J. Mach. Learn. Res., vol. 11, no. 12, pp. 3371–3408, Dec. 2010. [Online]. Available: http://www.jmlr.org/papers/v11/vincent10a.html

[85] J. Gabela et al., "The Effect of Linear Approximation and Gaussian Noise Assumption in Multi-Sensor Positioning Through Experimental Evaluation," in IEEE Sensors Journal, vol. 19, no. 22, pp. 10719-10727, 15 Nov.15, 2019.

[86] B. Ding, H. Qian and J. Zhou, "Activation functions and their characteristics in deep neural networks," 2018 Chinese Control and Decision Conference (CCDC), Shenyang, 2018, pp. 1836-1841.

[87] Clevert, Djork-Arné & Unterthiner, Thomas & Hochreiter, Sepp. (2015). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). Under Review of ICLR2016 (1997).

[88] Ding, Bin & Qian, Huimin & Zhou, Jun. (2018). Activation functions and their characteristics in deep neural networks. 1836-1841.

[89] K. ADEM and S. KILIÇARSLAN, "Performance Analysis of Optimization Algorithms on Stacked Autoencoder," 2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Ankara, Turkey, 2019, pp. 1-4.

[90] R. N. Singarimbun, E. B. Nababan and O. S. Sitompul, "Adaptive Moment Estimation To Minimize Square Error In Backpropagation Algorithm," 2019 International Conference of Computer Science and Information Technology (ICoSNIKOM), Medan, Indonesia, 2019, pp. 1-7.

[91] Kurniabudi, Deris Stiawan, Darmawijoyo, Mohd Yazid Bin Idris,, Alwi M. Bamhdi ,and Rahmat Budiarto, "CICIDS-2017 Dataset Feature Analysis With Information Gain for Anomaly Detection", IEEE,Volume 8, July 30, 2020, [81] chollet, F &others. (2015).Keras. GitHub,https://github.com/fchollet/keras

[92] Swaroop C. H., July 2015, "A Byte of Python", CreateSpace Independent Publishing Platform 7290 Investment Drive B North Charleston SC United States.

[93] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy, https://www.nature.com/articles/s41586-020-2649-2

[94] I. Stančin and A. Jović, "An overview and comparison of free Python libraries for data mining and big data analysis," 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 2019

[95] Virtanen, Pauli & Gommers, Ralf & Oliphant, Travis & Haberland, Matt & Reddy, .etl. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods. 17. 1-12. 10.1038/s41592-019-0686-2.

[96] Houssem Ben Braiek and Foutse Khomh, "A TensorFlow Library for Detecting Training Issues in Neural Network Programs", 19th International Conference on Software Quality, Reliability and Security (QRS), IEEE, 2019

[97] 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.

[98] V. Kumar, V. Choudhary, V. Sahrawat and V. Kumar, "Detecting Intrusions and Attacks in the Network Traffic using Anomaly based Techniques," 2020 5th International Conference on Communication and Electronics Systems (ICCES), COIMBATORE, India, 2020, pp. 554-560,

[99] Y. Zhang, X. Chen, L. Jin, X. Wang and D. Guo, "Network Intrusion Detection: Based on Deep Hierarchical Network and Original Flow Data," in IEEE Access, vol. 7, pp. 37004-37016, 2019.

[100] N. Marir, H. Wang and G. Feng, "Unsupervised Feature Learning with Distributed Stacked Denoising Sparse Autoencoder for Abnormal Behavior Detection Using Apache Spark," 2019 IEEE 2nd International Conference on Knowledge Innovation and Invention (ICKII), Seoul, Korea (South), 2019, pp. 473-476.

[101] S. Zavrak and M. İskefiyeli, "Anomaly-Based Intrusion Detection from Network Flow Features Using Variational Autoencoder," in IEEE Access, vol. 8, pp. 108346-108358, 2020.

Appendix A
Python Code

```
import pandas as pd
import numpy as np
from scipy import stats
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, precision_recall_curve
from sklearn.metrics import recall_score, classification_report, auc, roc_curve
from sklearn.metrics import precision_recall_fscore_support, f1_score
from sklearn.preprocessing import StandardScaler,MinMaxScaler
from pylab import rcParams
from keras.models import Model, load_model
from keras.layers import Input, Dense
from keras.callbacks import ModelCheckpoint, TensorBoard
from keras import regularizers
#from datetime import datetime
from sklearn import preprocessing
import copy
import binascii
import ipaddress
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential, load_model
from tensorflow.python.framework import ops
import datetime
import random as rn
from sklearn.manifold import TSNE

sns.set(style='whitegrid', context='notebook')
df = pd.read_csv('C:/Users/tekeste/Desktop/minmaxscale.csv')

RANDOM_SEED = 21
TRAINING_SAMPLE = 1800000
```

```
TRAINING_SAMPLE = 1800000
VALIDATE_SIZE = 0.20

np.random.seed(RANDOM_SEED)
tf.set_random_seed(RANDOM_SEED)
rn.seed(RANDOM_SEED)
df.columns = map(str.lower, df.columns)

df = df.drop(['unnamed: 0', 'unnamed: 0.1'], axis = 1)
df.head()

df.columns
# splitting by class
fraud = df[df.label == 1]
clean = df[df.label == 0]

fraud.shape
clean.shape

LABELS = ["Clean","Fraud"]
count_classes = pd.value_counts(df['label'], sort = True)
count_classes.plot(kind = 'bar', color = 'pink', rot=0, )
plt.xticks(range(2), LABELS)
plt.title("Frequency by observation number")
plt.xlabel("label")
plt.ylabel("Number of Observations");

RATIO_TO_FRAUD = 4
clean = clean.sample(frac=1).reset_index(drop=True)
clean.shape
# training set: exlusively non-fraud transactions
X_train = clean.iloc[:TRAINING_SAMPLE].drop('label', axis=1)
X_train.shape

# testing  set: the remaining non-fraud + all the fraud
X_test = clean.iloc[TRAINING_SAMPLE:].append(fraud).sample(frac=1)
X_test.shape
```

```
from sklearn.model_selection import train_test_split
# train // validate - no labels since they're all clean anyway
X_train, X_validate = train_test_split(X_train, test_size=VALIDATE_SIZE,
                                       random_state=RANDOM_SEED)
X_train.shape
X_validate.shape
# manually splitting the labels from the test df
X_test, y_test = X_test.drop('label', axis=1).values, X_test.label.values

X_test.shape
y_test.shape

from sklearn.preprocessing import Normalizer, MinMaxScaler
from sklearn.pipeline import Pipeline
# configure our pipeline
pipeline = Pipeline([('normalizer', Normalizer()),('scaler', MinMaxScaler())])
pipeline.fit(X_train);

X_train_transformed = pipeline.transform(X_train)
X_validate_transformed = pipeline.transform(X_validate)
# Load the extension and start TensorBoard
%load_ext tensorboard
%tensorboard --logdir logs
%tensorboard --logdir {logs_base_dir}

input_dim = X_train_transformed.shape[1]
input_dim1 = int(70)
input_dim2 = int(60)
input_dim3 = int(50)
input_dim4 = int(40)
input_dim5 = int(30)
input_dim6 = int(20)
input_dim7 = int(10)
```

```
latent_dim = 2
BATCH_SIZE = 256
EPOCHS = 100

autoencoder = tf.keras.models.Sequential([
    tf.keras.layers.Dense(input_dim, activation="tanh",input_shape=(input_dim, )),
    tf.keras.layers.Dense(input_dim1, activation="tanh",input_shape=(input_dim1, )),
    tf.keras.layers.Dense(input_dim2, activation="tanh",input_shape=(input_dim2, )),
    tf.keras.layers.Dense(input_dim3, activation="tanh",input_shape=(input_dim3, )),
    tf.keras.layers.Dense(input_dim4, activation="tanh",input_shape=(input_dim4, )),
    tf.keras.layers.Dense(input_dim5, activation="tanh",input_shape=(input_dim4, )),
    tf.keras.layers.Dense(input_dim6, activation="tanh",input_shape=(input_dim4, )),
    tf.keras.layers.Dense(input_dim7, activation="tanh",input_shape=(input_dim4, )),

    tf.keras.layers.GaussianNoise(0.01),
    tf.keras.layers.Dense(latent_dim, activation="tanh"),

       tf.keras.layers.Dense(input_dim7, activation="tanh"),
    tf.keras.layers.Dense(input_dim6, activation="tanh"),
    tf.keras.layers.Dense(input_dim5, activation="tanh"),
    tf.keras.layers.Dense(input_dim4, activation="tanh"),
    tf.keras.layers.Dense(input_dim3, activation="tanh"),
      tf.keras.layers.Dense(input_dim2, activation="tanh"),
       tf.keras.layers.Dense(input_dim1, activation="tanh"),
       tf.keras.layers.Dense(input_dim, activation="tanh"),

    ])

autoencoder.summary();
autoencoder.compile(optimizer='adam',
                    loss= 'mse',
                    metrics=['acc',])
```

```
autoencoder.summary();
autoencoder.compile(optimizer='adam',
                    loss= 'mse',
                    metrics=['acc',])
import os
from datetime import datetime

# current date and time
yyyymmddHHMM = datetime.now().strftime('%Y%m%d%H%M')

import datetime
log_dir="logs\\fit\\" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
# new folder for a new run
log_subdir = f'{yyyymmddHHMM}_batch{BATCH_SIZE}_layers{len(autoencoder.layers)}'

encoding_dim = 80
hidden_dim = int(encoding_dim / 2) #i.e. 7
learning_rate =10e-7
early_stop = tf.keras.callbacks.EarlyStopping(
    monitor='loss',
    min_delta=0.00005,
    patience=21,
    verbose=1,
    mode='max',
    restore_best_weights=True
)

save_model = tf.keras.callbacks.ModelCheckpoint(
    filepath='autoencoder_best_weights.hdf5',
    save_best_only=True,
    monitor='val_acc',
    verbose=0,
    mode='auto'
)
```

```
tensorboard = tf.keras.callbacks.TensorBoard(
   f'logs/{log_subdir}',
    batch_size=BATCH_SIZE,
    update_freq='batch'
)

# callbacks argument only takes a list
cb = [early_stop, save_model, tensorboard]

history = autoencoder.fit(
    X_train_transformed, X_train_transformed,
    epochs=EPOCHS,
    callbacks=cb,
    batch_size=BATCH_SIZE,
    shuffle=True,
    validation_data=(X_validate_transformed, X_validate_transformed)
);

##############################################################################
autoencoder = load_model('autoencoder_best_weights.hdf5')
plt.plot(history.history['loss'], linewidth=2, label='Train')
plt.plot(history.history['val_loss'], linewidth=2, label='Test')
plt.legend(loc='upper right')
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
#plt.ylim(ymin=0.70,ymax=1)
fig.savefig('accuracy.png')
plt.show()

plt.plot(history.history['acc'], linewidth=2, label='Train')
plt.plot(history.history['val_acc'], linewidth=2, label='Test')
plt.legend(loc='upper right')
plt.title('Model acc')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.show()
```

```
# transform the test set with the pipeline fitted to the training set
X_test_transformed = pipeline.transform(X_test)

# pass the transformed test set through the autoencoder to get the reconstructed result
reconstructions = autoencoder.predict(X_test_transformed)

X_test_predictions = autoencoder.predict(X_test_transformed)
mse= np.mean(np.power(X_test_transformed - reconstructions, 2), axis=1)
error_df = pd.DataFrame({'recostruction_error': mse, 'True_class':y_test})
error_df.describe()
sample_size = 100

# showing the reconstruction losses for a subsample of transactions
print(f'Mean Squared Error reconstruction losses for {sample_size} clean transactions:')
print(mse[np.where(y_test==0)][:sample_size])
print(f'\nMean Squared Error reconstruction losses for {sample_size} fraudulent transactions:')
print(mse[np.where(y_test==1)][:sample_size])
```