University of Nairobi

College of Physical and Biological sciences

Department of Computing and Informatics

MSc. DISTRIBUTED COMPUTING TECHNOLOGY

IMPROVEMENT OF EXISTING INTRUSION DETECTION SYSTEMS THROUGH NEURAL NETWORKS

WAWERU SALOME WANJIKU

P53/35492/2019

Supervisor: Dr. ELISHA.O. ABADE

Research Project submitted in partial fulfillment of the requirements for the award of Master of a Science in Distributed Computing Technology of University of Nairobi

DECLARATION

DECLARATION BY THE STUDENT

I declare that this research project is my original work and has never been submitted to or used as part of the award of a degree or any other Certificate in any institution of higher learning or any institution approved by the Ministry of Education, Science and Technology.

Sign: ....[illegible]........ Date: ....10/12/2021....

WAWERU SALOME WANJIKU

P53/35492/2019

DECLARATION BY THE SUPERVISOR

I declare that Waweru Salome Wanjiku was under my guidance and supervision all through this research. This research project has been submitted with my approval as the University Supervisor.

Sign: ...[illegible]........ Date: ...10/12/2021........

DR. ELISHA ABADE:

Senior Lecturer,

University of Nairobi

ACKNOWLEDGEMENT

I acknowledge and express my deep sense of gratitude to these for the assistance and guidance through this project.

My utmost gratitude to my supervisor Dr. Elisha Abade for the continuous constructive criticism which shaped this research and the guidance that led to completion of this project.

Sincerely grateful to my parents Mr. and Mrs. Jesse Waweru for providing the emotional support and always being there to encourage me to keep pursing my goals.

To my friends who kept pushing me and giving me hope in times of failure during research of this study.
To my mentors for the principles they taught me during research and their dedication to point me to the right direction of this study.

And above all, To God Almighty, the giver of wisdom and strength. Thankful for His graces for enabling me complete this milestone.

DEDICATION

This research is dedicated to Almighty God, the giver of life, source of this inspiration. Glory be to Him. I also dedicate this research to my parents and family for always pushing me to achieve more academically and get out of my comfort zone. Finally, I dedicate this research to Lucyann Wahito, a friend who always has my back in terms of need. Thank you.

ABSTRACT

Recently, there is an increased rate of improved and unknown intrusions in the Intrusion detection system (IDS) field. In this study, artificial neural networks (ANN) model is proposed and integrated to develop the IDS execution on classification. Continual semi-supervised learning is the primary method conducted on artificial neural networks to create the model. Neural networks undergo pre-training with extensive benchmark datasets used for intrusion detection depending on how the user profiles have been created that have exact simulation of events and behaviors that are checked on the network. The evaluation domain is summarized from a unique set of features from the data set. The dataset are different due to the different user profiles. The ANN are trained for diverse attacks like Network infiltration, DDOS, Web attacks and many other arising attacks.

The continual learning allows the model to learn latest advancements in deep learning and updates on recent anomalies detected. This improves the IDS to detect anomalies accurately. The main contribution of the proposed model to mitigate the gap identified is to assist in better classification of data, through the continual learning of neural networks, so that false positives that pass as normal data can be reduced in the system. It is demonstrated with experimental findings that the technique proposed can maintain a real time feedback to the anomaly.

Keywords- **Anomaly-based Intrusion, Intrusion detection system, Machine Learning, Neural networks, artificial neural networks**

Contents

Table of Figures

List of Tables

CHAPTER ONE: INTRODUCTION

1.1Background

A competent Intrusion detection system (IDS), should not only monitor a network for malicious activity but also keep auto-updating list of attacks so as to overcome new types of attacks (Alyousef & Abdelmajeed, 2019). Cyber-attacks have been on the rise, new threats and number of vulnerabilities keep increasing due to the spread of technologies. Most organizations are always trying to mitigate the risk of intrusion of their systems and networks. According to research done by Accenture, security breaches have inflated by 11% as from 2018 and 67% as from 2014(Annual et al., 2018).Hence both organizations and individuals are working towards gaining a secure network system that provides confidentiality, Integrity and availability.

The adoption of neural networks have been considered to improve Intrusion detection systems to be more efficient, due to their fast technological evolving nature. Neural networks act as an alternative result for tackling issues where the specific information required to create a proficient system is not accessible. Hackers whether from an outside intruder or an inside legitimate user mainly target the vulnerabilities of the system. The weaknesses of the system might be only specific to a certain version of the computer software or hardware. Therefore it is only mandatory to build a system that specifically monitors the actions of the users and not looking for any known weaknesses.(Shah & Issac, 2018)

This study proposes the need of a better anomaly intrusion detection system that in integrated with neural networks which improve its anomaly detection. The model involves training of neural networks methods to prevent attempt of intrusion through learning rules of intrusion characteristics (fadly. et al., 2009) . As mentioned, the model will use neural networks that will continuously be trained to detect anomalies, updates anomalies in an attack list in a database to assist in better diagnosis of intrusions. The neural network handles an investigation of the information then deliver a probability estimate to show that the data actually matches the characteristics trained to observe.

Intrusion Detection System

An efficient IDS continuously checks the network or a system to search for any doubtful behavior. Hence it is important that the IDS should recognize an attack and generate timely an alert in case of one and also provide a good remedy. (Sani et al., 2009)**.** The main purpose of the IDS is to detect and alert in case of any intrusion before the intruder does any major damage to the resources. An IDS should also protect the network from any compromise and misuse. The ultimate goal of any IDS is to catch perpetrators in the act before they do real damage to resources. An IDS should monitor network activity, check the data integrity, audit the active network and more. IDS, these days, have become vital component in the security toolbox. An IDS adds three major purposes functions: monitoring, detecting and generating an alert.

The intrusion detection systems (IDS) are commonly used as a way of resolving the anomalies issue. IDSs can either be:

- Host based IDSs: They monitor, check for suspicious information and issue an alert on an attack for an individual system

- Network Based IDSs: Through the use of sensors, they monitor the network infrastructure and generate alerts in case of an attack. These IDSs reviews the headers and contents of the packets in the network to detect anomalies that prove the network is a target of an attack.

The IDSs are mainly divided into two classes:

1. Anomaly-based Intrusion
2. Signature based Intrusion

An anomaly is a grouped set of data that does not adhere with the set standard behavior. Anomaly detection is therefore the problem of identifying the cluster that doesn't comply with the original behavior. In this study, the main anomaly detection will be focused on attacks like DDoS and web attacks. The anomalies found in data can consulate to useable high priority information hence anomaly detection shows its importance through detecting. Anomaly detection techniques have been developed and advanced to different varieties according to the need and usage of the IDS. (Chandola et al., 2009).

Signature (misuse) detection refers to the weak spots in the system or examples of well-known attacks to identify any intrusion. Signature-based Intrusion looks for specific patterns, like a particular sequence in the network traffic (Lateef et al., 2019). Zero day attacks cannot be identified by of signature-based intrusion which is a major disadvantage

Anomaly based Intrusion techniques

This technique is built to discover a certain trend of behavior that are not normal and flags anything that is not normal behavior. As illustrated on the figure below, there are different Anomaly intrusion techniques used to flag the abnormal behavior, namely: Statistical based technique, Cognitive based technique, Data mining based technique, Machine learning based techniques and computer immunology (Satam, 2017) .

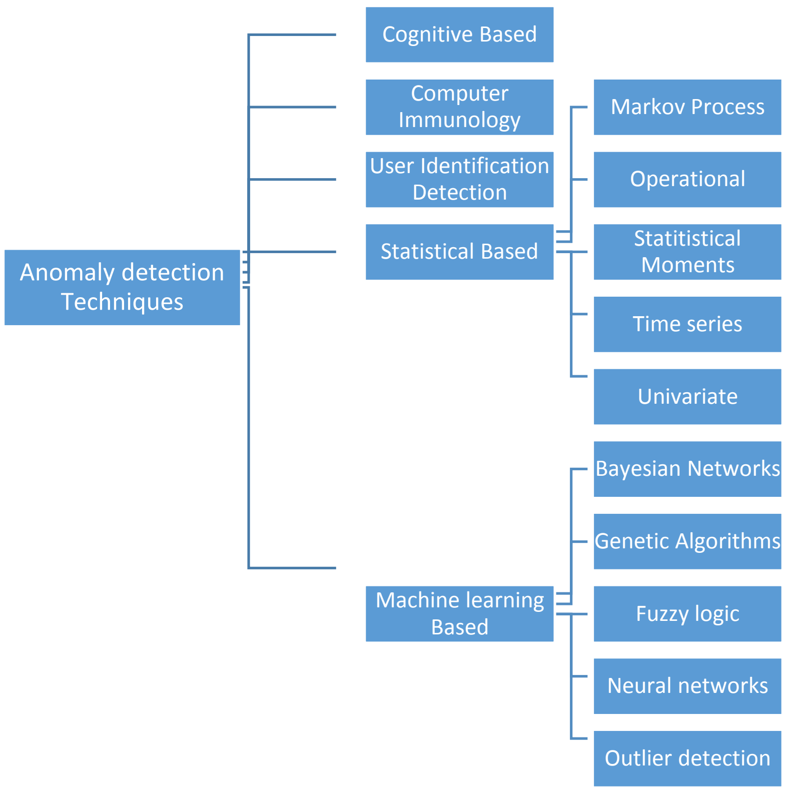


Figure 1: Anomaly detection techniques

Neural networks which are one of the various approaches in machine learning based techniques, will be used to build the model. The neural networks might have long training times and have high algorithmic complexity but they are flexible and adaptable (Satam, 2017).

Introduction to Neural Networks

A set of neurons make up a neural network. The neuron elements are connected and convert a set of inputs to a set of weighted outputs. The attributes of the elements and the weights that are associated in the interconnections among the neurons determine the outcome of the transformation.

Neural networks have a potential for learning which make them a promising component of an IDS handling anomaly detection. Neural networks are also used in anomaly detection to   assist in isolating new threats to adapt continuously to learn normal and abnormal for a system. (Sani et al., 2009)

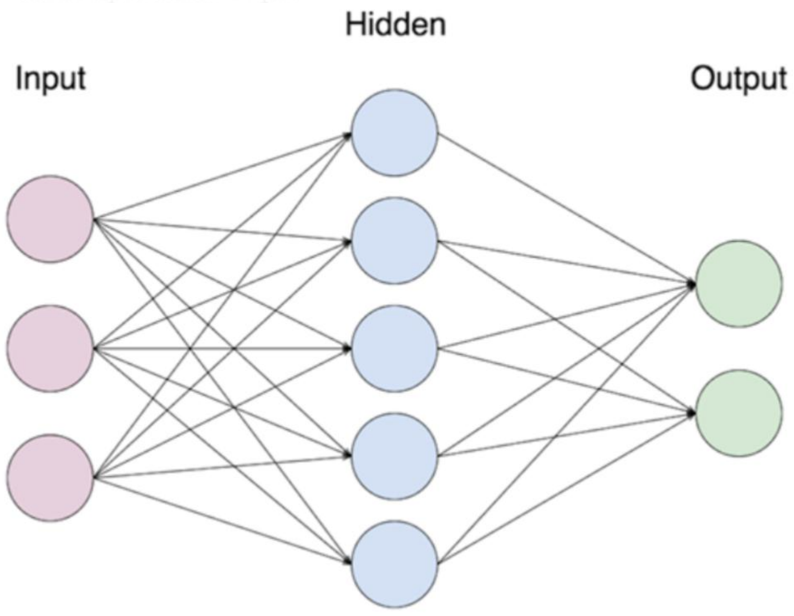


Figure 2: neural network model (chilakapati, 2019)

Deep neural networks are a series of algorithms used to identify the similarity of data by replicating the operation of a human brain. Intrusion detection systems is one of the many applications improved through the advancing neural networks. There are three different divisions that the neural networks can gain from which are:

- Supervised (discriminative): Training set has labeled examples and an observation is matched with one class only.

- Unsupervised manners (generative): Training set has no labeled examples and information do not belong to any class. During training, observations are grouped and calculated to its level of similarity.
- Hybrid deep architecture: and a combination of supervised and discriminative (Hybrid deep architecture) from data that is labelled and unlabeled (Lateef et al., 2019).

Snort IDS

Snort is a system that detects and also prevents network intrusions thorough packet sniffing and it is open source. Network traffic is analyzed real time by the snort system by scrutinizing the packet's payload or any skeptical anomalies. A set of rules are set and are continuously updated in snort. These rules are relied upon by snort to detect and give feedback on the malicious traffic.

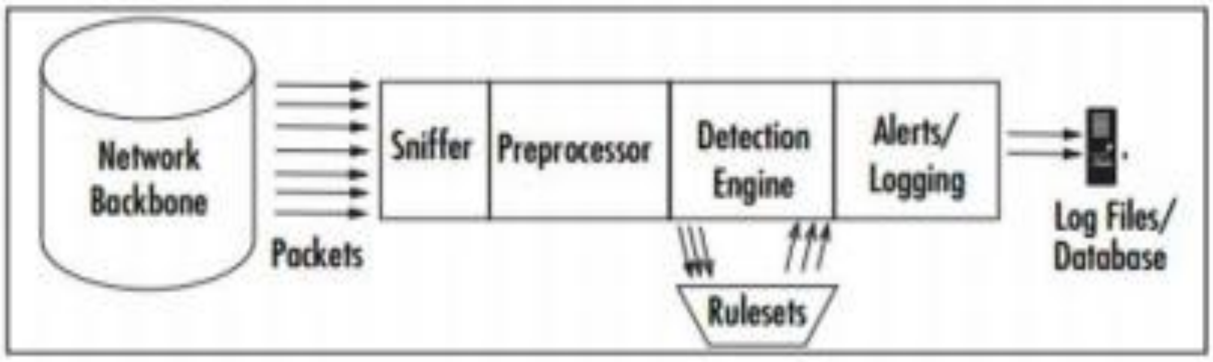


Figure 3: Snort architecture (Isaac, 2018)

Rules are set to enable network detection mode so that not every single packet sent to the traffic doesn't have to be recorded.

Snort has been optimized through other machine language algorithms with the aim of reducing false alarms (Raza Shah & Issac, 2017). Support vector machine, Decision tree, Fuzzy logic, Random Forest are some of the algorithms which have been implemented to improve the detection accuracy of the system.

It is therefore crucial to learn on machine learning algorithms that have high performance before using the techniques in snort(Shah et al., 2018).

In this study, Snort will be improved through deep learning neural networks and a performance comparison of snort results and model results to be conducted. Accuracy and precision are the metrics used to develop the ANN model and improve on the detection of the machine. True positives, false negatives and false positives are the metrics to be used to make a comparison on the detection accuracy of the chosen machine learning algorithm.

1.2 Problem Statement

There is a need to continuously improve the intrusion detection systems considering the high number of cyber-attacks recorded lately. Most data passes through the IDS as false positives due to poor detection analysis. Anomalies captured are not well diagnosed of the causing factors hence intrusions keep recurring.

1.3 General objective

The key goal of this study was to develop a machine learning model using neural networks that is continuously trained to better its clustering of data to reduce, update anomalies in an attack list in a database to curb future anomalies which in turn help in better diagnosis of intrusions. The model will be applied to improve the working mechanism of an existing IDS.

1.4 The Specific Objectives

The specific goals for this research are

1. To evaluate how the prototype model improves its anomaly detection mechanism by improving accuracy.
2. To evaluate the continual training of the prototype model and how it integrated to IDS to assist with better classification of data.
3. To compare the reduction of false positives using a normal existing IDS and that of the hardened IDS, that has improved through the neural network prototype model

1.5 Justification

The study will improve an existing IDS by improving the detection period to avoid further damage. The study will also improve documentation of the threats to an organization for better reporting. Better IDS will help in quality control in terms of designing the security and its administration. Finally, any intrusion information or the causing factors will be provided by the improved IDS which will assist in better diagnosis, and correction of the causing factors

Main contributions are continual learning for an updated IDS. An IDS should reduce the data to be inspected without decreasing its quality. Secondly, online detection of intrusions as early as possible to reduce any further damage with the intrusion. Thirdly, An intrusion detection system to classify data accurately to reduce the high false positives in anomaly-based IDS.

CHAPTER TWO: LITERATURE REVIEW

2.1 Past Works

Machine learning application on advancement of anomaly intrusion detection has been researched by several authors. The author (Jawhar, 2010) proposed that neural networks have the potential to address more issues. The author also suggested the basic characteristics of the users to be trained to the neural networks to allow identify major changes from the user's fixed behavior. During training phase new behaviors should be learnt automatically by the neural networks

This model also factors in the performance of an IDS and how to cope with a huge network traffic. Once an IDS face a large amount of network traffic, it might get overwhelmed and drop some packets hence missing some intrusions. (Alyousef & Abdelmajeed, 2019) highlight how the IDSs deal with a large number of anomalies, hence they propose the use of small databases to improve and enhance the IDS and how it works.

A denoising auto-encoder (DAE) model is recommended by (Shone et al., 2018) to discover important feature representations from the imbalanced training data and generate a model to detect normal and abnormal behaviors. This model is pre-trained using an unsupervised learning algorithm to avoid over fitting and local optima. A softmax classifier is supplemented above the model to show the deserved outputs. A total of 10% designated to be the test data from the KDDCUP99 data set was used and a detection accuracy of 94.71% was achieved by use of this method.

Further research studies propose the use of various data sets so as to expose the model to as much behavioral features as possible. (Vinayakumar et al., 2019) requests a model that is to classify and detect unforeseen attacks developed through deep neural network. Example of the datasets used are, Kyoto, UNSW-NB15, NSL-KDD, CICIDS 2017, WSNDS and KDDCup 99. SHIA which is an intrusion detection framework which is scalable and hybrid is used to detect malicious characteristics by processing large amount of host level and network level events in order to provide correct alerts to the admin.

(Yin et al., 2017) recommends a deep learning model that uses recurrent neural networks (RNN) to accelerate IDS performance. Aim of the model is to learn, in a multiclass classification environment, the achievements of the RNN-IDS and the distinctive learning rate on the number of neurons and how accuracy of the model is affected.

(Shah & Issac, 2018) proposes to advance an IDS, specifically snort through machine learning by comparing the accuracy of snort IDS plugged in with five different algorithms and performance of the algorithms in three datasets (DARPA, NSA, NSL-KDD). Compared to the other datasets, NSA dataset assisted to improve snort’s detection accuracy and reduced the false positives.

Author (Anas I. Al Suwailem, 2019) attempts to prove a machine learning accuracy concept of detecting anomalies which have already been recognized as classes. Unknown and known classes that are true positives and false negatives are classified by the decision tree algorithm. KDD’99 dataset is used. The dataset has 1 normal class and 22 attack classes where one class has been set an instance in the dataset. The aim of the research is finding another way of snort to detect new anomalies that assists with snort to detected false alarms generated

2.2 Gap

Most algorithms do training separately and testing separately which takes too much time in training. Performance of the neural networks also vary and take time to fully inspect packets for intrusion detection. Intrusion detection system are not able to work in an efficient manner due to poor classification of data.

2.3 Linking related work to this study

Various research studies bring forth the urgency to promote the accuracy of neural networks and advance execution of the neural networks. Most algorithms are built with old datasets, building the algorithms with recent algorithms will improve anomaly detection due to observance of new and recent behavioral features.

Previous work propose to train models with only the normal behavioral features is not enough and the framework should be changed in order to leave out the anomalous instances. Neural networks should be made known to the metrics of heretical structured data.

2.4 Conceptual Architecture

Below is the conceptual architecture for this study

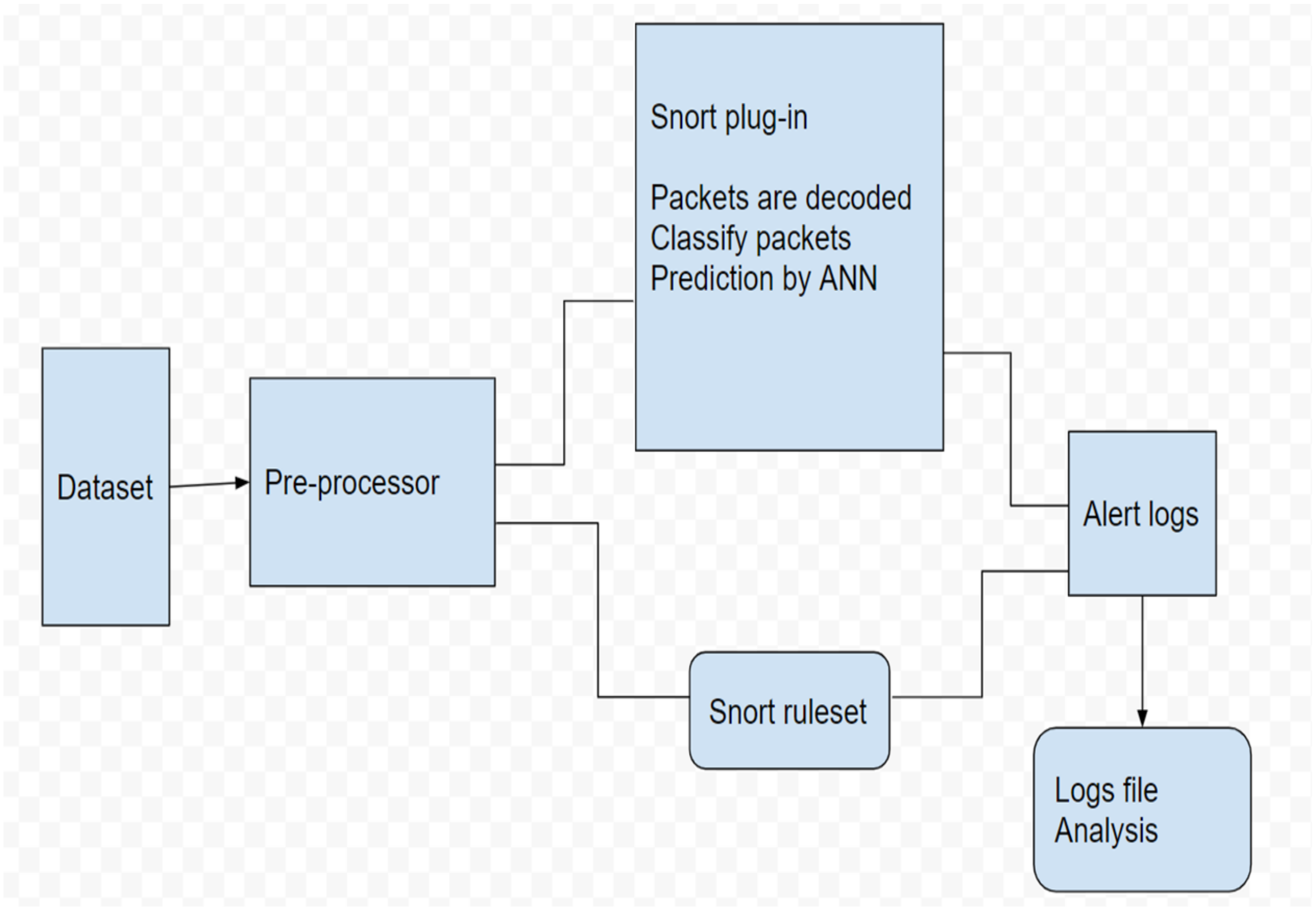


Figure 4: Conceptual architecture

CHAPTER THREE: METHODOLOGY

3.1 Introduction

This chapter elaborates on the procedure involved in the development of the prototype as well as the research methods used for data collection. A quantitative methodology will be adopted.

3.2 Research Design

This study will adopt an experimental research design. A set of procedures will be used to collect, analyze and process the data for this study. The procedures assist with possible approaches that can be used to carry out research. Therefore, the research result is to gain knowledge and advance science.

To enhance the experimental design, the variables are considered and their interrelation is well indicated to assist in making predictions which will be testable and specific

3.3 Data Collection

Data for this study is captured from an IDS, snort, and used to train a machine learning model. The dataset act as a simulation and provides learning data. The logs captured from the system are preprocessed and rationalized to extract the useful features.

The dataset captured has both normal and attack data simulated from a real network to ensure the training environment is accurate.

3.3.1 Quantitative Methods

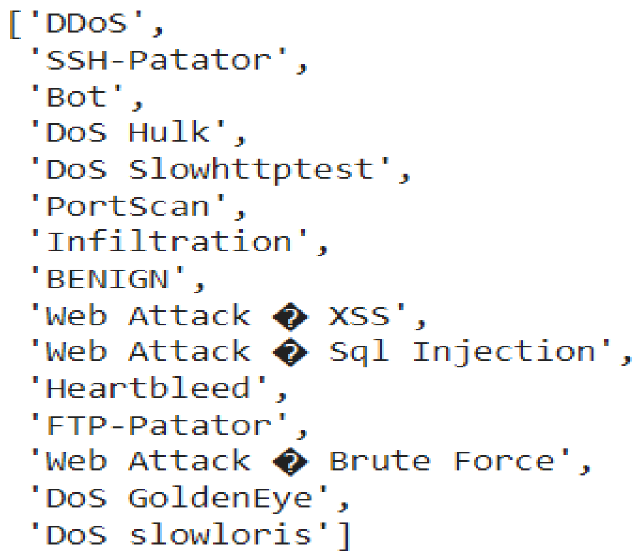
Quantitative method of research assists in quantifying the issue by producing data which is numerical that can be converted into useful statistics. This method of research uses computable data to develop facts and formulate patterns in research.

3.3.1.1 Datasets

A dataset is required for training and testing a machine learning model. A data set is a database table or a statistical matrix. A dataset is also pieces of data collected and regarded to the computer as one entity for purposes of prediction and analytics. The database table represents a particular variable in every column and rows corresponds to specific members in the data set. Data sets are used to interpret, test and assess intrusions. A modern data set will have both attack and non-anomalous data that are helpful to train the neural network model. The non-anomalous data will cluster into a certain pattern and then will be compared with data which is anomalous.

In this study CICIDS-2017 dataset that was accessed from the consortium of ISCX. The dataset consists of a Machine Learning CSV folder that consists of eight CSV files which are sessions captured during traffic audit. These files contain both normal and anomaly traffic. Normal traffic is marked as “Benign” whereas anomaly traffic is referred to as “Attack”. There are 14 classes of attacks in total. The data was collected in a period of 5 days.

Below are attack classes in the data:



```
['DDoS',
 'SSH-Patator',
 'Bot',
 'DoS Hulk',
 'DoS Slowhttptest',
 'PortScan',
 'Infiltration',
 'BENIGN',
 'Web Attack � XSS',
 'Web Attack � Sql Injection',
 'Heartbleed',
 'FTP-Patator',
 'Web Attack � Brute Force',
 'DoS GoldenEye',
 'DoS slowloris']
```

Figure 5: Dataset classes

3.4 Data Analysis and Interpretation

Various datasets are analyzed to evaluate the various features in the data. The data are to be harvested, compared and summarized under one criteria to evaluate consistency. Data that will not meet the set criteria will be deleted and more datasets to be used. A total of 8 datasets from one source are used for training and testing. The features are checked and analyzed to check similarity of the data to ensure the model learns as much as possible from a uniformed set of data.

3.5 System Development Methodology

3.5.1 Snort configuration

Snort is configured with the correct libraries to enable the system read the pcap files.

The datasets are sourced as pcap files and are run through snort IDS to check the original findings that is later be compared to the new findings of snort after the ANN is integrated.

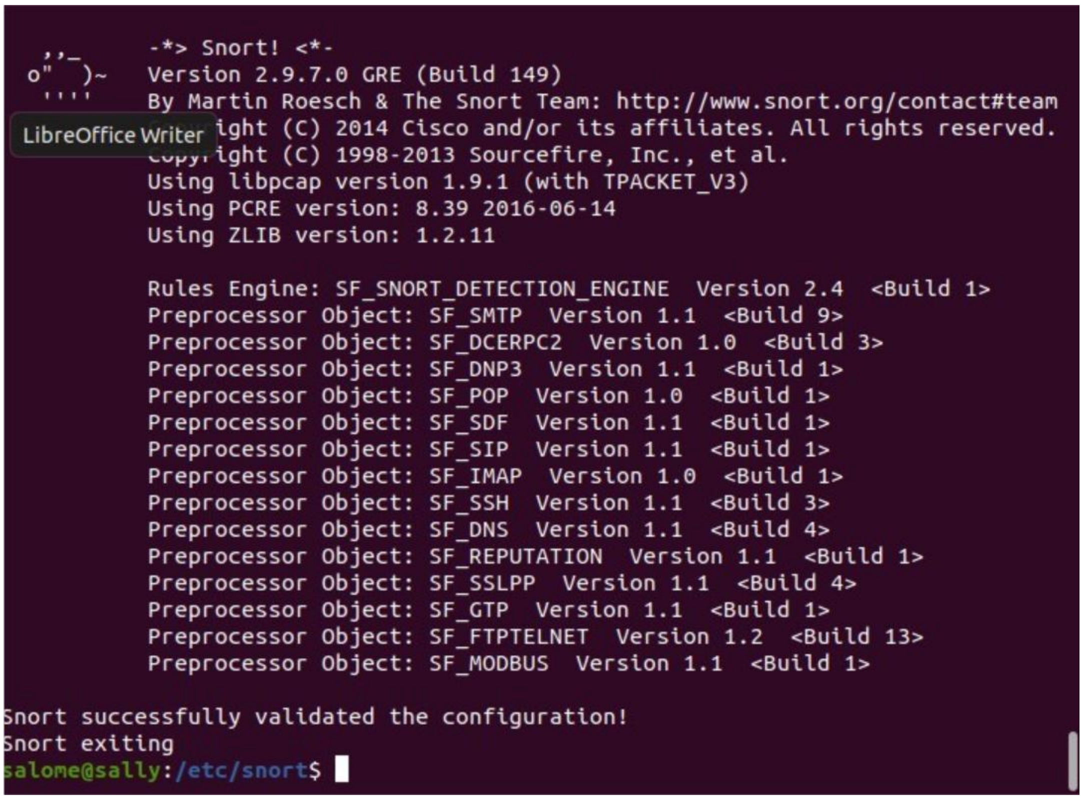
Snort intrusion detection system is appended some rules that could best capture the data. Below is the format of rules configured with the main protocols that are to be monitored

Table 1: Snort Rule Format

Snort Rule format	
Alert Example	alert icmp any any -> 192.168.56.101/24 any
Actions	Alert ,drop, log, pass, drop, reject,
Protocols	UDP, ICMP, IP,TCP,

Once the rules and libraries are configured, snort is validated to check if they are functional in detection of system.

A total of 3467 rules were configured in the ruleset under snort.conf configuration file.



```
   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.7.0 GRE (Build 149)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
              ight (C) 2014 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.9.1 (with TPACKET_V3)
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.11

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 2.4  <Build 1>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>

Snort successfully validated the configuration!
Snort exiting
salome@sally:/etc/snort$
```

Figure 6: Snort configuration

3.5.1.1 Snort findings

Below are the results that were recorded once the dataset are loaded into snort

Below is the summary of the findings.

Table 2: snort findings

Dataset	Actual bytes	Rate (packet per second)	Unrecognized data	ALERTS
(CICIDS2017)	116888	35456.875	216,269	31385

3.5.2 Prototype development

Prototype model development is illustrated on the figure below to show the process of how the prototype model will be built.

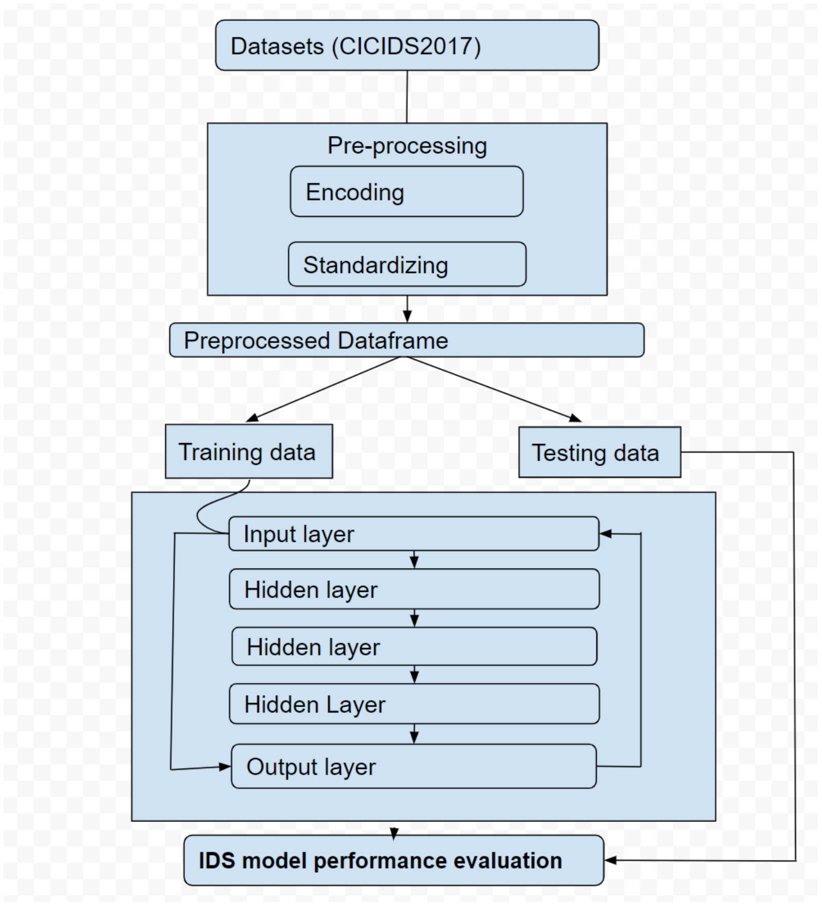


Figure 7: Proposed ANN Model

Below is the step by step method of training of the model

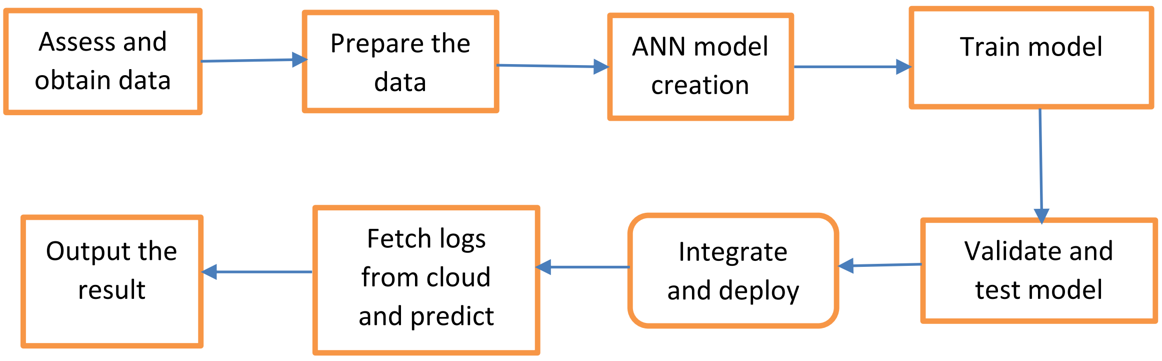


Figure 8: ANN model development

3.5.2.1 Problem assessment

Network-connected devices in the network and their respective functions are accounted for. Identification of high priority systems that store sensitive information is also done. The assessment is an analysis of the most familiar discovered attacks. The review assesses any vulnerabilities that the system is exposed to, assign the levels of severity to the vulnerabilities and finally recommends a mitigation for the vulnerabilities.

3.5.2.2 Acquisition of data

Data can be acquired from a network with detection system as the security system. The logs can be captured and extracted. Data can also be acquired from existing datasets.

In this study CICIDS2017 is a publicly available dataset that has benign data which is considered normal and common attacks which are most recent. The CSV file has data flows dependent on timestamps, destination and source IP addresses and ports. The features are complex and large in number to be observed for detection of anomalies.

3.5.2.3 Data preparation

CIC-IDS2017 data set has redundant features hence the redundant features are removed and only 20% will be used for training the model. The 20%remaining data is divided into training and testing data. 70% for training and 30 % for testing.

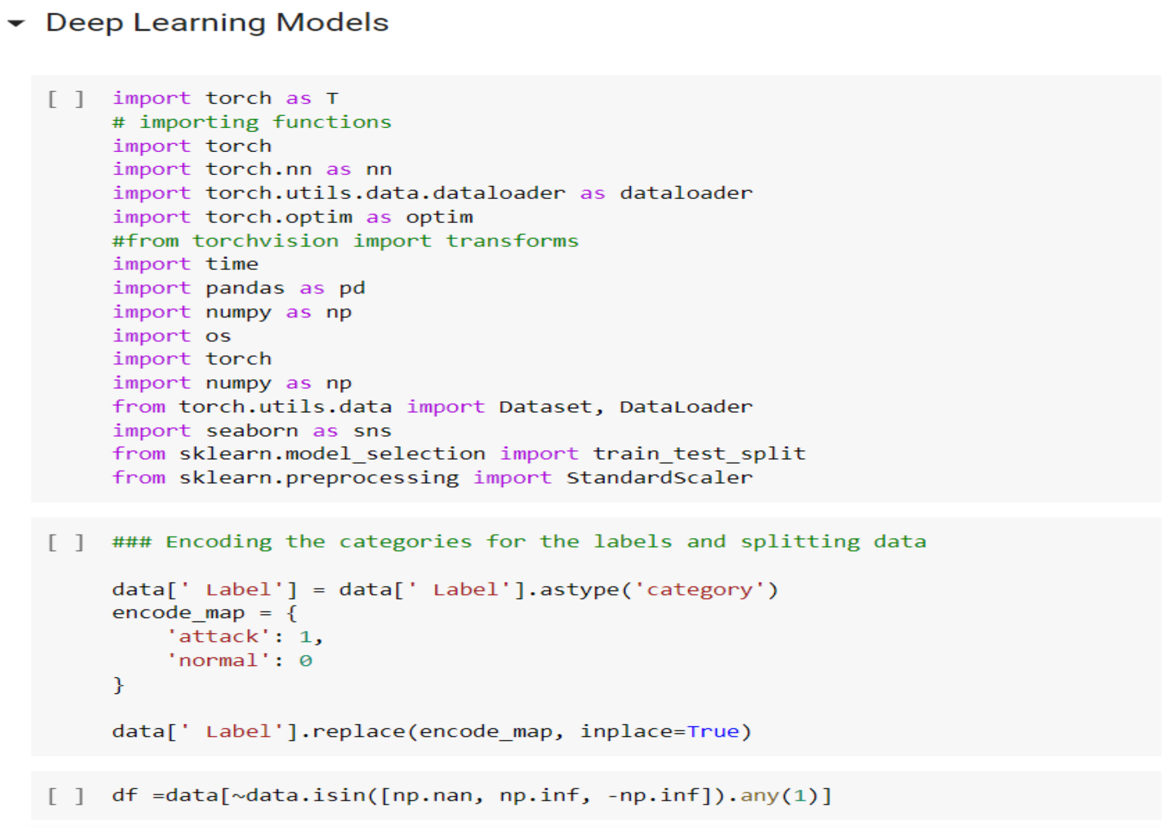
Selection of features is done on 70% of the data used for training. The features selected are grouped to specific weights. The model is therefore trained through unsupervised learning.

The training data has to be numerical within datatypes of integer, float64, and float32. The eight datasets are combined into one data frame as one large set of data

3.5.2.4 ANN model creation

Neural networks are trained using semi-supervised learning for the proposed model. The neural networks are to be built into a simple baseline model then more layers to be built on the baseline as the continuous training of the neural networks. The necessary libraries are imported.

Encoding the categories for the labels is done for training purposes for the model to learn the normal and attack data. The data is cleaned to remove the missing data, features that are not numbers and any infinity numbers.



▾ Deep Learning Models

```
[ ]  import torch as T
     # importing functions
     import torch
     import torch.nn as nn
     import torch.utils.data.dataloader as dataloader
     import torch.optim as optim
     #from torchvision import transforms
     import time
     import pandas as pd
     import numpy as np
     import os
     import torch
     import numpy as np
     from torch.utils.data import Dataset, DataLoader
     import seaborn as sns
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import StandardScaler
```

```
[ ]  ### Encoding the categories for the labels and splitting data

     data[' Label'] = data[' Label'].astype('category')
     encode_map = {
         'attack': 1,
         'normal': 0
     }

     data[' Label'].replace(encode_map, inplace=True)
```

```
[ ]  df =data[~data.isin([np.nan, np.inf, -np.inf]).any(1)]
```

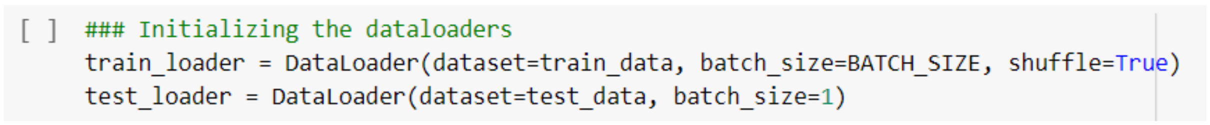
Figure 9: ANN data encoding

3.5.2.5 Training the model

Once the data is ready it will be scaled due to the huge amount of data to avoid bias during training or skewness.

Before training commences, the shape of the training and testing data is defined then converted to numpy arrays. The arrays are then stacked to allow it to be in the correct format for analysis.

Data loaders are used to load the data in chunks and loops to avoid too much usage of RAM all at once.



```
[ ]  ### Initializing the dataloaders
     train_loader = DataLoader(dataset=train_data, batch_size=BATCH_SIZE, shuffle=True)
     test_loader = DataLoader(dataset=test_data, batch_size=1)
```

Figure 10: ANN model data loaders

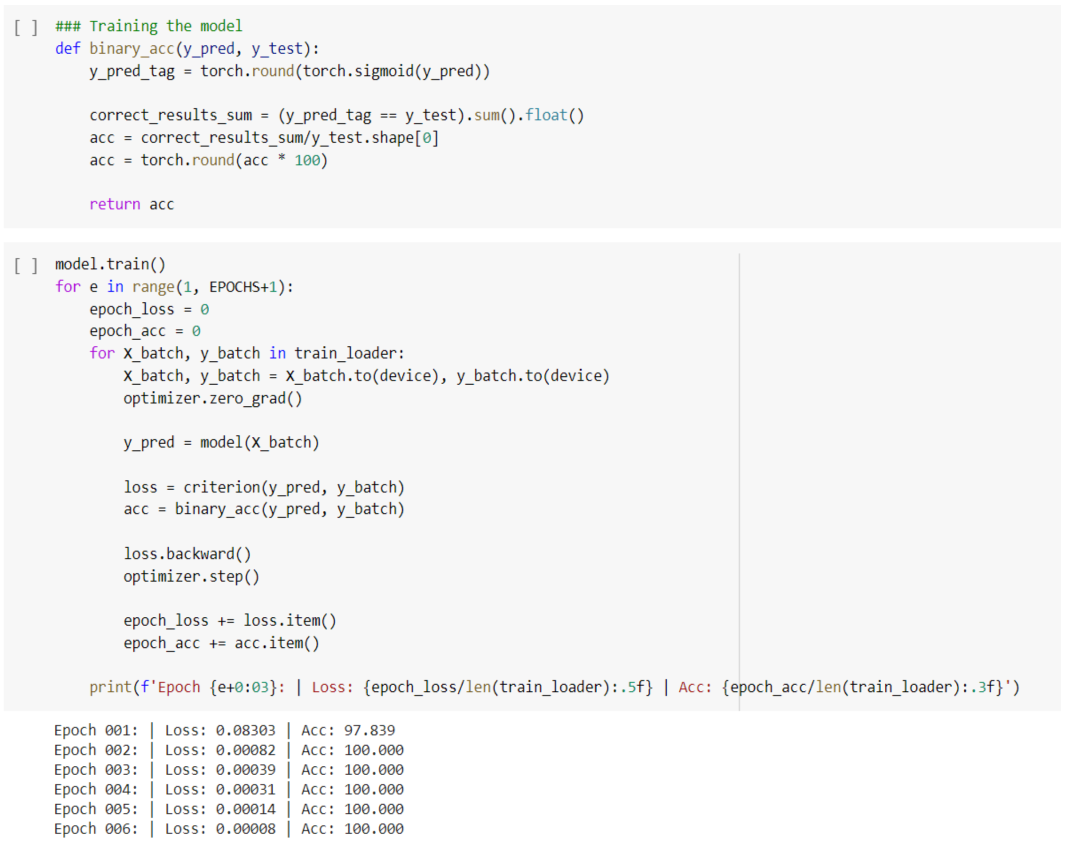
The model has 5 layers and the inputs are initialized before training. The inputs and outputs are initialized from the first to the fifth layer. Output of the first layer is initialized as input of the adjacent layer below.

Below are the parameters used to train the ANN model.

Table 2: Model Parameters

Model Parameters	
Epoch	2048
Learning rate	0.01
Input size	size of the training data
Drop out	0.1
Optimizer	Adam

Epoch is the number of complete loops through the five layers during training. The drop out shows how the data is distributed through the hidden layers during the learning and train stage



```
[ ]  ### Training the model
     def binary_acc(y_pred, y_test):
         y_pred_tag = torch.round(torch.sigmoid(y_pred))

         correct_results_sum = (y_pred_tag == y_test).sum().float()
         acc = correct_results_sum/y_test.shape[0]
         acc = torch.round(acc * 100)

         return acc
```

```
[ ]  model.train()
     for e in range(1, EPOCHS+1):
         epoch_loss = 0
         epoch_acc = 0
         for X_batch, y_batch in train_loader:
             X_batch, y_batch = X_batch.to(device), y_batch.to(device)
             optimizer.zero_grad()

             y_pred = model(X_batch)

             loss = criterion(y_pred, y_batch)
             acc = binary_acc(y_pred, y_batch)

             loss.backward()
             optimizer.step()

             epoch_loss += loss.item()
             epoch_acc += acc.item()

         print(f'Epoch {e+0:03}: | Loss: {epoch_loss/len(train_loader):.5f} | Acc: {epoch_acc/len(train_loader):.3f}')

     Epoch 001: | Loss: 0.08303 | Acc: 97.839
     Epoch 002: | Loss: 0.00082 | Acc: 100.000
     Epoch 003: | Loss: 0.00039 | Acc: 100.000
     Epoch 004: | Loss: 0.00031 | Acc: 100.000
     Epoch 005: | Loss: 0.00014 | Acc: 100.000
     Epoch 006: | Loss: 0.00008 | Acc: 100.000
```

Figure 11: ANN model training

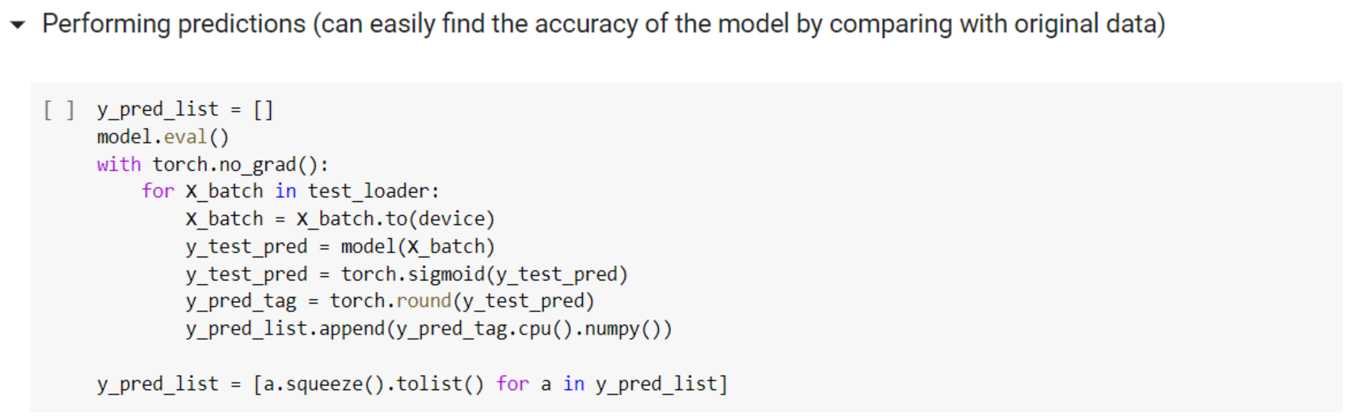
The neural network parameters are improved to merge outputs from the previous layer with reciprocal input patterns. The output for each round of training will be compared to the expected output during validation then the correct cost is assigned to the complete operation. The cost will determine the extent of modification to both the threshold and the weights. To ensure the IDS improves its performance, the training exercise is done continuously.(Sani et al., 2009).

During training, the model is built depending on the behavioral features of the user profiles in the specified system. Observations of intrusions are done manually or automatically. After observations, the model shows its availability by detecting the intrusions by matching the observed traffic.

Once the training is done according to the parameters set, the model is saved. The model can act as a baseline of the next model to transfer the learning to more layers of the neural networks that can be added.

3.5.2.6 Testing the Algorithm

Prediction is done to confirm Penetration testing is also conducted to assess the model. Status reports from the logs collected are done for further diagnosis by the network administrators.



Performing predictions (can easily find the accuracy of the model by comparing with original data)

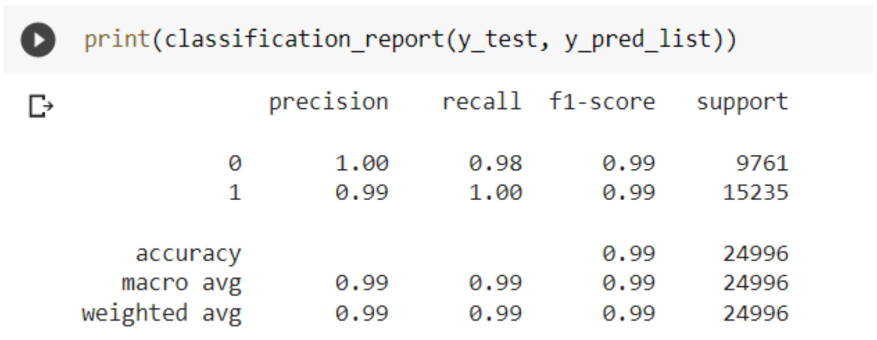
```
[ ]  y_pred_list = []
     model.eval()
     with torch.no_grad():
         for X_batch in test_loader:
             X_batch = X_batch.to(device)
             y_test_pred = model(X_batch)
             y_test_pred = torch.sigmoid(y_test_pred)
             y_pred_tag = torch.round(y_test_pred)
             y_pred_list.append(y_pred_tag.cpu().numpy())

     y_pred_list = [a.squeeze().tolist() for a in y_pred_list]
```

Figure 12: Prediction of model

3.5.2.7 Prediction of logs to test the accuracy of the ANN model

Logs captured from another network are retrieved from the cloud are stored in the cloud. The folder is copied from the cloud to google colab to test the neural network model prediction capability.



```
print(classification_report(y_test, y_pred_list))
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99      9761
           1       0.99      1.00      0.99     15235

    accuracy                           0.99     24996
   macro avg       0.99      0.99      0.99     24996
weighted avg       0.99      0.99      0.99     24996
```

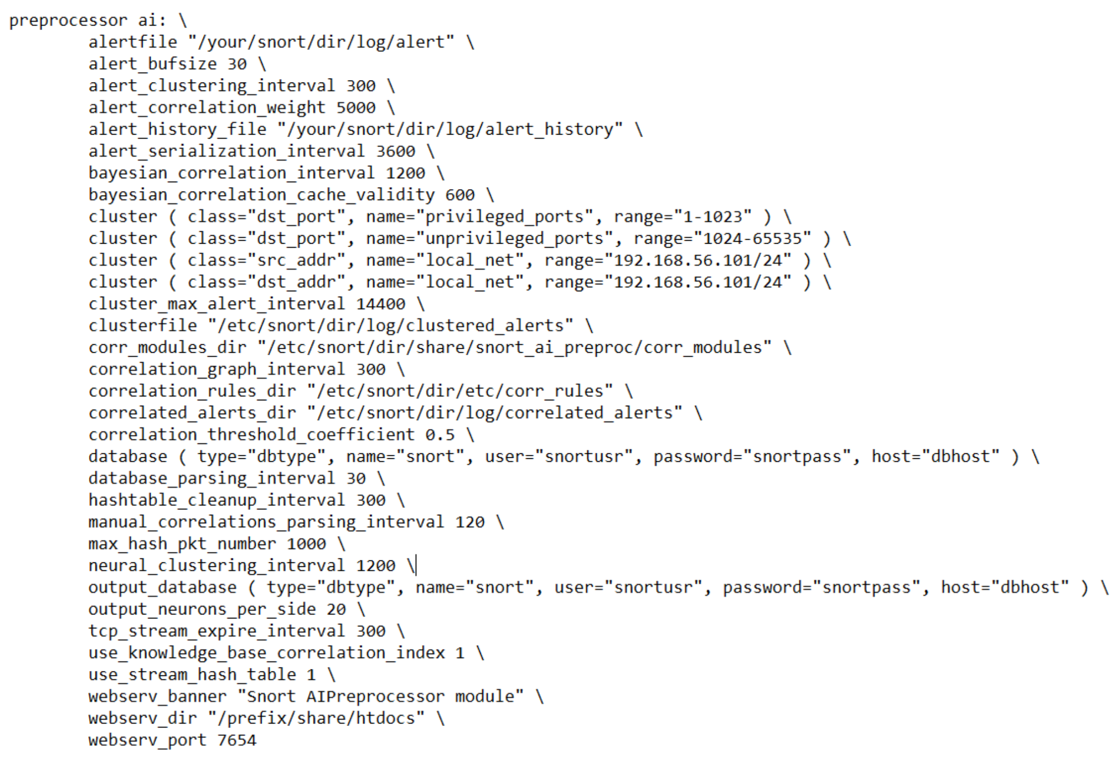
Figure 13: Classification report

3.5.3 Integration of proposed model to snort

3.5.3.1 AI preprocessor

Snort is integrated with the ANN proposed model. The integration is done through a snort plug-in which is to work side by side with the rules set in snort. The plugin through c language is imported to the Linux operating system. The required libraries are added to allow the processor to read pcap files.

The figure below shows how the preprocessor was configured in snort.conf configuration file. The preprocessor is helpful in clustering both true and false alarms from traffic over ports and IP addresses. The preprocessor is also configured to output alerts.



```
preprocessor ai: \
        alertfile "/your/snort/dir/log/alert" \
        alert_bufsize 30 \
        alert_clustering_interval 300 \
        alert_correlation_weight 5000 \
        alert_history_file "/your/snort/dir/log/alert_history" \
        alert_serialization_interval 3600 \
        bayesian_correlation_interval 1200 \
        bayesian_correlation_cache_validity 600 \
        cluster ( class="dst_port", name="privileged_ports", range="1-1023" ) \
        cluster ( class="dst_port", name="unprivileged_ports", range="1024-65535" ) \
        cluster ( class="src_addr", name="local_net", range="192.168.56.101/24" ) \
        cluster ( class="dst_addr", name="local_net", range="192.168.56.101/24" ) \
        cluster_max_alert_interval 14400 \
        clusterfile "/etc/snort/dir/log/clustered_alerts" \
        corr_modules_dir "/etc/snort/dir/share/snort_ai_preproc/corr_modules" \
        correlation_graph_interval 300 \
        correlation_rules_dir "/etc/snort/dir/etc/corr_rules" \
        correlated_alerts_dir "/etc/snort/dir/log/correlated_alerts" \
        correlation_threshold_coefficient 0.5 \
        database ( type="dbtype", name="snort", user="snortusr", password="snortpass", host="dbhost" ) \
        database_parsing_interval 30 \
        hashtable_cleanup_interval 300 \
        manual_correlations_parsing_interval 120 \
        max_hash_pkt_number 1000 \
        neural_clustering_interval 1200 \
        output_database ( type="dbtype", name="snort", user="snortusr", password="snortpass", host="dbhost" ) \
        output_neurons_per_side 20 \
        tcp_stream_expire_interval 300 \
        use_knowledge_base_correlation_index 1 \
        use_stream_hash_table 1 \
        webserv_banner "Snort AIPreprocessor module" \
        webserv_dir "/prefix/share/htdocs" \
        webserv_port 7654
```

Figure 14: AI preprocessor configuration

The datasets are run in snort, the pre-processor receives and feeds the data for detection. Packet decoding is the step done. The decoding of data entails analyzing the packet flow parameters.

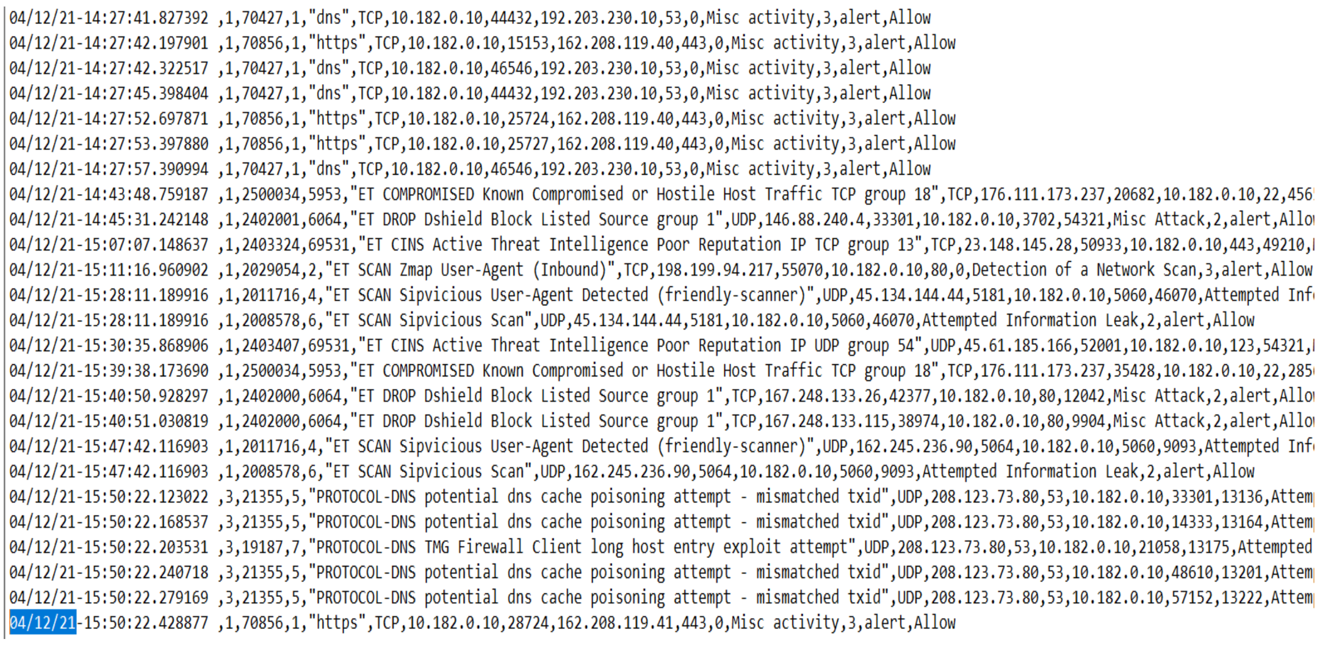
The next step is both normal and attack data is classified. Snort processes the data and logs the alarms. The model also properly classifies the data which reduces true positive and false positive alarms and logs the alarms. Alerts that are sent by the IDS are logged and sent for analysis in the log file.

The summary of the log file analysis assists the snort to have a better ruleset and also assist to detect packets that might have passed as normal data.

Python library is added to snort. The library allows the addition of the neural code to snort.

3.5.3.2 Alert file

An alert log file from the IDS is output and saved for analysis. The logs are analyzed for clarification on the anomalies detected



```
04/12/21-14:27:41.827392 ,1,70427,1,"dns",TCP,10.182.0.10,44432,192.203.230.10,53,0,Misc activity,3,alert,Allow
04/12/21-14:27:42.197901 ,1,70856,1,"https",TCP,10.182.0.10,15153,162.208.119.40,443,0,Misc activity,3,alert,Allow
04/12/21-14:27:42.322517 ,1,70427,1,"dns",TCP,10.182.0.10,46546,192.203.230.10,53,0,Misc activity,3,alert,Allow
04/12/21-14:27:45.398404 ,1,70427,1,"dns",TCP,10.182.0.10,44432,192.203.230.10,53,0,Misc activity,3,alert,Allow
04/12/21-14:27:52.697871 ,1,70856,1,"https",TCP,10.182.0.10,25724,162.208.119.40,443,0,Misc activity,3,alert,Allow
04/12/21-14:27:53.397880 ,1,70856,1,"https",TCP,10.182.0.10,25727,162.208.119.40,443,0,Misc activity,3,alert,Allow
04/12/21-14:27:57.390994 ,1,70427,1,"dns",TCP,10.182.0.10,46546,192.203.230.10,53,0,Misc activity,3,alert,Allow
04/12/21-14:43:48.759187 ,1,2500034,5953,"ET COMPROMISED Known Compromised or Hostile Host Traffic TCP group 18",TCP,176.111.173.237,20682,10.182.0.10,22,456
04/12/21-14:45:31.242148 ,1,2402001,6064,"ET DROP Dshield Block Listed Source group 1",UDP,146.88.240.4,33301,10.182.0.10,3702,54321,Misc Attack,2,alert,Allo
04/12/21-15:07:07.148637 ,1,2403324,69531,"ET CINS Active Threat Intelligence Poor Reputation IP TCP group 13",TCP,23.148.145.28,50933,10.182.0.10,443,49210,
04/12/21-15:11:16.960902 ,1,2029054,2,"ET SCAN Zmap User-Agent (Inbound)",TCP,198.199.94.217,55070,10.182.0.10,80,0,Detection of a Network Scan,3,alert,Allow
04/12/21-15:28:11.189916 ,1,2011716,4,"ET SCAN Sipvicious User-Agent Detected (friendly-scanner)",UDP,45.134.144.44,5181,10.182.0.10,5060,46070,Attempted Inf
04/12/21-15:28:11.189916 ,1,2008578,6,"ET SCAN Sipvicious Scan",UDP,45.134.144.44,5181,10.182.0.10,5060,46070,Attempted Information Leak,2,alert,Allow
04/12/21-15:30:35.868906 ,1,2403407,69531,"ET CINS Active Threat Intelligence Poor Reputation IP UDP group 54",UDP,45.61.185.166,52001,10.182.0.10,123,54321,
04/12/21-15:39:38.173690 ,1,2500034,5953,"ET COMPROMISED Known Compromised or Hostile Host Traffic TCP group 18",TCP,176.111.173.237,35428,10.182.0.10,22,285
04/12/21-15:40:50.928297 ,1,2402000,6064,"ET DROP Dshield Block Listed Source group 1",TCP,167.248.133.26,42377,10.182.0.10,80,12042,Misc Attack,2,alert,Allo
04/12/21-15:40:51.030819 ,1,2402000,6064,"ET DROP Dshield Block Listed Source group 1",TCP,167.248.133.115,38974,10.182.0.10,80,9904,Misc Attack,2,alert,Allo
04/12/21-15:47:42.116903 ,1,2011716,4,"ET SCAN Sipvicious User-Agent Detected (friendly-scanner)",UDP,162.245.236.90,5064,10.182.0.10,5060,9093,Attempted Inf
04/12/21-15:47:42.116903 ,1,2008578,6,"ET SCAN Sipvicious Scan",UDP,162.245.236.90,5064,10.182.0.10,5060,9093,Attempted Information Leak,2,alert,Allow
04/12/21-15:50:22.123022 ,3,21355,5,"PROTOCOL-DNS potential dns cache poisoning attempt - mismatched txid",UDP,208.123.73.80,53,10.182.0.10,33301,13136,Attem
04/12/21-15:50:22.168537 ,3,21355,5,"PROTOCOL-DNS potential dns cache poisoning attempt - mismatched txid",UDP,208.123.73.80,53,10.182.0.10,14333,13164,Attem
04/12/21-15:50:22.203531 ,3,19187,7,"PROTOCOL-DNS TMG Firewall Client long host entry exploit attempt",UDP,208.123.73.80,53,10.182.0.10,21058,13175,Attempted
04/12/21-15:50:22.240718 ,3,21355,5,"PROTOCOL-DNS potential dns cache poisoning attempt - mismatched txid",UDP,208.123.73.80,53,10.182.0.10,48610,13201,Attem
04/12/21-15:50:22.279169 ,3,21355,5,"PROTOCOL-DNS potential dns cache poisoning attempt - mismatched txid",UDP,208.123.73.80,53,10.182.0.10,57152,13222,Attem
04/12/21-15:50:22.428877 ,1,70856,1,"https",TCP,10.182.0.10,28724,162.208.119.41,443,0,Misc activity,3,alert,Allow
```

3.6 Tools Analysis

The proposed model is developed on google Colab . Python language version 3.9 is used. Pytorch version 1.9 is the main library to be use in the environment.

Snort, the IDS tool is installed on ubuntu, which is a linux distribution. The Ubuntu is hosted on a virtual box

3.7 Methodology Summary

Figure 14 shows the summary of the chapter to show a step by step improvement of snort.

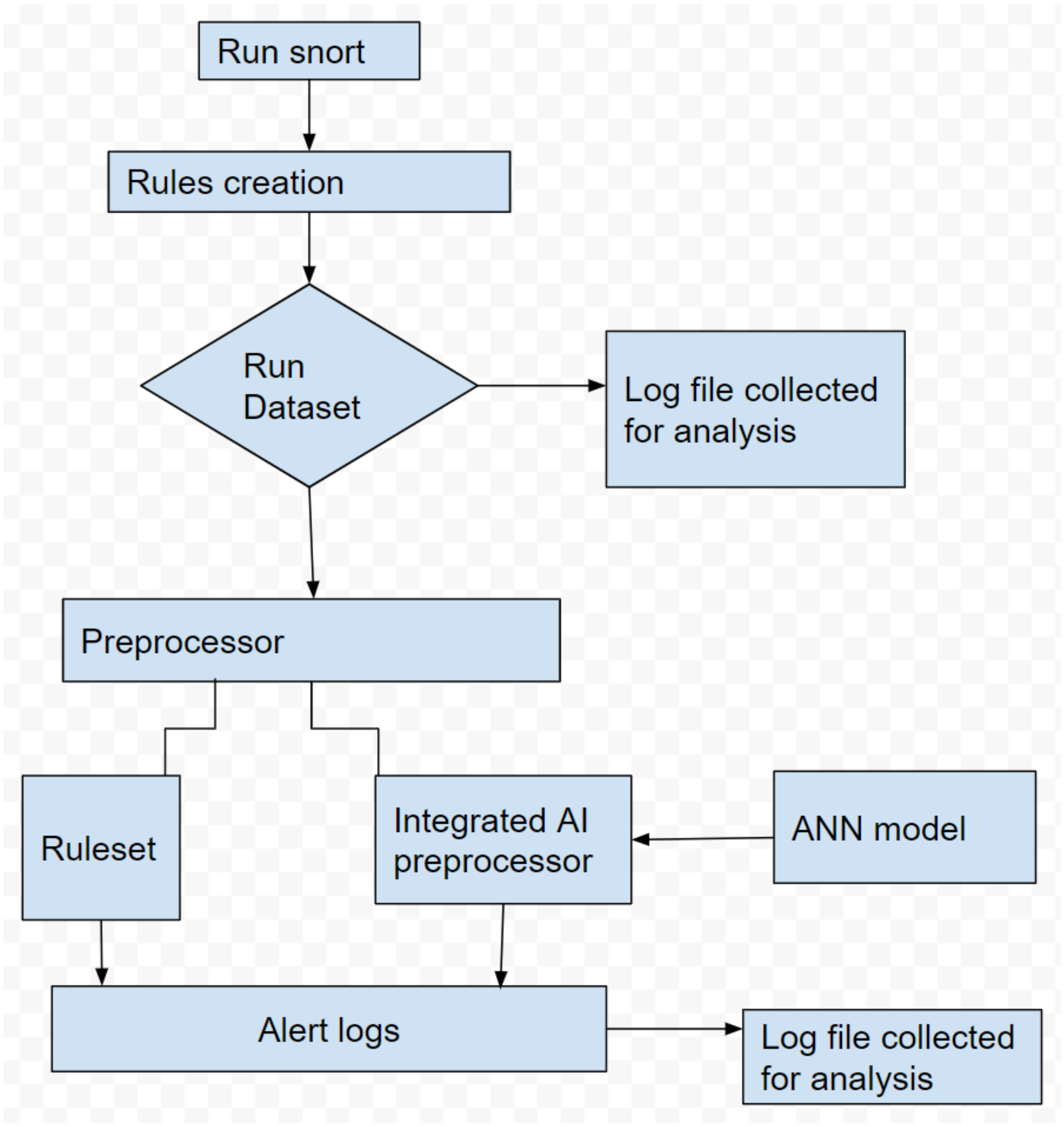

Run snort
Rules creation
Run Dataset
Log file collected for analysis
Preprocessor
Ruleset
Integrated AI preprocessor
ANN model
Alert logs
Log file collected for analysis

Figure 15 : methodology summary

CHAPTER FOUR: EXPECTED RESULTS

4.1 Introduction

The proposed model is expected to capture all network logs then filter the data. The data is processed then analyze the output whether it’s normal or not normal. The model keeps on learning hence prediction time reduces with time.

4.2 Nature of end product

The datasets which act as a network simulation are run and the alert logs captured are analyzed to check the normal and attack data. The reason of analysis is to assist in finding new methods to curb the anomalies that are not easily detected by the IDS

The model is an algorithm that is trained in a semi-supervised manner to achieve the desired result. The neural network will keep updating according to the behavior of the users they learn from hence it will try to improve the anomaly detectors to capture new threats. The algorithm will also assist with recording of the anomalies detected on the intrusion database.

A classification report is done

	Precision	**recall**	**F1-score**	**support**
0	0.83	1	0.91	9761
1	1	0.87	0.93	15235
Accuracy			0.92	24996
macro avg	0.92	0.94	0.92	24996
weighted avg	0.93	0.92	0.92	24996

- Precision shows accuracy of prediction of true positives
- Recall is the capacity of the classifier to monitor positive instances
- The weighted mean of recall and precision done is referred to as F1 score.

An evaluation of the logs is done to check before and after results of the alert logs. This assists to show how the integration of ANN has led to an improvement of data classification. The improvement was noted with a rise of 1.2% in detection and a rise in classification of 2.1%.

Table 3: Evaluation Table

Dataset		**Actual bytes**	**Rate (packet per second)**	**Unrecognized data**	**ALERTs**
(CICIDS2017)	**Ruleset**	116888	35456.875	216,269	31385
(CICIDS2017)	**AI preprocessor**	116,888	50332.84	104,163	36396

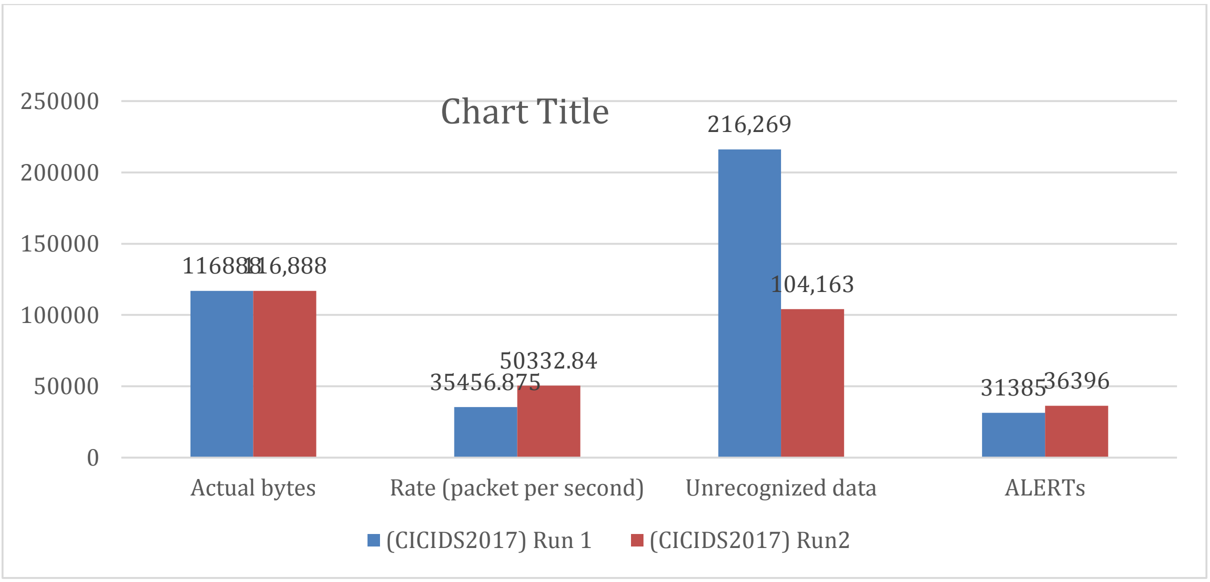


Figure 16: logs Analysis

4.3 Conclusion

The proposed model adds advantages and value to anomaly intrusion detectors by improving their normal way of operation through the set metrics. Anomalies are to be detected as early as possible which assist to protect the network and the sensitive information in the network.

Integration of the ANN model assists to act as backup in better classification of data. This helped with checking the anomalies that are wrongly set and classified by the system. The log files are interpreted to check the improvement of accuracy in terms of data classification.

Finally, ICT officer and administrators will benefit in terms of proper reporting of the anomalies detected. The analysis of the reports assists the administrators to enhance the rules and policies on the network which might assist with continual improvement of the auto encoders. The model assists to less supervise the IDS.

CHAPTER FIVE: REFERENCES

(. S., 2009) (2009). Intrusion Preventing System using Intrusion Detection System Decision Tree Data Mining. *American Journal of Engineering and Applied Sciences*, *2*(4), 721–725. https://doi.org/10.3844/ajeassp.2009.721.725

Alyousef, M. Y., & Abdelmajeed, N. T. (2019). Dynamically detecting security threats and updating a signature-based intrusion detection system’s database. *Procedia Computer Science*, *159*, 1507–1516. https://doi.org/10.1016/j.procs.2019.09.321

Annual, N., Of, C., & Study, C. (2018). *AT-A-GLANCE Unlocking the value of improved cybersecurity protection THE EXPANDING THREAT LANDSCAPE AND NEW BUSINESS INNOVATION*. 2018–2019.

Chandola, V., BANERJEE, A., & KUMAR, V. (2009). Survey of Anomaly Detection. *ACM Computing Survey (CSUR)*, *41*(3), 1–72. https://doi.org/10.1145/1541880.1541882

Jawhar, M. M. T. (2010). *Design Network Intrusion Detection System using Hybrid Fuzzy-Neural Design Network Intrusion Detection System using hybrid Fuzzy-Neural Network Muna Mhammad T . Jawhar detection rate and low false negative . Training and testing data were obtained from th*. *April*.

Lateef, A. A. A., Al-Janabi, S. T. F., & Al-Khateeb, B. (2019). Survey on intrusion detection systems based on deep learning. *Periodicals of Engineering and Natural Sciences*, *7*(3), 1074–1095. https://doi.org/10.21533/pen.v7i3.635

Raza Shah, S. A., & Issac, B. (2017). Performance Comparison of Intrusion Detection Systems and Application of Machine Learning to Snort System. *ArXiv*.

Sani, Y., Mohamedou, A., Ali, K., Farjamfar, A., Azman, M., & Shamsuddin, S. (2009). An overview of neural networks use in anomaly intrusion detection systems. *SCOReD2009 - Proceedings of 2009 IEEE Student Conference on Research and Development, SCOReD*, 89–92. https://doi.org/10.1109/SCORED.2009.5443289

Satam, P. (2017). Anomaly Based Wi-Fi Intrusion Detection System. *Proceedings - 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems, FAS*W 2017*, 377–378. https://doi.org/10.1109/FAS-W.2017.180

Shah, S. A. R., & Issac, B. (2018). Performance comparison of intrusion detection systems and application of machine learning to Snort system. *Future Generation Computer Systems*, *80*(March), 157–170. https://doi.org/10.1016/j.future.2017.10.016

Shah, S. A. R., Issac, B., & Jacob, S. M. (2018). Intelligent Intrusion Detection System Through Combined and Optimized Machine Learning. *International Journal of Computational Intelligence and Applications*, *17*(2). https://doi.org/10.1142/S1469026818500074

Shi, Y., Lei, J., Yin, Y., Cao, K., Li, Y., & Chang, C.-I. (2019). Discriminative Feature Learning With Distance Constrained Stacked Sparse Autoencoder for Hyperspectral Target Detection. *IEEE Geoscience and Remote Sensing Letters*, *16*(9), 1462–1466. https://doi.org/10.1109/lgrs.2019.2901019

Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A Deep Learning Approach to Network Intrusion Detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *2*(1), 41–50. https://doi.org/10.1109/TETCI.2017.2772792

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep Learning Approach for Intelligent

Intrusion Detection System. *IEEE Access*, *7*, 41525–41550. https://doi.org/10.1109/ACCESS.2019.2895334

Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access*, *5*, 21954–21961. https://doi.org/10.1109/ACCESS.2017.2762418