

**INTRUSION DETECTION MODELS USING ENHANCED
DENOISING AUTOENCODERS AND LIGHTGBM CLASSIFIER WITH
IMPROVED DETECTION PERFORMANCE**

SHEIKH ABDUL HAMEED

DOCTOR OF PHILOSOPHY (ENGINEERING)

**LEE KONG CHIAN FACULTY OF ENGINEERING AND SCIENCE
UNIVERSITI TUNKU ABDUL RAHMAN
JUNE 2023**

**INTRUSION DETECTION MODELS USING ENHANCED
DENOISING AUTOENCODERS AND LIGHTGBM CLASSIFIER WITH
IMPROVED DETECTION PERFORMANCE**

By

SHEIKH ABDUL HAMEED

A thesis submitted to the Department of Electrical and Electronic Engineering,
Lee Kong Chian Faculty of Engineering and Science,
Universiti Tunku Abdul Rahman,
In partial fulfillment of the requirements for the degree of
Doctor of Philosophy (Engineering)
June 2023

ABSTRACT

INTRUSION DETECTION MODELS USING ENHANCED DENOISING AUTOENCODERS AND LIGHTGBM CLASSIFIER FOR IMPROVED DETECTION PERFORMANCE

Sheikh Abdul Hameed

An intrusion detection system (IDS) is a software developed to monitor network traffic for suspicious activities to secure data transmission. The conventional IDS strategies are vulnerable to distorted high dimensional network traffic. To overcome this, we proposed an IDS that combines a denoising autoencoder (DAE) and LightGBM classifier. The DAE aims to reduce the distortions in the network traffic by extracting the compressed hidden features representation. The LightGBM classifier aims to classify the samples using the histogram bins of the extracted features with larger gradients, which possibly boost the predictive capacity of the model. To eliminate the deviations in the latent structure, the DAE is enhanced. They are 1. DAE with Jacobian Gradient Norm, which minimizes the larger partial derivatives of the encoder activation 2. DAE with Iterating Thresholding Function, which minimizes the larger magnitude values of the encoder activation weight 3. DAE with Data Pairwise Similarity Weight, which groups the similar data points with strong similarity weight in the encoder activation clusters 4. DAE with Approximated Standard Normal Distribution, which approximates the latent structure to the standard normal distribution using inference strategy. To evaluate the effectiveness of the proposed models, they are experimented using various benchmark datasets. Notice that our proposed models achieve higher detection rate, which outperform the existing IDS models against all the eight commonly used datasets.

ACKNOWLEDGEMENTS

I would like to express my great appreciation to my supervisor, Ts. Dr. Yap Wun She, for his valuable and constructive suggestion during the planning and development of this research work. His willingness to give his time and patient guidance so generously has been very much appreciated. I would also like to thank my co-supervisor, Dr. Moris Ezra Abraham for his enthusiastic encouragement and useful critiques to improve this research work. This thesis could not have been completed without the professional assistance from both of my supervisors. I would like to show my special thanks to my aunty Dr. Mumtaj Begam, who helped me a lot in my research project. I am particularly grateful to my family, faculty and staff for their moral support throughout my graduate study. The research was supported by the Ministry of Higher Education (MoHE) Malaysia through Trans-disciplinary Research Grant Scheme project TRGS/1/2016/UTAR/01/2/2.

APPROVAL SHEET

This thesis entitled **“INTRUSION DETECTION MODELS USING ENHANCED DENOISING AUTOENCODERS AND LIGHTGBM CLASSIFIER WITH IMPROVED DETECTION PERFORMANCE”** was prepared by SHEIKH ABDUL HAMEED and submitted as partial fulfillment of the requirements for the degree of Doctor of Philosophy (Engineering) at Universiti Tunku Abdul Rahman.

Approved by:



(Ts. Dr. YAP WUN SHE)

Supervisor

Department of Electrical and Electronic Engineering
Universiti Tunku Abdul Rahman

Date: 27/6/2023
.....

LEE KONG CHIAN FACULTY OF ENGINEERING AND SCIENCE
UNIVERSITI TUNKU ABDUL RAHMAN

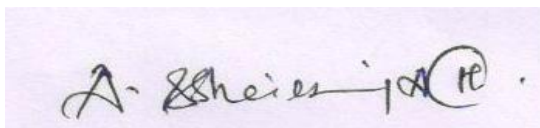
Date: 27/06/2023

SUBMISSION SHEET
SUBMISSION OF THESIS

It is hereby certified that **SHEIKH ABDUL HAMEED** (ID No: 19UED00549) has completed this thesis entitled “INTRUSION DETECTION MODELS USING ENHANCED DENOISING AUTOENCODERS AND LIGHTGBM CLASSIFIER WITH IMPROVED DETECTION PERFORMANCE” under the supervision of Ts. Dr. Yap Wun She (Supervisor) from the Department of Electrical and Electronics Engineering, Lee Kong Chian Faculty of Engineering and Science, and Dr. Ezra Moris Abraham (Co-supervisor) from the Universiti Tunku Abdul Rahman.

I understand that University will upload softcopy of my thesis in PDF format into UTAR Institutional Repository, which may be made accessible to UTAR community and the public.

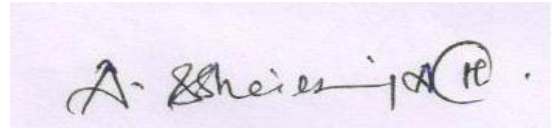
Yours truly,

A handwritten signature in black ink on a light purple background. The signature appears to be 'A. Sheikh' followed by a circled 'H' and a period.

(*SHEIKH ABDUL HAMEED*)

DECLARATION

I hereby declare that the thesis is based on my original work except for quotation and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTAR or other institutions.

A handwritten signature in black ink on a light purple background. The signature reads "A. Sheikh" followed by a stylized circular mark containing the letters "A" and "H".

Name: SHEIKH ABDUL HAMEED

Date: 27/06/2023

TABLE OF CONTENTS

	PAGE
ABSTRACT	i
ACKNOWLEDGEMENT	ii
APPROVAL SHEET	iii
SUBMISSION SHEET	iv
DECLARATION	v
TABLE OF CONTENTS	vi
LIST OF FIGURES	viii
LIST OF TABLES	xiii
LIST OF ABBREVIATIONS	xvii
1.0 INTRODUCTION	1
1.1 Intrusion Detection System	1
1.2 Machine Learning	4
1.3 Problem Statement	12
1.4 Objectives	15
1.5 Contributions	17
1.6 Organization of the Thesis	18
2.0 LITERATURE REVIEW	20
2.1 Supervised Models	20
2.2 Unsupervised Models	29
2.3 IDS with Classifiers and without Features Selection	45
2.4 IDS with Classifiers and Features Selection	49
2.5 IDS with Classifiers and Features Extraction	56

3.0 RESEARCH METHODOLOGY	64
3.1 Enhanced IDS Design	64
3.1.1 Data-Preprocessing	65
3.1.2 Performance Metrics	66
3.2 Proposed Enhanced DAE Models	68
3.2.1 Enhanced DAE Model 1	70
3.2.2 Enhanced DAE Model 2	74
3.2.3 Enhanced DAE Model 3	78
3.2.4 Enhanced DAE Model 4	80
3.3 Hyperparameter Tuning	87
4.0 RESULTS AND DISCUSSIONS	89
4.1 Datasets	89
4.2 Experimental Setup	96
4.3 Results and Discussions	98
4.4 Performance Comparison	164
4.5 Overview of the Classification Results	177
5.0 CONCLUSION AND FUTURE WORK	184
5.1 Conclusion	184
5.2 Future Work	185
REFERNCES	186
LIST OF PUBLICATIONS	197
APPENDIX: FEATURES OF THE DATASETS	198

LIST OF FIGURES

Figure		Page
Figure 1.1	Intrusion Detection System	1
Figure 1.2	Network and Host based IDS models	3
Figure 1.3	Machine learning process	5
Figure 1.4	The steps involved in a machine learning model	7
Figure 1.5	The machine learning process in a network security system	9
Figure 1.6	<u>The problem of distortions in a benchmark Dataset</u>	13
Figure 1.6a	Normal Traffic Signal Vs Noise	14
Figure 1.6b	The performance graph with noise	14
Figure 1.7	<u>The deviated latent structure of the DAE for a benchmark dataset</u>	15
Figure 2.1	The supervised machine learning process	20
Figure 2.2	The light gradient boosting process flow	24
Figure 2.3	Histogram-optimized strategy	27
Figure 2.4	Leaf-wise strategy	29
Figure 2.5	The unsupervised machine learning process	29

Figure 2.6	The block-diagram of an autoencoder	31
Figure 2.7	The architecture of an ANN	32
Figure 2.8	The output computation of a neuron	34
Figure 2.9	The convex optimization cost function	39
Figure 2.10	Architecture of an autoencoder	41
Figure 2.11	<u>Architecture of denoising autoencoder</u>	43
Figure 3.1	<u>The overall design of the enhanced hybrid IDS models</u>	64
Figure 3.2	<u>Architecture of an enhanced DAE model 1</u>	70
Figure 3.3	<u>Architecture of an enhanced DAE model 2</u>	74
Figure 3.4	<u>Architecture of an enhanced DAE model 3</u>	80
Figure 3.5	<u>Architecture of generative DAE</u>	83
Figure 3.6	The continuous latent representation of the generative DAE	87
Figure 4.1	<u>The detection loss of the base hybrid model for the benchmark datasets</u>	102
Figure 4.2	The detection loss of the base hybrid model is shown in graphical bar	103
Figure 4.3	The detection rate of the base hybrid model is shown in graphical bar	104
Figure 4.4	The presence of distortions in the performance graph of a benchmark dataset	106

Figure 4.5	<u>The presence of hidden distortions in a benchmark dataset</u>	107
Figure 4.7	<u>The detection loss of the enhanced hybrid model 1 for the benchmark datasets</u>	111
Figure 4.8	The detection loss of the enhanced hybrid model 1 is shown in graphical bar	112
Figure 4.9	The detection rate of the enhanced hybrid model 1 is shown in graphical bar	113
Figure 4.10	<u>The detection loss of the enhanced hybrid model 2 for the benchmark datasets</u>	118
Figure 4.11	The detection loss of the enhanced hybrid model 2 is shown in graphical bar	119
Figure 4.12	The detection rate of the enhanced hybrid model 2 is shown in graphical bar	120
Figure 4.13	<u>The detection loss of the enhanced hybrid model 3 for the benchmark datasets</u>	126
Figure 4.14	The detection loss of the enhanced hybrid model 3 is shown in graphical bar	127
Figure 4.15	The detection rate of the enhanced hybrid model 3 is shown in graphical bar	128
Figure 4.16	<u>The detection loss of the enhanced hybrid model 4 for the benchmark datasets</u>	133
Figure 4.17	The detection loss of the enhanced hybrid model 4 is shown in graphical bar	134
Figure 4.18	The detection rate of the enhanced hybrid model 4 is shown in graphical bar	135
Figure 4.19	<u>The removed distortions in a benchmark dataset</u>	137

Figure 4.20	<u>The undeviated clustered latent structure by the enhanced DAE</u>	137
Figure 4.21	The classification performance of the proposed models for IoTID	142
Figure 4.22	The classification performance of the proposed models for BoT- IoT	146
Figure 4.23	The classification performance of the proposed models for CIC-IDS	151
Figure 4.24	The classification performance of the proposed models for CIDDS-001	155
Figure 4.25	The classification performance of the proposed models for ISCX-URL	159
Figure 4.26	The classification performance of the proposed models for CIDDS-002	160
Figure 4.27	The classification performance of the proposed models for ISCX-TOR	161
Figure 4.28	The classification performance of the proposed models for UNSW-NB	163
Figure 4.29	The detection loss comparison for CIDDS-001 is shown in graphical bar	165
Figure 4.30	The detection rate comparison for CIDDS-001 is shown in graphical bar	165
Figure 4.31	The detection loss comparison for CIDDS-002 is shown in graphical bar	166
Figure 4.32	The detection rate comparison for CIDDS-002 is shown in graphical bar	167
Figure 4.33	The detection loss comparison for IoTID is shown in graphical bar	168
Figure 4.34	The detection rate comparison for IoTID is shown in graphical bar	168
Figure 4.35	The detection loss comparison for ISCX-URL is shown in graphical bar	169

Figure 4.36	The detection rate comparison for ISCX-URL is shown in graphical bar	170
Figure 4.37	The detection loss comparison for CIC-IDS is shown in graphical bar	171
Figure 4.38	The detection rate comparison for CIC-IDS is shown in graphical bar	172
Figure 4.39	The detection loss comparison for ISCX-TOR is shown in graphical bar	173
Figure 4.40	The detection rate comparison for ISCX-TOR is shown in graphical bar	174
Figure 4.41	The detection loss comparison for UNSW-NB is shown in graphical bar	175
Figure 4.42	The detection rate comparison for UNSW-NB is shown in graphical bar	175
Figure 4.43	The detection loss comparison for BoT-IoT is shown in graphical bar	176
Figure 4.44	The detection rate comparison for BoT-IoT is shown in graphical bar	177
Figure 4.45	<u>The detection rate of the minority classes for the benchmark datasets</u>	182

LIST OF TABLES

Table	Title	Page
Table 3.1	The categorical values of the attribute	65
Table 3.2	The encoded values of the categorical attribute	65
Table 3.3	Confusion matrix for a binary classification problem	66
Table 4.1	The software setup for the proposed models	96
Table 4.2	The hardware setup for the proposed models	96
Table 4.1a	The hyperparameter tuning of DAE for different datasets for all the proposed models.	99
Table 4.2a	The hyperparameter tuning of LightGBM for different datasets for all the proposed models	99
Table 4.3	The classification performance of the base hybrid model	101
Table 4.4	The class-wise performance of the base hybrid model	105
Table 4.5	The classification performance of the enhanced hybrid model 1	110
Table 4.6	The class-wise performance of the enhanced hybrid model 1	114
Table 4.7	The classification performance of the enhanced hybrid model 2	117
Table 4.8	The class-wise performance of the enhanced hybrid model 2	121

Table 4.9	The classification performance of the enhanced hybrid model 3	125
Table 4.10	The class-wise performance of the enhanced hybrid model 3	129
Table 4.11	The classification performance of the enhanced hybrid model 4	132
Table 4.12	The class-wise performance of the enhanced hybrid model 4	136
Table 4.13	The confusion matrix for the Mirai class	138
Table 4.14	The confusion matrix for the DoS class	139
Table 4.15	The confusion matrix for the Scan class	139
Table 4.16	The confusion matrix for the MITM ARP Spoofing class	140
Table 4.17	The confusion matrix for the Normal class	141
Table 4.18	The confusion matrix for the DDoS class	142
Table 4.19	The confusion matrix for the DoS class	143
Table 4.20	The confusion matrix for the Reconnaissance class	144
Table 4.21	The confusion matrix for the Theft class	144
Table 4.22	The confusion matrix for the Normal class	145
Table 4.23	The confusion matrix for the DoS Hulk class	146
Table 4.24	The confusion matrix for the DoS Golden Eye class	147
Table 4.25	The confusion-matrix for the DoS Slow Loris class	148
Table 4.26	The confusion-matrix for the DoS Slow Http Test class	148

Table 4.27	The confusion-matrix for the Heartbleed class	149
Table 4.28	The confusion-matrix for the Normal class	150
Table 4.29	The confusion-matrix for the DoS class	151
Table 4.30	The confusion-matrix for the Brute Force class	152
Table 4.31	The confusion-matrix for the Port Scan class	153
Table 4.32	The confusion-matrix for the Ping Scan class	153
Table 4.33	The confusion-matrix for the Normal class	154
Table 4.34	The confusion-matrix for the Malware class	155
Table 4.35	The confusion-matrix for the Defacement class	156
Table 4.36	The confusion-matrix for the Spam class	157
Table 4.37	The confusion-matrix for the Phishing class	157
Table 4.38	The confusion-matrix for the Normal class	158
Table 4.39	The confusion matrix of CIDDS-002	159
Table 4.40	The confusion matrix of ISCX-TOR	160
Table 4.41	The confusion matrix of UNSW-NB	162
Table 4.42	Proposed model result for CIDDS-001 with different existing schemes	164
Table 4.43	Proposed model result for CIDDS-002 with different existing schemes	166

Table 4.44	Proposed model result for IoTID with different existing schemes	167
Table 4.45	Proposed model result for the ISCX-URL with different existing schemes	169
Table 4.46	Proposed model result for the CIC-IDS with different existing schemes	171
Table 4.47	Proposed models result for ISCX-TOR with different existing schemes	173
Table 4.48	Proposed models result for UNSW-NB with different existing schemes	174
Table 4.49	Proposed models result for BoT-IoT with different existing schemes	176
Table 4.50	The main functionalities of the DAE and the four enhanced DAE models	180

LIST OF ABBREVIATIONS

ADAM	Adaptive Moment Estimation
AE	Autoencoder
AIDS	Anomaly Intrusion Detection System
ANN	Artificial Neural Network
ARP	Address Resolution Protocol
BI	Bayesian Inference
B-Profile	Beta Profile
CFS	Correlation Features Selection
CIDDS	Coburg Intrusion Detection Data Sets
CNN	Convolutional Neural Network
CSV	Comma Separated Values
DAE	Denoising Autoencoder
DDoS	Distributed Denial of Service
DMZ	Demilitarized zone
DoS	Denial of Service

FTP	File Transfer Protocol
GPRS	General Packet Radio Service
HIDS	Host Intrusion Detection System
HTTP	Hyper Text Transfer Protocol
IDE	Integrated Development Environment
IG	Information Gain
IoT	Internet of Things
IP	Internet Protocol
ISCX	Installation Support Center of Expertise
ITF	Iterative Thresholding Function
JCVN	Jacobian Gradient Vector Norm
KDD	Knowledge Discovery in Databases
KL-Divergence	Kullback–Leibler Divergence
KNN	K-Nearest neighbor
LR	Logistic Regression
LS-SVM	Least Square Support Vector Machine

LightGBM	Light Gradient Boosting
ML	Machine Learning
MLP	Multilayer Perceptron
MITM	Man in the Middle Attack
NAG	Nester Accelerated Gradient
NB	Naïve Bayes
NDAE	Non-Symmetric Autoencoder
NIDS	Network Intrusion Detection System
NIMS	Network Information Management and Security
NSL	Network Security Laboratory
PCA	Principal Component Analysis
PIN	Personal Identification Number
P2P	Peer to Peer
RBF	Radial Basis Function
RELU	Rectified Linear Unit
RFE	Recursive Feature Elimination

RF	Random Forest
SDN	Software Defined Network
SGD	Stochastic Gradient Descent
SHAP	Shapely Additive Explanation
SIDS	Signature Intrusion Detection System
SMOTE	Synthetic Minority Oversampling Technique
SSH	Secure Shell
SSL	Secure Socket Layer
SVM	Support Vector Machine
TOR	The Onion Router
UDBS	Uniform Distribution based Balancing Scheme
URL	Uniform Resource Locator

CHAPTER 1

INTRODUCTION

1.1 Intrusion Detection System

Rapid technological developments necessitate increased usage of the internet, which in turn results in the growth of big data, cloud computing, internet of things (IoT) and software defined networks. Thus, huge amount of data has been daily flowing in the network system. Any attack behaviour that compromises the security of a network system is called an intrusion. The attackers try to intrude the network traffic and may possess security issues such as malicious interruption, theft, corruption of data ([Zarpelao et al., 2017](#)) and cause severe damage to the valuable assets. The intruders grow in numbers, that are highly vulnerable to efficient and secure data transmission in the network (Hanson and Hunt, 2005). Thus, Intrusion Detection System (IDS) is an essential component of the network security and is shown in Figure 1.1.

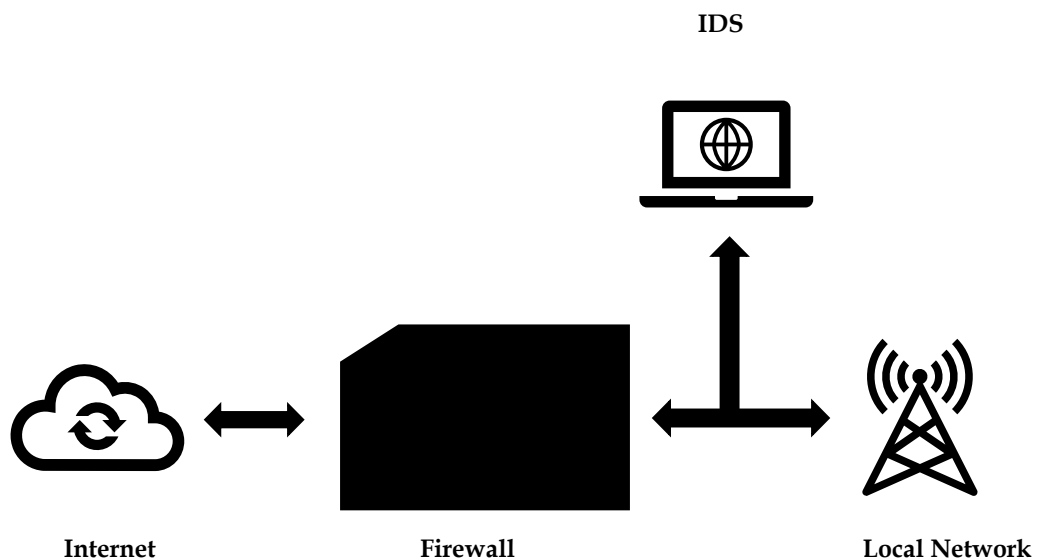


Figure 1.1: Intrusion Detection System ([Denning, 1987](#))

IDS ([Denning, 1987](#)), an automated software that monitors incoming and outgoing network traffic to find any unusual behavior in the network including attack attempts and generates an alarm to signal network admin to take preventive actions and helps in securing the network system. It can be used for detecting any type of harmful and dangerous attackers. The IDS is classified into two types (Verwoerd and Hunt, 2002), one is host-based intrusion detection system (HIDS) and another is network-based intrusion detection system (NIDS).

As shown in Figure 1.2, HIDS monitors security issues related to a single host. It monitors the internal environment of a host system such as resources, system files, applications, audit records, mails, OS log files, and system tables etc. It operates on a single system and provides security to its servers of the system

As shown in Figure 1.2, NIDS monitors the entire inbound and outbound network traffic, i.e., data packets travelling in the network traffic to find any unusual behaviour within it to prevent it from illegal activities. It can be placed anywhere within the network (e.g., can be deployed in demilitarized zone, as part of an intelligent firewall, virtual private network servers, remote access servers and wireless network access points) and monitor the traffic activities.

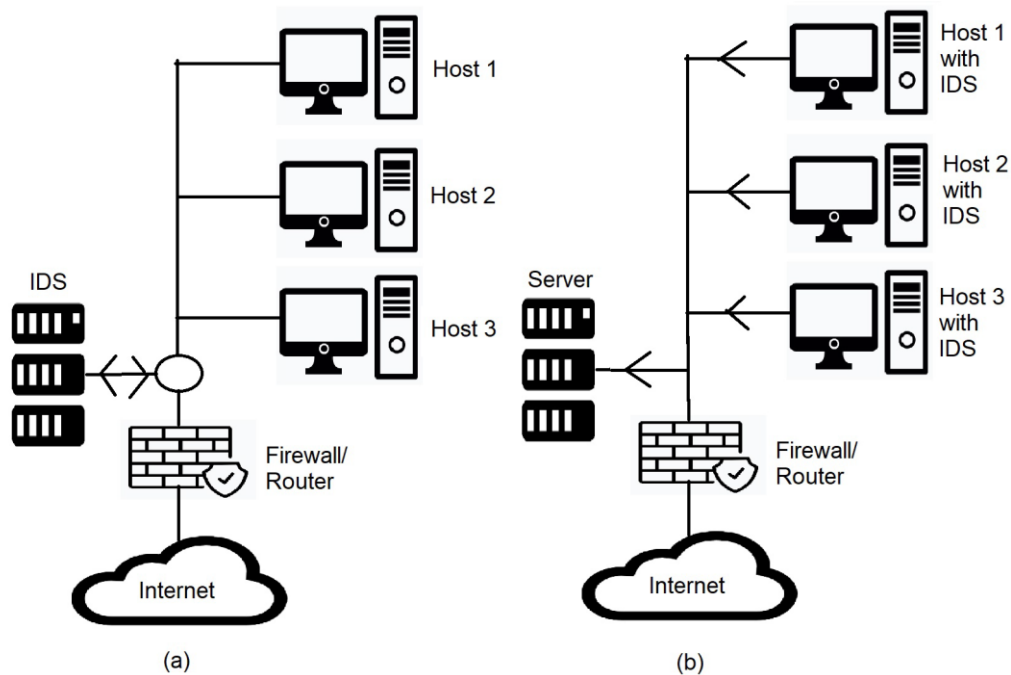


Figure 1.2: Network and Host Based IDS Models

NIDS is also sub-classified (Depren *et al.*, 2005) as a signature-based detection system and an anomaly-based detection system. SIDS is based on signatures such as metadata and file fingerprints. It compares signatures of the network activity with previously stored signatures in the library database. The signatures need to be updated periodically within the database. It is not suitable for detecting any new patterns that are not stored in the library database. On the other hand, AIDS monitors the entire network operations and predicts the intruders by observing any unusual behaviour in the network traffic that deviates from the normal behaviour. The baseline of the normal behaviour is already defined in the IDS. It is very efficient in detecting any new abnormal patterns.

1.2 Machine Learning

Artificial intelligence was defined by Minsky (1968) as “The science of making machines do things that would require intelligence if done by men”. Another similar definition for the field of artificial intelligence provided by Chollet (2017) was that it is simply “The effort to automate intellectual tasks normally performed by humans.”

According to Samuel (1959), machine learning was described as “It is a field of study that gives computers the ability to learn without being explicitly programmed to”. Meanwhile, Mitchell (1997) defined a machine learning algorithm as follows: “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E ”.

In our thesis, task T is to classify the category of network flow activity, computer program is the NIDS model that we design, experience E is network traffic used as input to train the model. Each sample is defined as the set of features in certain quantitative measurements, i.e. $x_i \in R_n$, where x_i denotes the training samples corresponding to an individual feature R_n . Lastly, performance measure P is the standard metrics that we use to evaluate our model.

Machine learning ([Awad and Khanna, 2015](#)) is a branch of artificial intelligence that makes the system to learn from past occurrences and predicts the outcome for new observations. It is visually shown in Figure 1.3. The prediction ability of a machine learning is measured in terms of standards metrics.

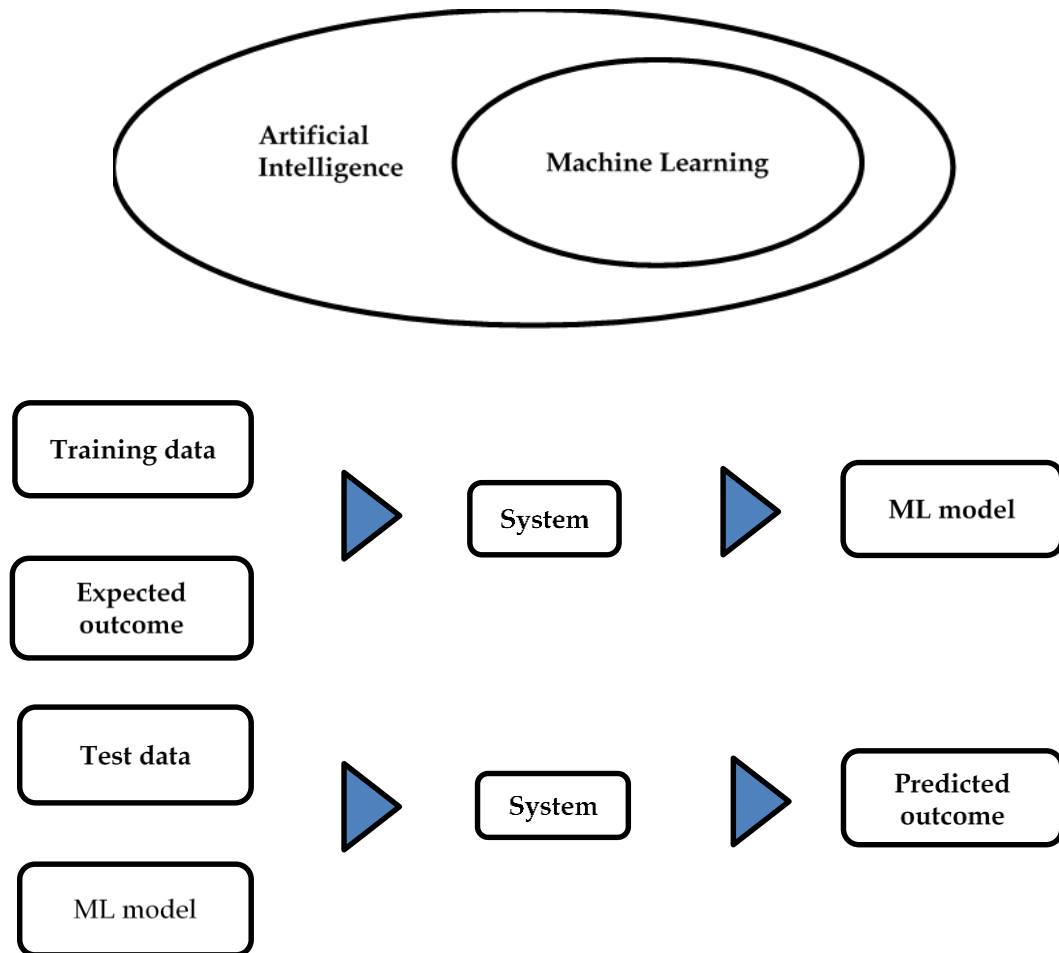


Figure 1.3: Machine Learning Process

The machine learning algorithm is broadly classified into two types (Mitchell, 1997):

1. Supervised machine learning and
2. Unsupervised machine learning.

The supervised machine learning technique uses labelled data to train the IDS model, and then perform the classification task. The supervised ML is mainly used for the

classification problem. It learns to map input data to output class labels by proper mapping function and intent to map a suitable category for new inputs. The perturbed high dimensional non-linear features manifold is highly challenging for the supervised machine learning classifiers. They are highly vulnerable to noise and corruptions in the high-dimensional data records otherwise called adversarial samples, that affect the performance of those classifiers.

The unsupervised learning techniques use unlabelled data and are mainly used as the dimensionality reduction strategy. It can cluster similar data points that belong to identical class labels and thereby it can discover the compressed low-dimensional hidden patterns and structure from the high dimensional input, which is the main source for the classification problem. It can possess a better representation of the original input data.

Machine learning is applied in different research fields (Sharda *et al.*, 2018) such as processing natural language, speech recognition, computer vision, audio recognition, machine translation and social network filtering, customer relationship management, banking, retail and logistics, manufacturing and production, insurance, computer hardware and software, government and defence, travel industry, healthcare and medicine, entertainment industry, homeland security, law enforcement and sports. Figure 1.4 shows an overview of the machine learning process. The processes involved ([Fayyad *et al.*, 1996](#)) in the machine learning process are listed as follows:

1. Data Collection: The benchmarking network security datasets are collected from various online sources.

2. **Data Pre-processing:** The raw datasets that have been collected from the online web sources should be transformed into a proper format that the machine learning model should accept.
3. **Feature Extraction/Dimensionality Reduction:** As the cyber security datasets are of high-dimensional, it suffers from the “curse of dimensionality” that will adversely affect the performance of the machine learning model. To reduce the dimensionality of the dataset and extract hidden patterns from the raw data, the dimensionality reduction process is needed. The dimensionality reduction task belongs to the unsupervised learning category.
4. **Model Training:** The machine learning classifier is trained using the patterns received from the dimensionality reduction stage. The overlearning of the model makes the model stuck in overfitting whereas under learning of the model will lead to underfitting. Both two issues would decrease the performance of the machine learning model.
5. **Model Testing and Deployment:** The trained model is tested, evaluated and measured in terms of standard quality metrics and the model with best predictive ability can be deployed and implemented as a real-time model.

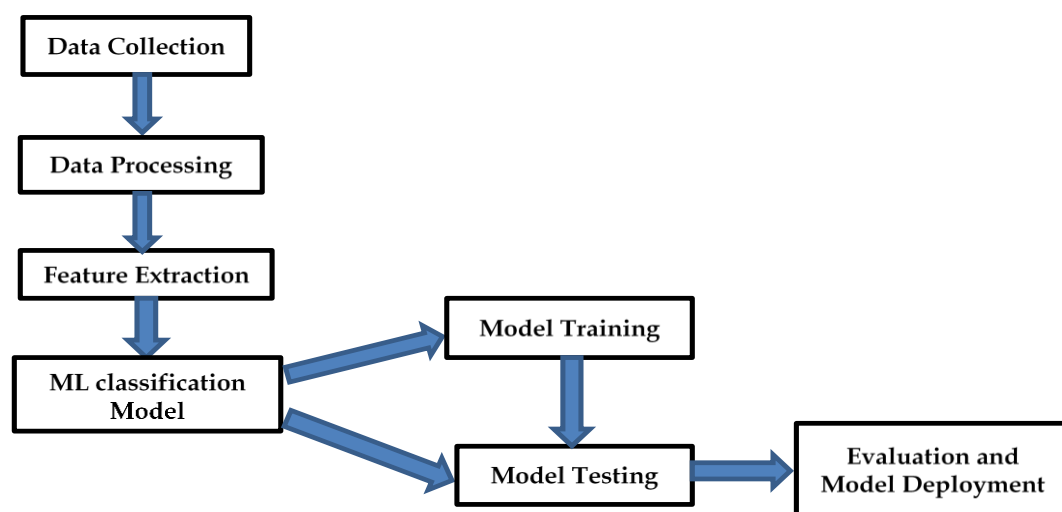


Figure 1.4: The Steps Involved in a Machine Learning Model

With the rapid advancement in areas of network technology, machine learning and data mining techniques can be used to deal with challenges of cyber security issues (Tsui, 2009). Machine learning can be applied in the research areas of signature detection, anomaly detection, scan detection, network traffic profiling and privacy-preserving data mining.

The task of predicting an unusual behaviour in the network traffic is termed as a classification problem that falls under the supervised machine learning technique (Kotsiantis *et al.*, 2007). It is a technique used in machine learning that the learning algorithm aims to match a suitable category label for the input samples through a proper mapping function. For a function $y = f(x)$, the machine learning model maps the input function x to an output class label y . There are two kinds of classification problems based on the category of labels present in the dataset. In both classification tasks, the goal of the learning model is to map a function f to k labels where $k = 2$ for binary classification and $k \geq 3$ for multi-classification. When $k = 2$, the dataset contains only two labels and the model can classify the records as either normal or attack. When $k \geq 3$, the dataset has multiple target labels where the model can classify different categories of attacks.

To learn network traffic activities, input is given in the form of a dataset. The dataset describes the characteristics of any network traffic samples. These samples are divided into training and test sets to train and test the proposed classifier. A classifier is trained by samples with known outcome and its performance is evaluated using new samples that are not existing in the training set. When a test set is applied to a machine learning classifier, it can predict the outcome of new samples. The process is picturized in Figure 1.5. The performance analysis of

the classification problem is based on the number of correct predictions on the test set and is measured in terms of standard quality metrics

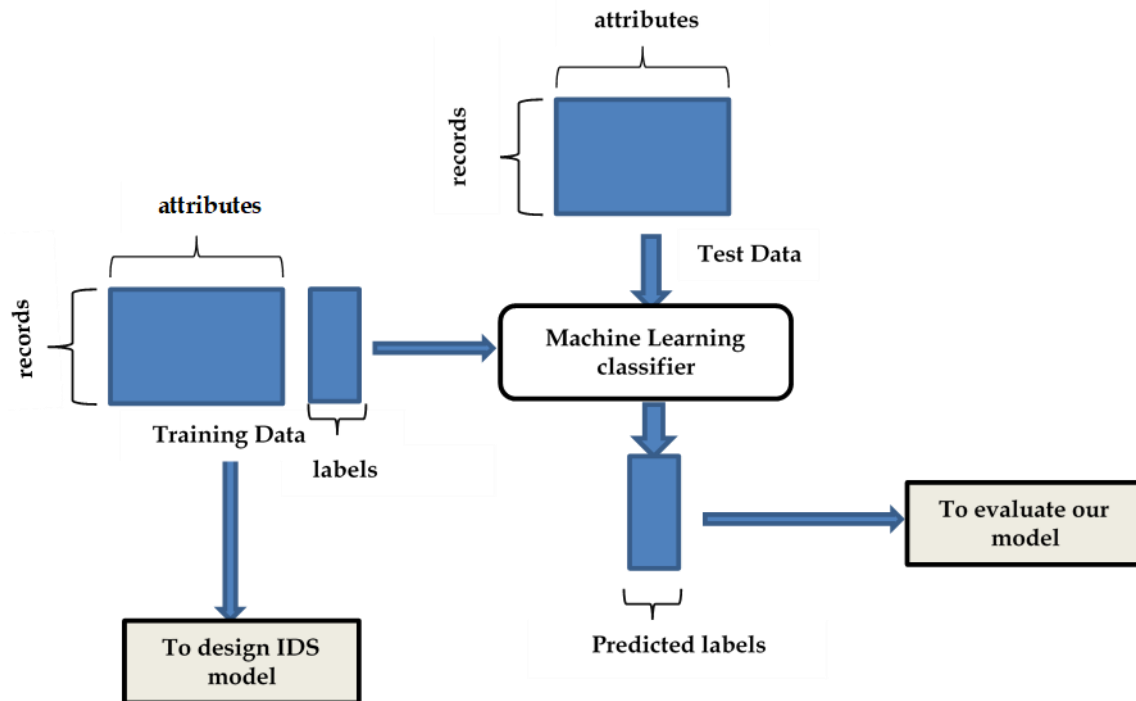


Figure1.5: The Machine Learning Process in a Network Security System

In a machine learning approach, a classifier is trained using a machine learning algorithm on a dataset of normal and abnormal traffic patterns. A trained model may subsequently be employed to predict suspicious traffic in real-time. Such a dataset typically considers each pattern across several features of an associated target class, which denotes whether the pattern corresponds to normal or abnormal usage. Further training on new instances allows the classifier to adapt to the current network state. Thus, the main focus is to increase the detection performance of the deployed intrusion detection system.

Various machine learning techniques ([Mishra et al., 2018](#)) have been used to build different IDS models. Some of the conventional machine learning techniques involved are Support Vector Machine (SVM) ([Snehal et al., 2010](#)), K-Nearest Neighbour (KNN) ([Liao &](#)

Vemuri, 2002), Naive Bayes (NB) (Panda & Patra, 2007), Multilayer Perceptron (MLP) (Lee & Heinbuch, 2001), Random Forest (RF) (Zhang *et al.*, 2008), Decision Trees (DT) (Snehal *et al.*, 2010) , Logistic Regression (LR) (Shukla *et al.*, 2017).The conventional machine learning classifiers resulted in low detection performance due to the high dimensional traffic features of the input dataset, while, noise and corruptions in the input dataset will still affect the stability and performance of an IDS model. The conventional machine learning strategies are vulnerable to distortions in the network traffic, that lead to deviations in the extracted patterns, while the IDS classifiers on learning it, misclassify the input samples and make wrong classification results, which give a low classification rate and high classification loss.

Although there are so many technologies to prevent the intrusion, still the network is vulnerable to so many undetected attacks. There are many ways in which a cyber-attack can be performed. Therefore, to prevent such attacks, there are many IDS suggested and implemented by researchers, still, there is a room for improvement. Many conventional IDS models give lower detection performance due to the perturbations in the network traffic. We aim to remove the distortions in the network traffic by the DAE. The gaussian noise in the input layer of the DAE enforces it to reduce the distortions by extracting the robust compressed hidden patterns, where those patterns are used by the LightGBM classifier to classify the intrusive samples. The classifier uses the histogram bins of the extracted patterns with larger gradients to construct the classification model, instead of evaluating every single feature value, which can boost the predictive capacity of our IDS model. The highly efficient IDS is proposed and designed by combining DAE and LightGBM to monitor the network traffic activities to assure secure data communications.

To evaluate the effectiveness of our proposed model, the model is tested using various benchmark datasets for both binary classification and multi-classification tasks. The proposed base hybrid model shows better detection rate. However, still the predictive capacity of the model is affected, since the representation capacity of the DAE deviates from the original input patterns due to the removed hidden distortions in the latent space, which affects the features learning capacity and predictive capacity of the model. To suppress the deviations in the latent structure, thereby the model could completely group similar data points that belong to identical classes in each clusters, and accelerate the features learning capacity of our model, we propose to insert some of the additional novel regularization strategies on the encoder side of the DAE and develop enhanced DAE models. As machine learning is a stochastic process, there is no free lunch theorem in it. So, we trail and test four novel strategies separately. The strategies have its own unique mathematical algebraic properties in removing the deviations and grouping similar data points of the identical classes in each concerned cluster, thus enhancing the features learning capacity of our IDS model. The patterns extracted from the latter models are fed to the LightGBM classifier for the attack prediction task, which enhance the predictive capacity of our IDS model.

The proposed enhanced models show better detection performance improvement in terms of detection rate, detection-loss, precision, accuracy, and f1-score over eight benchmark datasets including CIDDS-001, CIDDS-002, ISCX-URL2016, UNSW-NB15, CIC-IDS-2017, ISCX-TOR2016, BoT-IoT2018, and IoTID2020 for both binary classification and multi-classification tasks as compared to other existing IDS. The CIDDS-002, ISCX-TOR, UNSW-NB are involved in bi-classification tasks whereas CIDDS-001, CIC-IDS, ISCX-URL, BoT-IoT, IoTID are involved in multiclassification tasks. The datasets are always highly

imbalanced. i.e. Each dataset contains certain category with large number of samples and certain category with a smaller number of samples and so the datasets contain unbalanced classes. The traditional methods are very much sensitive to the imbalanced nature of the datasets, giving lower detection score for the minority category.

The patterns extracted by the enhanced DAE models are more descriptive, informative and discriminative in classifying the samples by the LightGBM classifier which boost the predictive capacity of our NIDS models. The enhanced DAE models and LightGBM together are blended to form robust, stable yet lighter IDS models. Lastly, our proposed schemes have higher learning and predictive capacity, that optimize the generalization capacity of our models and can perform well against the minority classes when a dataset contains unbalanced classes in terms of the number of samples.

1.3 PROBLEM STATEMENT

The conventional IDS models are highly vulnerable to the perturbations in the high-dimensional network traffic. The IDS models on learning those distortions, extract the deviated latent representations, and misclassify the input samples and give wrong classification results. i.e. the intrusive traffic is wrongly predicted as normal and fails to generate and intimate an alarm to the network security administrator. On the other hand, the normal traffic is wrongly predicted as intrusive and generates an unwanted false alarm and threatens the network administrator, which both two cases highly affect the prediction performance and the security of the model. More importantly, the deviations in the latent structure i.e. deviated data points lead to gaps among the similar data points, such that the data points that belong to identical classes are distant apart and are not properly and completely grouped together in their

concerned clusters of the latent structure, which highly affect the adversarial robustness, representation capacity and predictive capacity of the model.

The impact of the deviations on the latent structure of the model are shown below:

1. Accelerate the underfitting and overfitting problem.
2. Affects the learning and predictive capacity of the model, which fails to optimize the generalization capacity of the model.
3. Inability to deal with the imbalanced nature of the high dimensional network traffic.
4. Produce a biased result.

The scatter plot graph that illustrates the problem of noise and corruptions (distorted samples) in one of the benchmarking datasets are shown below in Figure 1.6.

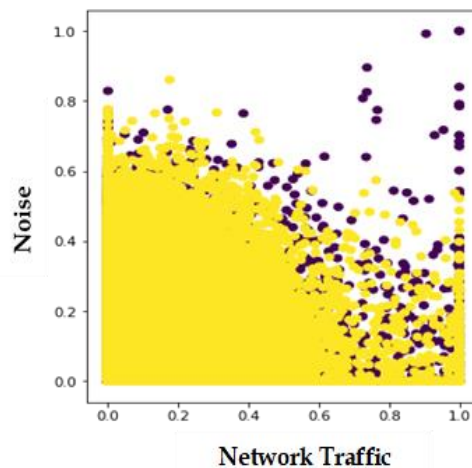


Figure 1.6: The Problem of Distortions in a Benchmark Dataset

The black colour denotes the distorted samples from the ordinary network traffic samples. The noisy data points generally have larger gradients and denser weights, which do not easily concentrate with the normal traffic data points on the original manifold as shown in Figure 1.6. It is noted from the figure that the noisy data points are farther away from the normal traffic points due to larger variance and do not concentrate with the normal traffic points on the original manifold.

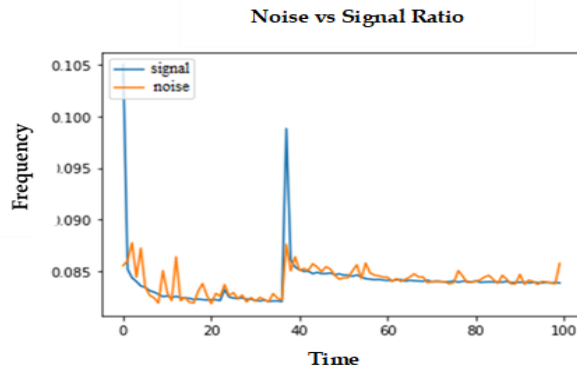


Figure 1.6a Normal Traffic Signal Vs Noise

The following Figure 1.6a shows how the noise deviates from the normal traffic signals in one of our benchmarking datasets. The normal traffic (blue) signals are stable, smooth and possess little variability, while the noise swings wildly and unpredictably from one value to another and obscure the normal traffic signal.

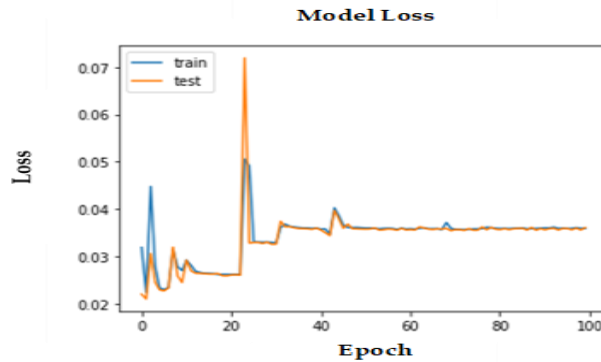


Figure 1.6b: The performance graph with noise

As seen from the Figure 1.6b, the performance graph with noise has no uniformity in the curve. The detection loss of a benchmark dataset initially converges at 0.02 and starts sudden spike of increasing to 0.05 and then decreasing till the loss saturated at 0.04. This confirms the presence of distortions in the network traffic. The noise is the cause for non-uniformity in the performance graph and there are severe fluctuations, ups and down (spikes) and the graph struck in severe overfitting issues since the model tries to overlearns the noise as patterns and starts memorizing and generalizing it.

The following Figure 1.7 shows the deviations i.e. deviated latent structure of the DAE for one of the benchmark datasets, are shown below

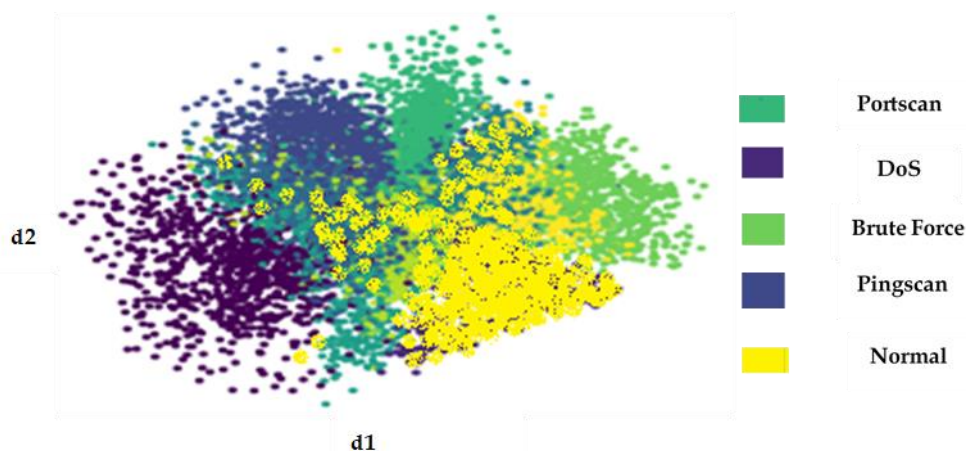


Figure 1.7: Deviated Latent Structure of the DAE for a Benchmark Dataset

The problem of gaps among the similar data points and those similar data points are not grouped in the respective clusters of the latent space of the DAE are shown in Figure 1.7.

1.4 OBJECTIVES

To remove the distortions in the network traffic and suppress the deviations in the extracted latent structure, thereby enhancing the adversarial robustness, representation capacity and predictive capacity of our model, we aim to propose, use and test DAE and enhanced DAE models with the LightGBM classifier. Firstly, we aim to use the DAE model with the LightGBM classifier and develop a base hybrid model. To overcome the shortcomings in the DAE, later it is enhanced to be the enhanced DAE models and associated with the LightGBM classifier and develop four enhanced hybrid models and evaluate the effectiveness of the proposed models on different broadly known network security datasets and so later they can be deployed as a real time models in any industrial network traffic.

The list of proposed models is given below:

- Base Hybrid Model----->DAE with LightGBM

- Enhanced Hybrid Model 1 ----->DAE with Jacobian Gradient Norm and LightGBM classifier
- Enhanced Hybrid Model 2 ----->DAE with Iterative Thresholding Function and LightGBM classifier
- Enhanced Hybrid Model 3 -----> DAE with Data Pairwise Similarity Distance Weight and LightGBM classifier
- Enhanced Hybrid Model 4 -----> DAE with Approximated Standard Normal Distribution using Inference Strategy and LightGBM classifier

The DAE is an enhanced version of a traditional autoencoder by the usage of gaussian noise in the input layer of an autoencoder. The gaussian noise could reduce the distortions in the network traffic and extract the compressed hidden patterns, where it functions and gives only partial regularization and robustness on the encoder activation to extract the hidden patterns i.e. partial robust patterns. The enhanced DAE is an enhanced version of a denoising autoencoder with the insertion of proposed novel strategies on the encoder activation of the DAE. Along with gaussian noise, the proposed strategies give complete regularization and robustness on the encoder activation to remove the deviations in latent dimensions and extract the compressed core-intrinsic structure and salient descriptive primitive patterns from the high-dimensional network traffic thereby improving the adversarial robustness and representation capacity of the DAE and boost up the predictive capacity of the model. Subsequently, the LightGBM classifier uses the histogram bins of the extracted patterns with larger gradients and avoids using each and every single feature value which possibly boost up the predictive capacity of the model.

1.5 CONTRIBUTIONS

The contribution of the thesis is depicted as follows:

- The overall contribution of the DAE and the four enhanced DAE models are removing the deviations in the latent structure and enhancing the quality of the features extracted. Improving the quality of the features extracted boost up the predictive capacity of our attack detection model. Thus, the patterns extracted from the enhanced DAE models are associated with LightGBM classifier to develop four enhanced hybrid models, which all the four models give higher detection rate with minimum detection loss and enhance the predictive capacity of the model.
- The first contribution is the proposal of base hybrid model consisting of DAE and LightGBM. The DAE reduces the distortions from the network traffic by extracting the patterns form the datasets and reduce its dimensionality, where it gives only partial regularization and partial robustness. The classifier LightGBM enhance the predictive performance of the model.
- The second contribution is the enhanced hybrid model 1 consisting of enhanced DAE 1 and LightGBM. The enhanced DAE 1 is the proposal of jacobian gradient norm on the encoder of the DAE. The strategy suppresses the deviations in the latent manifold by imposing squishing pressure on the larger partial derivatives of the jacobian matrix of the encoder activation.
- The third contribution is the enhanced hybrid model 2 combining enhanced DAE 2 with LightGBM. The enhanced DAE 2 is the proposal of iterative threading fuction on the encoder activation of the DAE. This strategy suppresses the

deviations by enforcing the magnitude of the encoder weight matrix to be sparse by firing only the n-strongest encoder neurons.

- The fourth contribution is the enhanced hybrid model 3 combining enhanced DAE 3 with LightGBM classifier. The enhanced DAE 3 is the proposal of data-pairwise similarity weight on the DAE. This strategy eradicates the deviations by directly grouping the similar data points of the identical class with strong similarity weight and filtering out the dissimilar data points with least similar weight in the distinctive clusters of the encoder structure.
- The fifth contribution is the enhanced hybrid model 4 combining enhanced DAE 4 with LightGBM classifier. The enhanced DAE 4 is the proposal of approximated standard normal probability distribution with inference strategy on the encoder activation of the DAE. This strategy eradicates the deviations by making the latent space to be continuous and complete by approximating it to the standard normal distribution.

1.6 ORGANIZATION OF THE THESIS

The overall structure of the dissertation are as follows:

- Section 1 is the introductory part that explains the security issues in the current network technology, need for intrusion detection system and the categories of the IDS models. It also elaborates the concept of machine learning in IDS, and steps involved in it. Section 1 also conveys about problem statement and research gap that conveys the shortcomings in the existing models and the necessity for the proposed IDS models to resolve those shortcomings. It also contains the objectives part that briefs about the objectives of this dissertation to design and evaluate the novel hybrid models and

contribution of all the four proposed hybrid models in improving the feature learning capacity and predictive capacity of the model.

- Section 2 is the literature review section that depicts the existing research works, current IDS models, techniques used in those models and their drawbacks. It also reveals the process of different supervised and unsupervised machine learning models, their shortcomings and the intention for the proposed new strategies.
- Section 3 consists of the research methodology part that explains the proposed modifications, techniques, architectural design, mathematical properties, theoretical explanation and hyperparameter tuning of the proposed strategies. It also contains the outer sketch of the proposed IDS models, data-processing techniques and the standard metrics used to measure the performance of our models.
- Section 4 is the results and discussion part that briefly discusses the results obtained by the proposed models and validate the performance of the proposed models with different existing models to show our models efficiency. It also discusses about the different benchmarking datasets used to evaluate our proposed designs and different hardware and software set up for the experimental design.
- Section 5 is the conclusion and future work part that shows the overall final review and findings of our research work, and further improvement that can be made on our research work.

CHAPTER 2

LITERATURE REVIEW

The machine learning algorithm is broadly classified into two types 1. Supervised machine learning 2. Unsupervised machine learning

2.1 SUPERVISED MODELS

The supervised machine learning ([Kotsiantis et al., 2007](#)) takes the labelled data and it is mainly used for the classification problem. The supervised machine learning learns training inputs with its associated class labels through a suitable function and the model aims to map a suitable class label for the test inputs as mentioned in Figure 2.1. Some of the supervised machine learning techniques include SVM, KNN, NB, MLP, & LR. The supervised machine learning process is shown in Figure 2.1.

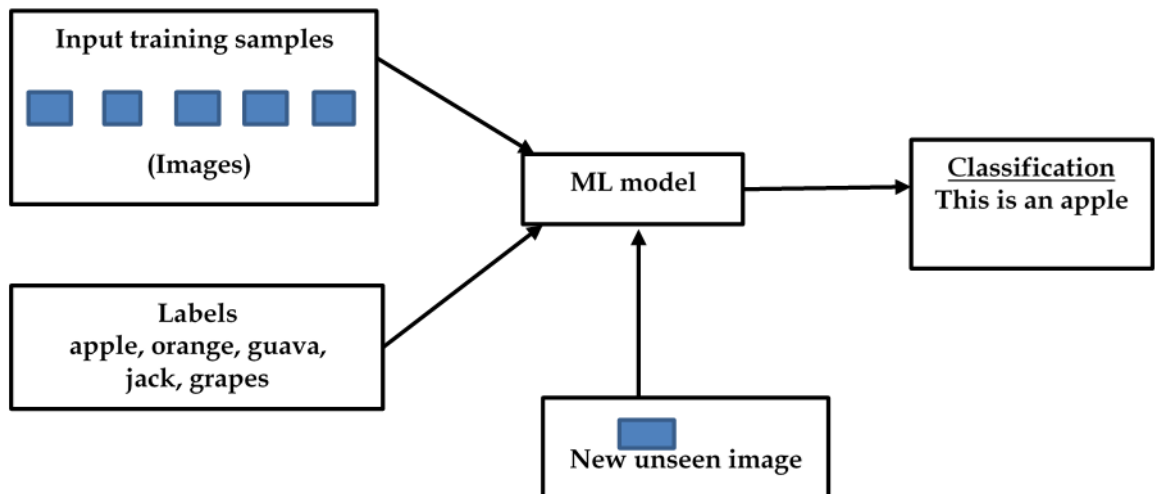


Figure 2.1: The Supervised Machine Learning Process

The KNN is an instance-based algorithm (Cover & Hart, 1967) and takes a longer time for the generalization process. It does not assume the structure and distribution of data hence it is also named as a non-parametric algorithm. The method finds the k -nearest neighbours for a data point in the test set based on the number of neighbours that are most similar in the training set. It determines which of the k -instances are most similar to a new test data point according to a predefined metric. The new data point is thus mapped to the category label of the majority voting of its neighbours. The value of k is a very important hyperparameter since it highly determines the classification performance.

The Naïve Bayes works on the principle of bayes theorem (Scott, 2004) and it is defined to be $P(Y/X) = P(X/Y) P(Y)/P(X)$ i.e. *posterior = (prior * likelihood)/ (evidence)*, where, $P(Y)$ is the probability of an outcome, $P(X)$ represents the probability of the data observed, $P(X/Y)$ is the probability of the data observed where the given outcome is Y and $P(Y/X)$ is the probability of an outcome for given observed data X . The aim is to find the probabilities of each possible outcome when an input X is given. It possesses the characteristic of conditional independence between the attributes and classifies the instances based on the concept of likelihood. After evaluating prior probability and likelihood criterion n , a test sample can be mapped to the category label that maximizes the posterior distribution.

The SVM (Cortes & Vapnik, 1995), aims to find an optimal hyper-plane that separates the classes with a maximum gap and support vectors are the data points that remain close to the hyperplane. It uses hyperplane as a boundary to differentiate samples into two distinct categories and classifies new samples into one of the categories. The classifier can give better performance on finetuning hyperparameters such as kernel function, gamma parameter etc.

The Logistic Regression is mainly suitable for binary classification. The regression strategy (Banks & Fienberg, 2003) is used to process and solve the classification problem using s-shaped sigmoidal logit function that gives the class probability values, either being 0 or 1. A linear regression-based logit function can convert any real value within the range between 0 and 1 and so the output value for the logistic regression always lies between 0 and 1.

$$\text{The equation for the logistic regression is shown as } z = \frac{1}{1+e^{-(b_0+b_1i)}}$$

Where z is the outcome, b_0 is the bias term and b_1 is the coefficient for the input i .

The DT algorithm (Quinlan, 1986) is the tree-based classifier that uses information gain as the source criteria in constructing the tree.

The decision tree comprises three essential components:

1. A decision node denotes a test on an attribute, where the DT can take a decision and functions as a decision engine.
2. A branch denotes one of the possible attribute values as the split node and it defines the outcome of the decision node.
3. A leaf node denotes the class label of the records

The DT works similar to a flow-chart like structure. It starts building the tree from the root node with highest information gain and makes a test on every feature value at each split decision node and decides the outcome of a node, i.e. leaf node. It can handle multi-dimensional data. It is human readable and interpretable since if-then rules is used to make decisions.

The DT is based on informational-theoretic principle. The DT algorithm classify the records using feature value. The weight of a feature value is determined by the information

gain criteria. The information content of a feature value pertaining to the output category is calculated in terms of entropy. The entropy is the expected reduction in impurity of a node and it is shown in equation below:

$$\sum_{i=1}^n p_i \log_2(p_i)$$

The classification of records starts from the root node and expands as the decisive nodes and terminated as the leaf node. The node in a DT classifier denotes a feature value to classify the records and the branch denotes a weight that the node has. No further splitting is required after the leaf node with larger gradient has reached for a branch. The branch can be expanded till it reaches the terminal node.

The current conventional machine learning classifiers employed in the IDS models for the attack prediction task are discussed so far. Those classifiers are highly suffered from longer training time, high computational complexity, severe overfitting and underfitting issues. Moreover, the classifiers do not give a higher classification score for the high-dimensional network traffic, since they suffer from the curse of dimensionality. These classifiers are best suited for small scale IDS models and are not the apt classifiers for the large scale high-dimensional network traffic datasets that we aim to design our security monitoring model.

We suggest using LightGBM classifier, an efficient, faster, lighter ensemble-boosting technique (Zhou, 2019) as the classifier for our proposed model. The LightGBM uses a group of decision trees to do the classification task, where the gradient of the previous DT is solved and minimized by the subsequent DT, till the residual reaches the minimum level and obtains the optimal solution of higher classification score. The process is shown in Figure 2.2.

Moreover, LightGBM has different user-customizable regularization and optimization techniques especially histogram-optimized leaf-wise strategies, that resolve the drawbacks of the aforementioned traditional classifiers.

Moreover, the conventional strategies are vulnerable to noise and corruptions in the high-dimensional network traffic datasets. In order to reduce the dimensionality of the datasets by extracting clean robust features, we need an efficient feature extraction strategy that is to be discussed in the unsupervised learning section

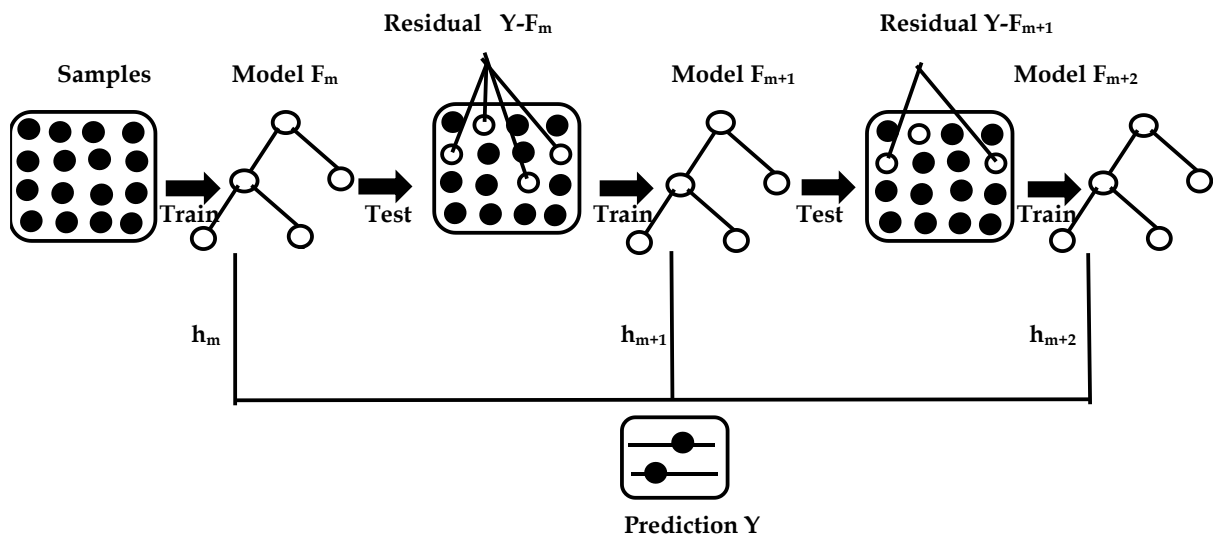


Figure 2.2: The Light Gradient Boosting Process-Flow ([LeCun et al., 1998](#))

As shown in Figure 2.2, the workflow of the LightGBM classifier is described below:

- Let F_m be the initial model.
- Let h_m be the base estimator to predict $Y=F$ by minimizing the pre-defined loss function $L(Y, F)$ and Y be the expected output function and F be the predicted output function
- Let the initial model F_m is fit with the base estimator h_m , such that $F_m + h_m = Y$ and residual $h_m = Y - F_m$ is calculated.

- Let the new model F_{m+1} be constructed by fitting the next estimator h_{m+1} to the residual gradients of its predecessor, such that $F_{m+1} = F_m(\text{residual}) + h_{m+1}$
- Let the iteration be continued till the loss function $L(Y, F)$ gets minimized and the optimal solution $Y = F$ is reached.

The LightGBM ([Chen & Guestrin, 2016](#)) is the improved and optimized gradient boosting algorithm discovered by Chen and Gastrin to accelerate the training time and reduce memory usage while achieving boosted performance as compared to other machine learning algorithms. The classifier uses a group of sequential decision trees, to classify the samples, which the classification error of the base decision tree is solved and minimized by the subsequent decision trees till the boosting classifier reaches the optimal performance score. The objective function consists of two parts,

$$Obj(\theta) = DL(\theta) + R(\theta)$$

i.e., detection loss and regularization (e.g., learning rate, tree pruning, column and row subsampling, leaf-wise strategy, histogram strategy). The various regularization parameters of the LightGBM classifier is used to minimize the model complexity namely overfitting and underfitting issues etc. The detection loss of the LightGBM model shows the predictive capacity of the model. Besides, it can handle datasets with good training speed and has high predictive power. It can automatically handle the missing values and also supports parallel processing with multithreading. LightGBM is used to handle both regression and classification problems.

Instead of scanning all the data instances to estimate the information gain of all possible decision tree split points, LightGBM only considers the data instances with large

gradients to estimate the information gain. This strategy can obtain a quite accurate estimation of the information gain with a smaller data size reducing the time and memory complexities. Besides, LightGBM also aims to reduce the number of features without significantly reducing the accuracy of split point determination. The LightGBM classifier works mainly using the two strategies as follows: The histogram optimized strategy accelerates the training speed of the model whereas the leaf-wise tree growth boosts the prediction performance of the model.

Histogram Building

- Histogram building algorithm requires $\Theta (\#data \times \#feature)$.
- The algorithm first discretizes continuous floating-point features into k discrete values (also known as bins) and construct a histogram with a width of k .
- Subsequently, the algorithm traverses the training data and compute the frequency of each discrete value in the histogram as shown in Figure 2.3. Besides, the frequency of each bin is sorted from high to low and the features with low frequency will be filtered.
- The constructed histogram of a feature compromises two types of information 1. Sum of gradients of the samples in each bin 2. The number of samples in each bin and the histogram can be divided into two portions namely 1. Left bin 2. Right bin.

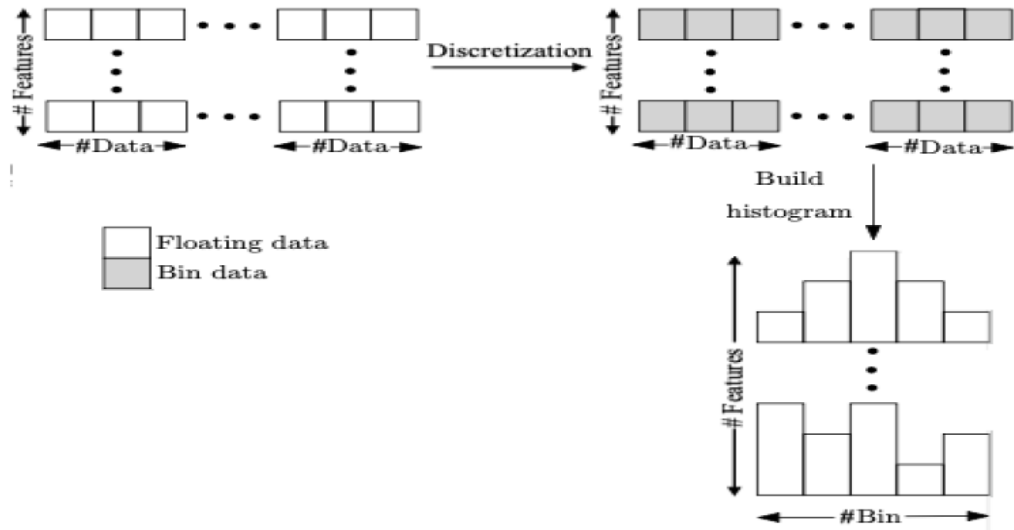


Figure 2.3: Histogram-Optimized Strategy

- The difference between average gradients and total samples of the parent bin and total gradients and average samples on one of the child bins give the sum gradients and number of samples of another sibling bin i.e. child node.
- It is computed using the formula $G_R = G_p - G_L$; $N_R = N_P - N_L$ and the loss is estimated as $\Delta_{loss} = G_L^2/N_L + G_R^2/N_R - G_P^2/N_P$

Where G_L and G_R , be the average gradients on the left and right bins, G_p denote gradient sum of parent bin. N_L and N_R denote the number of samples on the left and right bins, N_P denote the number of samples on the parent bin.

Leaf-wise Strategy

- After the histogram building, $\#bin$ is much smaller than $\# data$ since a lot of data are discarded and grouped into the same bin if the feature values are within a certain range. i.e. The complexity of the histogram strategy is only $\#bins$.
- The feature value with highest information gain is placed as the parent node.

- The information gain is measured in terms of entropy, which denotes the possible expected impurity reduction of a node.
- The information gain of a feature F with values I , where $V(F)$ is the set of possible values for a feature F , V_s is the subset of V for which the feature F has value s . If the target feature takes different values of V , then entropy set (I) and IG are denoted as follows:

$$Entropy(I) = \sum_{j=1}^s q_j \log_2 q_j \text{ where } q_j \text{ is the proportion of } I \text{ belongs to class } j$$

$$Gain(I, F) = Entropy(I) - \sum_{s \in Val(F)} V_s/V * Entropy(V_s)$$

- The information gain computation of the histogram strategy involves only *eigenvalue* * *histogram bins*, which generally speeds up the training process.
- LightGBM uses a leaf-wise tree growth strategy that searches the child node with larger information gain i.e. larger amount of data. The larger the gradient of the data, the higher the information gain.
- The gradient of a leaf node can be calculated based on the gradient difference between a parent node and another sibling node.
- The purity value of a node is the number of identical samples that reach the node. The process of tree construction stops until all the expanded branches reaches the terminal node. Figure 2.4 shows the leaf-wise tree growth strategy. The depth of a tree is capped to prevent overfitting while increasing higher efficiency.

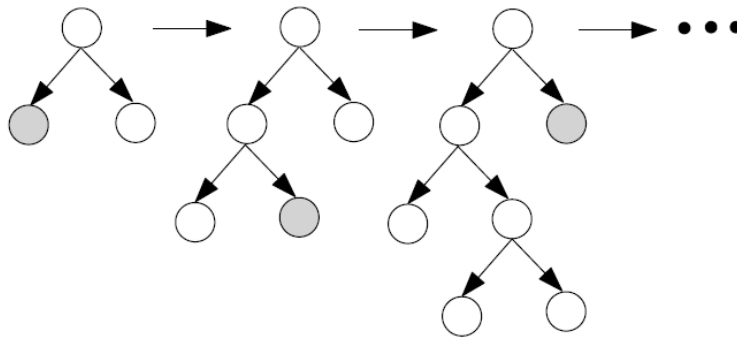


Figure 2.4: Leaf -wise Strategy

2.2 UNSUPERVISED MODELS

The unsupervised learning ([Eskin *et al.*, 2002](#); [Nguyen *et al.*, 2012](#)) is mainly used for the dimensionality reduction process. It accepts the high dimensional features and reduces them to the low-dimensional features by learning and extracting some hidden patterns from the original features as shown in Figure 2.5. Some of the unsupervised learning techniques included in our doctoral thesis namely PCA, K-Means Clustering and Autoencoder variants.

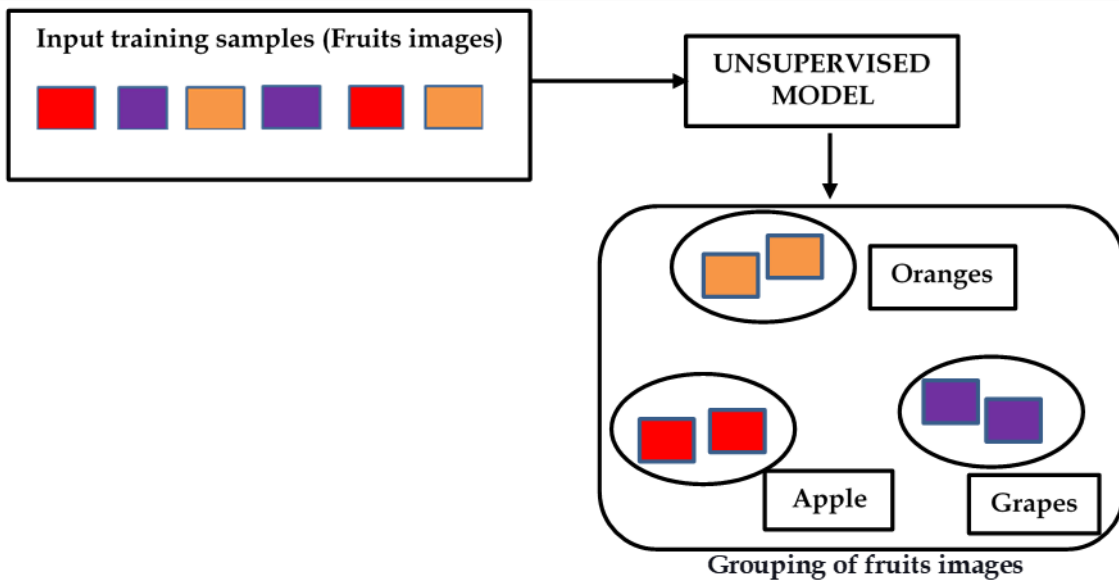


Figure 2.5: The Unsupervised Machine Learning Process

The K-Means Clustering (Jiang *et al.*, 2006) is used to find some patterns by grouping similar data points. The similarity is measured using the euclidean distance metric, where data points are grouped together to form a cluster to convey some information. It is used mainly in the outlier detection analysis. K denotes the number of clusters specified as input by any user.

PCA (Hong *et al.*, 2017) is the dimensionality reduction technique, which defines a new set of low-dimensional features from the high-dimensional features using matrix decomposition. The matrix decomposition gives eigenvalues and eigenvectors, through which it transforms the data points in each principal component in a lower-dimensional manifold. The eigenvector denotes the direction, whereas the eigenvalue denotes the magnitude of data points projection. Each principal component is perpendicular and orthogonal to the other. The first principal component is thus shown in the equation below:

$$z_1 = \varphi_1 x_1 + \varphi_2 x_2 + \varphi_3 x_3 + \dots + \varphi_n x_n$$

where φ denotes the direction and x denotes the data points, where z_1 is the first principal component with eigenvector of largest eigenvalue.

If a dataset contains M instances and N features, the matrix decomposition is denoted as $M \times N$, where the top K eigenvalues are chosen as the selected attributes. The projected new dimensional eigenvector contains the characteristic vector-matrix set $N \times K$. The synchronization among them is denoted as

$$Final\ Data\ (M \times K) = O\ (M \times N) * E\ (N \times K)$$

Where O is the original data, E is the eigenvector, M is the number of samples, N is the number of features, and K is the selected top eigen values.

It is very much efficient and applicable for the linear data transformations, where the network security datasets used for our evaluation are non-linear types, for which autoencoders with activation functions that support non-linear data structure can be used.

Autoencoder ([Hinton & Salakhutdinov, 2000](#); Coates *et al.*, 2011) is a type of artificial neural network (ANN), an unsupervised learning technique, widely used as the dimensionality reduction technique. The autoencoder produces a replica of the input in the output layer, (thereby the number of nodes in the input and output layers are similar) by learning and extracting patterns in the hidden layer, which conveys some information about the high-dimensional input data in compressed form, thereby the number of nodes in the hidden layer are lesser in number than the input and output layers. Unlike ordinary ANN, a supervised learning technique, which is used as a classifier, can predict the class label for the input samples, thereby the nodes in the output layer are same as the number of outcome labels in the input dataset. The autoencoder, a feature extraction strategy can reduce the dimensionality of the datasets, by learning and extracting some useful patterns from the original input dataset (transform high-dimensional representation to low-dimensional representation). The process is shown in Figure 2.6.



Figure 2.6: The Block-Diagram of an Autoencoder

Before proceeding to the autoencoder, let us discuss the ANN, which is the base for an autoencoder model. Artificial neural networks (Hornik *et al.*, 1989; [Nilsson, 1998](#)) are universal function approximators, which learn a mapping function from input x to outcome y . It is based on the interconnected system of neurons that can work and process artificially like

a human brain. The neurons have the capability of learning and extracting patterns and structure from the high-dimensional input dataset and it is the main computational processing unit in ANN. The architecture of an ANN is shown in Figure 2.7. The operations, structure and components are explained below

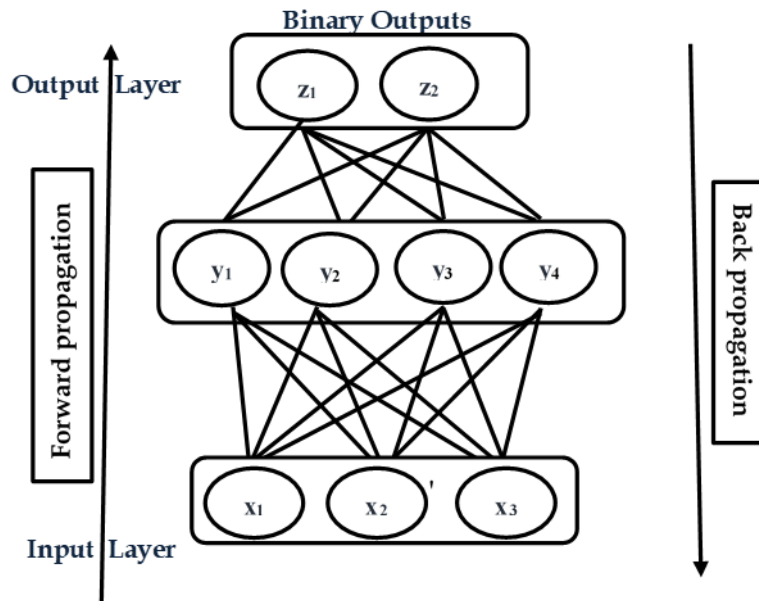


Figure 2.7: The Architecture of an ANN

The ANN consists of an input layer to accept input values, a hidden layer for learning patterns and an output layer to produce the outcome values. The layers and their neurons are all interconnected. The network is trained (Lee *et al.*, 2001) to learn suitable weights for each input values and so the estimated output would not vary much than the expected output. To measure the difference between those two outputs and evaluate the performance of a neural network, the cost function namely cross-entropy is used. The network is trained several times, i.e. forward and backpropagated to fine-tune the weights of the network and maximize the objective function. The stochastic gradient descent (SGD) statistics is used in the back-propagation procedure and a suitable optimiser is fixed within it to optimize the back-propagation process. The three layers of an ANN are:

Input layer: An input layer neuron receives input signal values $X (x_1, x_2, x_3)$. The vector of weights $W (w_1, w_2, w_3)$ in the weight matrix is randomly assigned to each input values $X (x_1, x_2, x_3)$ in the input matrix.

Let $X \in R$ be the input manifold values,

Let $Y \in R$ be the processing representation,

Let $Z \in R$ be the output function values

Processing layer: Otherwise called as hidden layer. The processing layer receives the matrix multiplication of the weighted sum of inputs i.e. inputs and their corresponding weights from the input layer and process it through the activation function to transform the signals to their encoded form and learn the required hidden representations from it. The formula used for calculating the output of a neuron is shown below:

$$z = \sigma \sum_{i=1}^n x_i w_i$$

where z is the output value, x_i is the input value and w_i is the associated weight value, σ is the non-linear activation function.

The equation shows that the summation of the input values and their corresponding weights give the weighted sum of inputs. Then the activation function is applied over the weighted sum to transform the non-linear signals into the required output signals. The ANN accepts the input, process it and produces the output by a summation- mapping function. Figure 2.8 denotes the flow to calculate the output of a neuron.

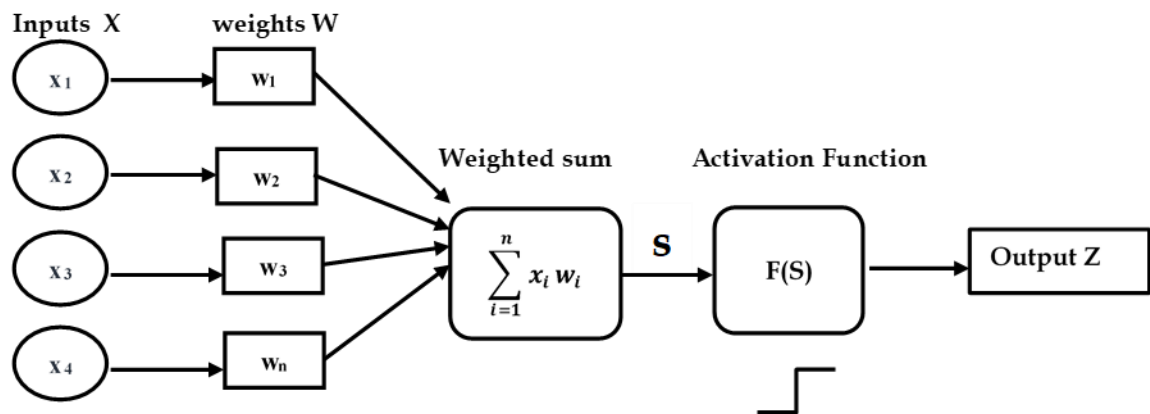


Figure 2.8: The Output Computation of a Neuron ([Hornik et al., 1989](#))

Output layer: It gives the processed information as an outcome.

The ANN is a fully connected dense layer network. The number of nodes in the input, hidden and output layers are determined according to the number of input features and the number of output values produced. Usually, there are a larger number of neurons in hidden layer than the input and output layers to process the input values. The binary classification problem requires two output nodes whereas the multiclassification problem requires multi nodes. The ANN randomly assigns the weights in the weight matrix to the inputs in the input matrix.

It contains two phases: 1. Feed-Forward Phase 2. Back-Propagate Phase

Feed-Forward Phase: The FFP of the ANN forward propagated the input data values to produce the output by the summation and the activation function in the hidden layer, through which it maps the input to output. The feed-forward phase computes the activations in the network and gives output values. The cost function measures the difference between real output and predicted output by the network.

The forward propagation of a neural network is shown in the equation below as follows: $Z = C(Y) = C(F(XW))$, where C is the cost function, F is the activation function, W is the weight of the network, and X is the input data.

The partial derivatives are taken to calculate the gradient of the cost function. The gradient of the cost function is computed for each weight of the network and is shown in the equation as follows:

$$\partial c / \partial w_j = \frac{\partial c}{\partial \bar{z}} \frac{\partial \bar{z}}{\partial \sigma} \frac{\partial \sigma}{\partial w_j}$$

We will find the derivative of the cost function in a closed-loop form in order to minimize the cost function C . i.e. $\min(C)$ where $C(W, X, Z)$. W is the weight matrix, X and Z are the input and output data. C is the cost function named cross-entropy to measure the difference between the predicted output and the actual output.

The cost function (Golik *et al.*, 2013) is used to measure the loss and evaluate the network performance. If z denotes the expected output and z' is the predicted output, then the difference between the expected and predicted outputs are measured by the cross-entropy function. For a binary classification problem, the cross-entropy is denoted as binary cross-entropy, where it uses the logistic sigmoid activation to estimate the cost function. The cost function is given by the equation, $C(z, z') = \sum_{j=1}^n [z_j \log z'_j + (1 - z_j) \log (1 - z'_j)]$.

For a multi-class classification problem, the cross-entropy is denoted as multi cross entropy, where it uses the SoftMax activation to compute the cost function. It is given by the equation, $C(z, z') = \sum_{j=1}^n z_j \log(z'_j)$, where z_j and z'_j are the j -th feature of z and z' respectively.

The cost function is continuously differentiable and the network uses chain rule to calculate the partial derivatives of the cost function, where it measures the difference between the desired and obtained outputs. A gradient is a partial derivative (i.e.) that rate of change of cost function. z_j is the probability of outcome for classifying records in one class, whereas z'_j is the probability of outcome for classifying records in another-class.

The activation functions (Nilsson, 1998) in the ANN support the transformation of non-linear input signals to the required encoded signals. The different activation functions used in the neural networks are given below:

The sigmoid is a non-linear logistic activation function, which is bounded by discrete values 0 and 1 and it is fully differentiable. The sigmoid function is denoted as in the equation

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

The RELU is a piece-wise non-linear activation function (Maas *et al.*,2013) that is neither differentiable nor bounded. It is mainly used as the encoder activation function, for transforming the input data signals to the latent-encoder signals. The RELU transforms the input in the range of $(0,1)$ to compute the probability of an output function and its derivative gradient can take only two values 0 and 1 . It squashes the negative input values to zero values to process it easily. The RELU makes the network converges faster and is not saturated at 1 , 0 or -1 values. It is very much resistant to vanishing gradient problem. The equation of the RELU function is $RELU(z) = \max(0, x)$

The SoftMax probabilistic function is used in the output layer for the multiclassification problem as like multinomial logistic regression, which is bounded by multiple target values. The function is given by $SoftMax(z) = e^{x_i} / \sum_{j=1}^n e^{z_j}$, where n is the number of class labels in the multi-class problem, e^{x_i} is the standard exponential value for an input function, e^{z_j} is the standard exponential value for an output function.

Back-Propagate Phase: The backward phase (Rumelhart *et al.*, 1986) is used to fine-tune the weights in a neural to minimize the cost function so that the optimal output has been achieved. After the gradient is computed in the feed-forward phase, the weights are finetuned in a small rate known as learning rate, denoted by the parameter β . It determines the steep size of the updates of weights. By the computed gradient $\frac{\partial(cost)}{\partial w_j}$, learning step β and the previous value of the weight w_j , the new value of the weight can be computed as shown in the formulae as below

$$w_{j+1} = w_j - \beta \frac{\partial(cost)}{\partial w_j}$$

The phase is carried out using the stochastic gradient descent algorithm and it is optimized by different optimizers. The SGD computes the gradients of an output function WRT the input function, i.e. rate of change in cost function and is minimized in the direction of gradients i.e. steep down from top to bottom. For larger dimensional security datasets, the input samples are grouped into batches, such that each batch contain a group of samples to speed up the computation process, since the gradient computation for every single sample is a tedious and time-consuming task

In back-propagation, a learning procedure called SGD (LeCun *et al.*, 1998) is employed to minimize the cost function, where the gradients of the cost function are computed and taken as a convex optimization problem. The structure of the convex optimization is taken in the shape of an inverted bowl, where the initial gradient computed is taken as the slope of the tangent as a starting point and the weights are finetuned WRT gradients and minimize the gradient at each learning step, till the gradients move downwards by the tangent of the slope and reach the bottom of the inverted bowl. The bottom of the inverted bowl denotes the zero-minimum level. We can use the derivatives in a feedback loop to update parameter vector w by finetuning it in the direction of the gradient. Thus, the gradient reaches zero level and converges to a minimal cost function. The slope of the tangent line determined by the gradient of the cost is calculated at each step till it reaches the bottom. The important criteria here is the learning rate since it determines the step size of the learning criteria i.e. weight updates in the direction of convex optimization, to move downwards the slope to minimize the cost function.

E.g. the gradient of the convex optimization function $y=5x^2$ to be $10x$. The gradient then becomes the tangent of an initial point on the inverted bowl. On the inverted bowl, the slope at any given point is a line tangential to that point. The SGD computes the derivative of the cost function to determine the gradient. The gradient determines the direction of the slope and thus the gradient moves downwards WRT weight updates at each iteration and reaches the bottom of the inverted bowl. The convex-optimization cost function is visually represented as shown in Figure 2.9.

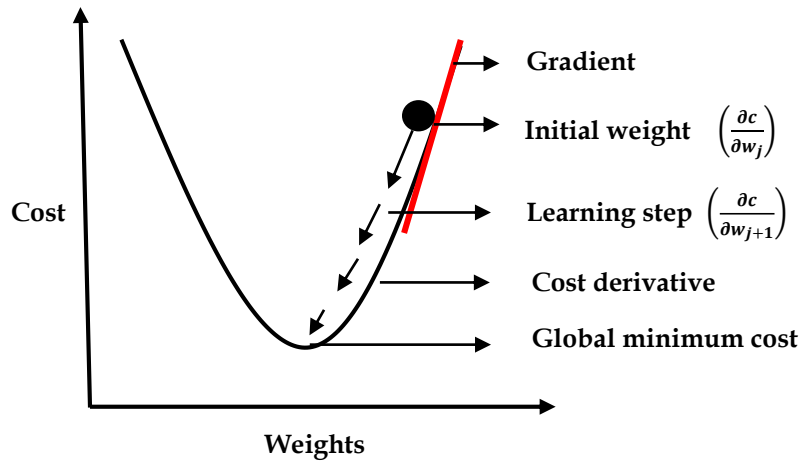


Figure 2.9: The Convex Optimization Cost Function ([Rumelhart et al., 1986](#))

The process is repeated for every single record of a dataset, whereas instead of updating the weights for every single record in an epoch which is a tedious and time-consuming task, group the set of samples into different batches, and the weights are updated for an entire batch in a single epoch to make the *SGD* back-propagation training faster. An epoch is a single optimization pass that scans the entire training set in a single round. More rounds of optimization passes are applied during *SGD* training to minimize the reconstruction error. The parameter (w) is updated for every epoch to minimize the gradients.

The optimizer (Reddi et al., 2018) is the function used in the *SGD* statistics that can optimise the back-propagation learning process to reduce the cost function and achieve the optimal solution. It also makes the back-propagation procedure faster. The Nester-Accelerated Adaptive Moment Estimation optimiser is used for the back-propagation procedure.

NADAM (Dozat, 2016) is the optimizer used in the back-propagation procedure to optimize and speed up the procedure. The NADAM optimizer is used to compute the gradients in a very faster manner to minimize the cost function, since it blends both properties of NAG

and ADAM optimizers. The learning process is accelerated by summing the exponential decay average of the previous and current gradients and by using this, the moments of the gradients are being estimated in the direction of the calculated gradients and is shown in equation as follows:

$$x(t) = x(t-1) - \beta / (\sqrt{n_{hat} + eps}) * n_{hat}$$

where β is the learning step, $\sqrt{\cdot}$ is the square root function, eps is the parameter added to avoid the divide by zero error, $x(t-1)$ is the past gradient at a time $t-1$, $x(t)$ is the current gradient to be calculated at a time t .

The concepts of ANN have been discussed detailly so far. Now let us discuss the concept of an autoencoder. The AE generally consists of two phases and the two phases of an autoencoder are 1. Encoder 2. Decoder.

The encoder transforms the high-dimensional representation to the low-dimensional representation by extracting the hidden patterns, that represents the structure of the input dataset. It is the most important part since the features encoded are used for the prediction task.

The encoder phase is shown in the equation below:

$$y = F(x)$$

Where x is the input representation, F is the encoder-activation function, y is the encoded representation.

The decoder transforms the encoded representation to the output value, which is similar to the original input representation. The decoder phase is shown in the equation below:

$$z = G(y)$$

Where z is the output representation, G is the decoder-activation function, y is the encoded representation.

The decoder reconstructs the input data values using the encoded representation by minimizing the reconstruction loss. i.e. difference between the reconstructed input and the actual input, which is measured in terms of cross-entropy. The reconstruction loss can be minimized using back-propagation optimization such as SGD by finetuning and updating the weights in a network. The weights are shared between encoder and decoder i.e. $w = \hat{w}$

$$C(w, \hat{w}) = \min C(x, z)$$

Where x is the actual input and z is the reconstructed input.

For each mini-batch, the average gradients computed are back-propagated and the weights are finetuned till the gradient reaches the bottom minimum level and obtain the optimal solution. The autoencoder has identical nodes in both input and output layers, to reproduce the input data in the output layer by minimizing the reconstruction cost. The architecture of an autoencoder is shown in Figure 2.10.

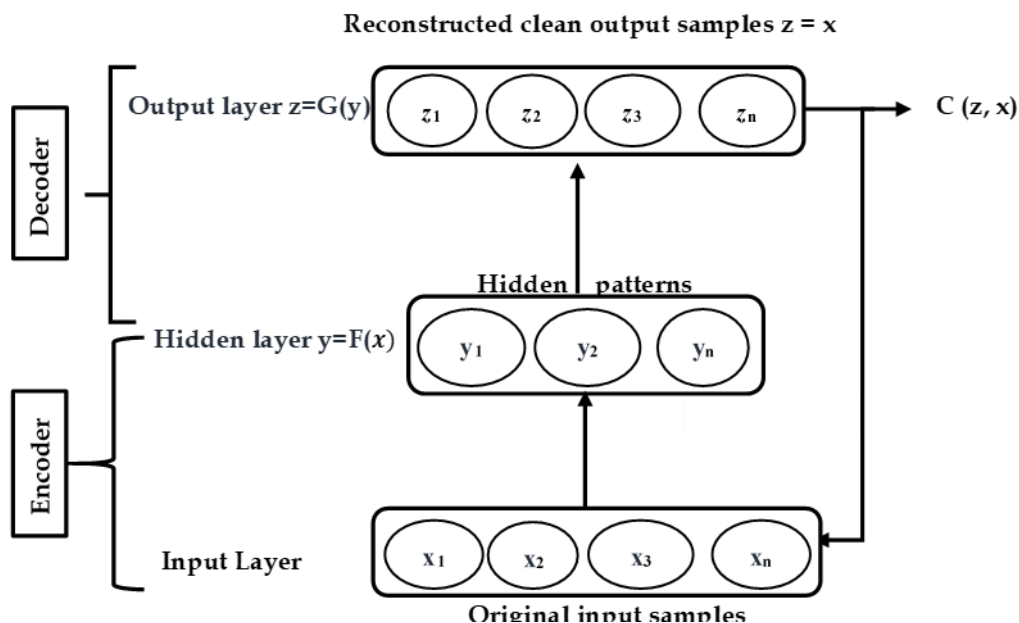


Figure 2.10: Architecture of an Autoencoder

The autoencoder consists of two types of parameters (Bergstra *et al.*, 2011) 1. Trainable parameter 2. Non-trainable parameter.

The trainable parameters are automatically learned by an autoencoder during the training phase that includes weights, bias etc. The non-trainable parameters are hyperparameters (Bergstra & Bengio, 2012) that can be finetuned by several investigations and fixed to get the reliable performance results. The hyperparameters include hidden layers, neurons, learning rate, optimizers, back-propagation algorithm etc.

Denoising autoencoder ([Vincent *et al.*, 2010](#)) is a type of an improved variant of a traditional AE which is robust to the noise and corruptions in the input datasets. It artificially corrupts a portion of the original input dataset by adding isotropic gaussian noise with zero mean and unit standard deviation. The added noise acts as the partial regularization function to learn the robust compressed features representation in the hidden layer and reconstructs the original input by removing the noise and corruptions. The structure of the denoising autoencoder is shown in Figure 2.11 with the following layers.

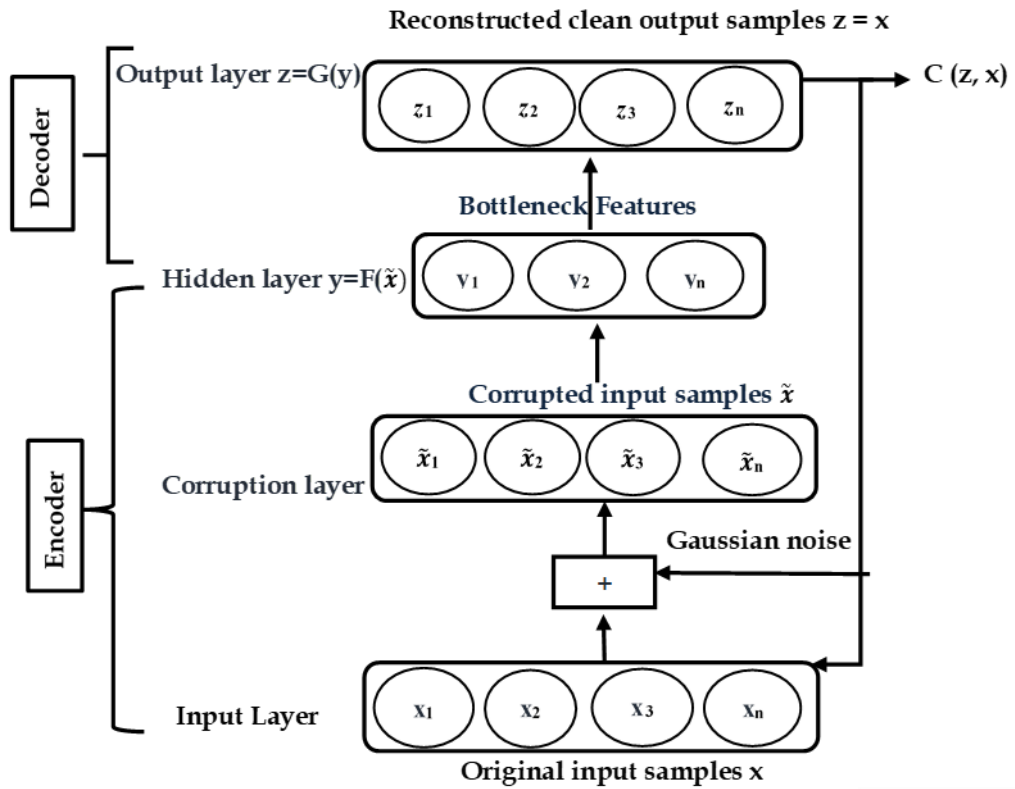


Figure 2.11: Architecture of a Denoising Autoencoder ([Vincent et al.,2010](#))

Input layer: Consists of initial input x .

Corruption layer: Corrupts the initial input x into partially corrupted input \tilde{x} by adding the gaussian noise as follows. $\tilde{x} = x + N(\mu, \sigma^2 I)$

where $N(\mu, \sigma^2 I) = N(0, 1)$ be the isotropic gaussian noise, μ is the mean, σ is the standard deviation and I is the identity matrix. The ratio of noise added to the initial input is within the range of 0.1 to 0.2 .

Encoder layer: The encoder maps the partially corrupted input features \tilde{x} to a hidden layer features y in a low dimensional space using the encoding function listed as follow

$$y = F(w\tilde{x})$$

where F is the encoder nonlinear activation function, w is the weight of the encoder.

Decoder layer: From the hidden layer features y , the decoder reconstructs the output z which is as close as possible to the uncorrupted input x , by minimizing the reconstruction loss over a training set. The decoding function is shown as follows

$$z = G(w'y)$$

where G is the decoder activation function, w' is the weight of decoder. The reconstruction loss between x and z is measured by using the cross-entropy loss function.

The reconstruction loss can be minimized by finetuning the parameters of the denoising autoencoder using the backpropagation algorithm. Nadam is the chosen optimizer used in the backpropagation algorithm. Notice that the hidden features generated from the encoder are used for classification by the LightGBM classifier. Lastly, the binary and categorical cross-entropy loss functions are used for binary and multi-classification purposes.

The different unsupervised learning strategies such as k-means clustering, pca, autoencoder have been studied so far. The k-means clustering is not an efficient dimensionality reduction technique. The pca is mainly suited for the linear data types, whereas the network traffic is not always linear. The autoencoder is a widely used non-linear dimensionality reduction technique but is highly vulnerable to noise and corruptions in the network traffic. Thus, we go for the DAE, a suitable dimensionality reduction strategy for our attack detection system. The DAE with the gaussian noise in the input layer can remove the distortions and compress the dimensionality of the dataset by learning and generating some patterns and structures from the high-dimensional network traffic. Reducing both distortions and dimensionality, are achieved in a single strategy. Thus, DAE functions as 2 in 1 technique.

So far, different supervised and unsupervised machine learning techniques used in designing various attack detection models are discussed. In this sub-section, literature summary of the current IDS models, techniques and datasets used in designing those models, performance and drawbacks of the current techniques are analyzed briefly. This survey also gives us knowledge and the need to propose new IDS models that are to be built by the novel and innovative strategies, which can predict any intruders with maximum prediction score.

2.3 IDS with Classifiers and without Features Selection

The different IDS models existed with only supervised learning classifiers without any feature selectors are discussed below in the following sub-section:

The DNN with fully connected layers is used for the IDS model (Vinayakumar *et al.*, 2019) over the standard benchmarking datasets such as NSL-KDD, UNSW-NB, Kyoto and CIC-IDS respectively. The network hyperparameters are tuned in a stochastic manner and the SoftMax activation function is used for the multiclass classification task and the model achieved a detection score of 92.7%, 78.5%, 65.1%, 95.6% for the datasets, NSL-KDD, UNSW-NB, Kyoto, and CIC-IDS respectively. It is to be noted that RELU is the better activation function to transform the non-linear signals in neural networks.

The security monitoring tool is proposed and designed (Ahmar *et al.*, 2018) by three different machine learning classifiers namely SVM, RF, MLP and their performance is measured and compared in terms of accuracy, precision, recall and f1-score. The SVM is employed with the RBF kernel. The MLP is with a single processing layer and the random forest is an ensemble of decision trees. The NSL-KDD dataset with 80% training set and 20%

testing set is employed to estimate the proposed model. The MLP outperformed the other classifiers. The MLP with a single processing layer has good processing capability to give a better prediction score.

To protect the network traffic from different attacks, intrusion detection system is developed and evaluated using the recent modern CIDDS-002 dataset. The conventional classifiers of DT and naïve bayes are trained separately (Razdan, 2021) by the dataset features to classify the samples. The IDS built on DT outperforms the naïve bayes classifier by obtaining 99% accuracy.

Tama and Rhee (2019) proposed an IDS by gradient boosting machine strategy against the three datasets NSL-KDD, UNSW-NB15, and GPRS. The GBM is an ensemble learning methodology, consists of a sequence of decision trees to solve the classification problem. The researchers evaluated and compared the performance of GBM with other different classifiers such as random forest (RF), deep neural network (DNN), support vector machine (SVM), decision trees (DT) in terms of standard performance metrics. Finally, they concluded the research by confirming that GBM with k-fold cross validation outperforms all other classifiers in all three datasets respectively. The hyper-parameters tuning of the classifiers are done using the grid search to find suitable hyper-parameters.

To ensure information security in the network, recurrent neural networks are involved (Yin *et al.*, 2017) to ensure information security in the network using NSL-KDD. The model is evaluated for both bi-class and multi-class problems and also compared with various traditional machine learning algorithms such as DT, ANN, RF, SVM. It is observed the RNN

is not much superior in performance when compared with other traditional methods. The RNN is most apt network for the sequential and temporal data types. However, the network traffic taken for our evaluation are all structured data types and RNN is not suitable for our model design.

The port scan behavior is predicted on the network traffic of the datasets namely NSL-KDD and UNSW-NB using the multilayer neural networks (Nguyen *et al.*, 2018), which consists of two stages 1. pre-training stage 2. refining stage. The pre-training stage is proceeded by the greedy-forward propagation algorithm, that forwards the activation from input to the output layer. The refining stage is carried out by the back-propagation strategy, which backpropagates the gradient of the cost function, to finetune the weights of the network and minimize the cost function.

The intruder monitoring tool is designed using the convolutional neural network (CNN), over the NSL-KDD (Wu *et al.*, 2018), where the data samples are converted to image format, before feeding to CNN, since it mainly supports only image format. As the dataset is highly imbalanced, cost function weight coefficient is proposed to fine-tune the weights of the cost function and the model achieved a detection score of 79%. It is clear from the results, that the CNN network is the most suitable for the image data types due to its convolution structure. The cyber security datasets we use to evaluate our proposed designs belong to CSV data formats. The CNN is not a suitable network architecture for the CSV datatypes.

The BoT-IoT dataset is used to design an IDS to protect the IoT network environment. The proposed model (Ferrag *et al.*,2020) consists of two stages: stage 1 classifies the IoT

network traffic as normal or attack by the decision tree classifier, stage 2 categorize the type of attack (multiclassification) using the random forest classifier. The two-stage tree-based classifiers are blended to build an IDS model and obtain a higher performance level.

Ahmim *et al.* (2019) describes the stacking approach consisting of three tree-based classifiers namely REP Tree, JRip algorithm and Forest PA to construct the intrusion monitoring system and evaluate the model against the CIC-IDS. The first two classifiers performed the binary classification task i.e. classifies the samples as normal or benign. The outcome of those are used by the third classifier for the multiclassification task i.e. predicts the category of attacks.

The classifiers without any feature selection technique scan and use every single value of all features including any insignificant feature values to classify the samples, which affect the classification score and increases the computational complexity, training time, memory resources etc. The feature selection is an important part in designing an IDS model for the attack prediction task, since the techniques used, select attribute values that can contribute some extent to obtain better prediction score and remove an attribute value that is adversary for any ML classifier performance. The literature survey also shows that instead of a single classifier, an ensemble technique with a group of classifiers i.e. more than a single classifier gives a better prediction score, but consumes a longer time to train the model.

2.4 IDS with Classifiers and Features Selection

The different intrusion detection models that are currently available contain two parts 1. feature selectors 2. classifiers. Feature selector which selects an optimal subset of features, through which any classifier can classify the samples with reliable score in short training time.

An IDS model is designed by Aksu *et al.* (2018) over the CIC-IDS dataset to predict DDoS attacks, where it can flood the network traffic by continuous messages from the intruders. The dataset is divided into 90% training data to train the model and 10% testing set to evaluate the model. The feature selection algorithm-fisher score, a filter method, that can rank the attributes based on the fisher's score in descending order and the attributes with least scores are eliminated. The filter methods, based on the statistical and numerical properties of a feature, are faster and less computationally expensive to eliminate the useless features of the dataset and the proposed model is built over three machine learning classifiers such as KNN, SVM and DT. In the KNN algorithm, the k-value is set as 4. The linear kernel is used in the SVM classifier and Gini's diversity index is used as the split criterion in the DT algorithm. Among them, the KNN classifier with 30 features performed best in terms of standard quality performance metrics for the bi-classification task. The number of nearest neighbours play a major role in predicting DDoS attacks.

The researchers (Ambusaidi *et al.*,2014) filtered the irrelevant features in the datasets such as KDD Cup99, NSL-KDD and Kyoto 2006+, respectively using the flexible mutual information method. The filter-based feature selection algorithm selects the predominant subset of features using the mutual information share i.e. cross-entropy among the features and it deals with the relevance of a feature for the class. After the feature selection process is over,

the selected attributes from the corresponding datasets are used for the classification process by the least square-support vector machine (LS-SVM) and the model achieves better performance in terms of accuracy for all the three corresponding datasets.

To secure the IoT-interconnected devices, a security monitoring tool is designed (Alsamiri & Alsubhi, 2019) against the BoT-IoT dataset that contains modern and realistic IoT network traffic. The importance of the traffic features is measured by the random forest regressor and the most weighted subset of features are evaluated by the random forest classifier to classify the samples.

The tor-traffic conceals the identity of network users and provides private communication among the network users, where the network traffic is misused by hackers and online criminals for their illegal transactions and fraudulent activities. To classify the tor traffic from the non-tor traffic, the ISCX-TOR dataset is used, where it contains imbalanced samples in the minor output category and it is balanced by the resampling technique. The importance of the features is measured using the SHAP values, which computes the feature weight by the average marginal contribution of each variable. The selected weighted features are being used by the different machine learning classifiers namely logistic regression (LR), support vector machine (SVM), and neural networks (NN) to perform the classification task (Chowdhury *et al.*, 2019). The neural network classifier overtakes the other two classifiers. In addition to this (Lashkari *et al.*, 2017), the same dataset is evaluated by the correlation coefficient feature selection algorithm with random forest classifier and the proposed model obtained an DR of 86.92% in 12.5s training time.

The IoT devices in the cloud networks are secured by the attack detection tool, developed by Mahajan *et al.* (2020) the decision tree (DT) classifier. The model is built over by the CIDD5-001 dataset. Firstly, the optimal subset of features is chosen by the entropy-statistical measure and the selected features are used by the DT classifier to discriminate the inputs.

The attack prediction system is built using five different classifiers namely LR, SVM, RF, GBM, and NB against the datasets such as NSK-KDD and KDD99 respectively, which are very old datasets. The authors (Gupta & Kulariya, 2016) evaluated the performance of those classifiers by the standard performance metrics and concluded that LR performed better in 289.105s training time using the features selected by the correlation feature selection (CFS) strategy, whereas the RF obtained a higher prediction score using the features selected by the chi-square feature selection strategy. The CFS removes the highly correlated features. The chi-square test deals with the hypothesis testing using pearson's chi-squared tests.

An IDS model (Amangele *et al.*, 2019) explores the use of machine learning algorithms against CIC-IDS for attack traffic detection in an Internet of Things (IoT) connected through a software-defined network (SDN). The seven categories of attacks are taken for the prediction. The feature selection is done using GINI importance, which calculates the variable weight as average over the number of splits, a tree-based strategy and the model is built using the classifiers namely LR, KNN, DT, NB, SVM. The LR is a simple logistic s-shaped activation function, mainly used for the regression task and bi-class classification problem. The DT classifier is based on if-then rules. The NB is the conditional probability that assumes conditional independence among the attributes and is based on its probability of success. The

SVM classifier performance is determined by the hyperplane constructed between the classes and the support vectors nearer to those hyperplanes. The performance among the classifiers is measured and compared. It is noted that the CART tree led in the race by giving better performance in lesser training time, but it is often stuck in overfitting like neural networks.

The internet of things is getting popular nowadays since it connects the world through digital devices and so the threats against IoT prevail and grow in numbers. To secure the IoT network against the vulnerabilities launched by the attackers in its application protocols (DNS, HTTP etc.), an intruder prediction model (Moustafa *et.al.*, 2018) is evaluated on the datasets namely UNSW-NB15 and NIMS botnet, using the ensemble method consisting of three conventional ML classifiers namely DT, NB and ANN respectively. Previously, split is involved in the feature reduction process. The proposed model achieved higher performance.

The weighted vote scheme by combining the classifiers such as KNN, DT, MLP, SVM is used in building the attack monitoring model (Sornsuwit & Jaiyen, 2019) and the correlation feature selection removed the uncorrelated features of the datasets namely UNSW-NB and NSL-KDD. Finally, the remaining selected optimal subset of features are fed to the weighted vote scheme technique to classify the samples.

Lopez *et al.* (2019) evaluated the CIC-IDS dataset for their IDS model design to predict the DDoS attack. The feature selection techniques of recursive feature elimination (RFE), and correlation feature selection (CFS) are used to select the optimal set of features. The RFE is a wrapper technique, that selects the features based on a classifier performance score. The CFS, a filter method, analyses the weight of a feature based on the correlation score.

Since the dataset is imbalanced, i.e. contains an unequal number of samples in the minority classes, stratified shuffle split technique, a cross-validation strategy is used to resolve the class imbalance problem. The samples of the dataset are classified using the classifiers, RF, DT, MLP, SNN, KNN and DNN and the performance is analyzed in terms of standard quality metrics. The model obtained higher detection rate and precision by RF classifier using its subsampling technique i.e. random selection of proportion of records at each iteration. The neural network took a long time to train the model.

The IDS is built by Mamun *et al.* (2016) to detect harmful malicious URLs on the web, through the ISCX-URL dataset, since the URLs are the main platform for online illegal activities. The model consists of two phases 1. feature Selection 2. URL detection. The feature selection process is performed by the IG and ranker methods. The IG selects the features in according to the entropy-informative measure and the ranker method ranks the attributes corresponding to the predictive capability of each feature. Using the selected subset of features, the URL samples are classified by the RF classifier and obtained higher precision and recall.

The two benchmarking datasets such as NSL-KDD and CIC-IDS are employed for the IDS evaluation. The authors, Bansal and Kaur (2019) performed the features selection using the ensemble strategy consisting of three filter based-feature selectors namely symmetric uncertainty (SU), chi-squared and relief. The symmetrical uncertainty computes the interaction gain of a feature measure. The chi-squared is a statistical hypothesis test, that measures the relevance of a feature. Relief is a statistical measure that estimates the weight of a feature based on the variance measure. The attack classification is carried out using the different ensemble classifiers like XGBoost, Ctree and stand-alone classifiers like SVM, neural net. The XGBoost

leads in the race by achieving an DR of 98.85% against the NSL-KDD and 98.42% DR against the CIC-IDS.

A new dataset named IoTID is developed and used by Ullah *et al.* (2020) to detect the modern category of intruders in an IoT network environment. The correlated features are removed first using CFS technique and the novel feature selection strategy named shapira-wilk algorithm, a test of normality in frequentist statistics is proposed to rank the features, where the ranking of the features is computed according to the regularity distribution of occurrences WRT the features and the highly ranked features are used by the various supervised classifiers namely SVM, LR, NB, DT, RF to classify the IoT traffic samples. The tree-based algorithm outperforms the rest of the classifiers.

The hybrid feature selection techniques such as correlation feature selection and bat algorithm, a bio-inspired algorithm, that functions based on the echolocation of bats, are blended by Zhou *et al.* (2019) to select the subset of features from the CIC-IDS. The CFS technique filters the uncorrelated features and the bat algorithm uses the frequency tuning and form the fitness function to select the subset of features and finally, 13 features are selected. The ensemble technique, consisting of classifiers namely DT, RF, and Forest by Penalizing Attributes are used to do the attack prediction task and the voting technique that combines the probability distribution of the aforementioned classifiers are finally employed in classifying the instances. The assembling of decision tree-based classifiers gives higher performance.

From the literature survey, we come over different feature selection techniques. Each feature selectors (Dash *et al.*, 1997) have its own mathematical, algebraic, and statistical

properties, that estimate the weight of a feature. The feature reduction methods may remove some features that are insignificant and irrelevant to their properties. The number of features in a reduced features subset are always lesser than the number of features in an original feature set. From the survey, we come to understand that the feature selection techniques improve the classification score and accelerates training speed of the model.

Though feature selectors remove some of the features in a dataset, there are cases that every single feature value may contribute some extent to classify the datasets samples in the IDS model. Moreover, the high-dimensionality of the cyber-security datasets is highly challenging for the ML classifiers to train and classify the samples. In such cases, the feature selection techniques do not contribute much successful results in reducing the dimensionality of the dataset by learning and extracting any useful patterns and structure from the high-dimensional datasets. Thus, we move our survey studies on different feature extraction strategies that can reduce the dimensionality of the dataset by learning and extracting the hidden patterns. The feature extraction techniques use all features in an original feature set and combine the features using its strategy, and transform the data points into the lower-dimensional manifold. The extracted features are lesser in number and are different from the original feature set, but includes all feature values in an encoded form of representation. The survey shows some of the feature extraction strategies used for the dimensionality reduction of the IDS models.

The dimensionality reduction techniques do not remove any subset of features from the original feature set. Instead, it reduces the high- dimensionality of the original features by

combining the feature values in an original feature set and projecting the values in another form of encoded representation in the lower-dimensional manifold.

2.5 IDS with Classifiers and Features Extraction

The intrusion detection models developed by combining different feature extraction strategies with machine learning classifiers are discussed in this sub-section.

The high-dimensional CIDDs-001 cyber security dataset is compressed by the PCA technique, an unsupervised dimensionality reduction strategy and SMOTE strategy is involved (Cuautla *et al.*, 2020) to increase the minority class samples of the dataset. The latent manifold is used by the KNN to classify the records. The hyperparameters are tuned by the grid-search optimization technique and obtain good performance scores such as an accuracy of 98.72%, precision of 98.17%, DR of 98.15%.

In another work by Yulianto *et al.* (2019), PCA is involved in the reduction of high-dimensional features of the CIC-IDS, where the imbalanced class distribution of the dataset is handled by the SMOTE to increase the samples of the minority classes. The ensemble-boosting technique is further employed to classify the samples and obtain better performance.

On continuation of evaluation on the CIC-IDS (Abdulhameed *et.al.*, 2019), the high-dimensionality of the dataset is reduced by the principal component analysis (PCA), and the lower-dimensional latent manifold extracted are used by the classifiers namely RF and NB respectively. The class imbalance problem of the dataset is handled by the uniform distribution

based balancing scheme (UDBS). The PCA-RF combined approach gives better detection rate in both binary and multi-classification tasks in 752.67s training time.

The PCA is the principal component analysis that projects the data points in each principal component through matrix decomposition, by decomposing the high-dimensional original input matrix into its eigenvalues and eigenvectors, that corresponds to the magnitude and direction of projection of data points in a lower-dimensional manifold. Each principal component is orthogonal to the other. The major drawback is that the PCA does not support any activation function to deal with the non-linear structure of the datasets and it is mostly suitable for the linear-data types, where the network traffic of the cyber-security datasets is not always linear in type. Thus, we focus our survey studies towards autoencoder, another feature extraction strategy that supports different non-linear activation functions.

The non-symmetric deep autoencoder (NDAE) is proposed by Shone *et.al.* (2018) for the feature learning task. The NDAE is not symmetric with an encoder-decoder paradigm and contains only an encoder paradigm to learn and extract the hidden features from the dataset namely NSL-KDD. Previously, the feature values with threshold values < 20 are removed and the RF classifier is involved to classify the samples and the performance of the model is measured in terms of standard quality metrics.

A hybrid-IDS model consisting of an autoencoder (AE) and DNN is employed by Catak and Mustacoglu (2019) for the DDoS attack detection on the UNSW-NB15, where the dataset is random-sampled. The latent features are extracted by the AE and the samples are classified by the DNN, that has been experimented with different activation functions. The SGD

optimization is the back-propagation technique and cross-entropy is the objective function used in this experimental design.

To secure the massive amount of data generated by the network technology from the attackers, the authors (Hsu *et al.*, 2019) proposed to design an intrusion detection system (IDS) using the stacked ensemble learning approach consisting of three machine learning approaches namely autoencoder (AE), support vector machine (SVM), and random forest (RF). The standard datasets such as UNSW-NB and NSL-KDD are used for the evaluation. The AE compressed the network traffic features, where the compressed features are then passed to the RF and SVM based stacking model to classify the records. The model achieved a considerable performance in both the datasets.

In the research work by Aygun and Yavuz (2017), the ANN technique namely autoencoder is trained over the NSL-KDD, to predict the attack, according to the novel stochastic anomaly threshold selection approach and considerable performance is achieved in this work.

The autoencoder, a neural network-based feature extraction technique that functions like PCA (Baldi & Hornik 1989) except the autoencoder adapts different activation functions that support the transformation of non-linear network traffic signals, which gives a better prediction score. So, AE is a better choice than PCA. Moreover, each dimension of an autoencoder are independent of each other and are not orthogonal to each other. But autoencoder is highly vulnerable to noise and corruptions present in the high-dimensional network traffic. These distortions are challenging for an autoencoder that adversely affects its

performance. The AE overlearns the distortions as hidden patterns, which cause deviations in the learnt and extracted patterns, and any machine learning classifier will misclassify the samples, on learning these deviated hidden patterns. So denoising autoencoder (DAE), a variant and an improved version over traditional AE, which is highly robust to noise and corruptions is suggested to use as the feature extraction technique for our IDS model. The gaussian noise in the DAE can reduce those distortions and functions as regularizer on the encoder side to assemble the robust similar data points that belong to identical class labels in each cluster, and extract the hidden patterns in the latent manifold by reducing the dimensionality of the dataset and deviations in the latent structure. Hence, we focus our literature survey on the DAE.

On continuation of the same work by Aygun and Yavuz (2017), DAE is used to extract the patterns from the NSL-KDD dataset, and classify the dataset samples using the patterns based on the novel stochastic anomaly threshold selection approach. The DAE overtakes the traditional AE due to the denoising criterion.

The unsupervised learning strategies namely autoencoder and its variant DAE are employed by Choi *et.al.* (2019) to develop attack detection models. In DAE, 10% gaussian noise is used, wherein Both DAE and AE, RELU non-linear activation function is taken to transform the non-linear data values and both models are evaluated by the NSL-KDD, where the classification of samples is based on the threshold of the reconstruction error. The detection score of the DAE is higher than AE.

The intrusion monitoring tool is designed by Abusitta *et al.* (2019), to monitor and secure the cloud networks using the hybrid model consisting of DAE and SVM. The proposed

model contains two phases, (i.e.) phase 1. feature extraction, phase 2. attack recognition. In the 1st phase, the DAE removes the distortions from the high-dimensional network traffic and reduce the dimensionality of the dataset by learning and extracting the patterns from the BoT-IoT dataset. In the 2nd phase, the SVM classifier recognises and classify the samples using the features extracted in phase 1. The proposed hybrid model obtains 95% accuracy.

The network intrusion prevention model to predict different categories of attacks on the NSL-KDD is built by Fahimen *et al.* (2018) over the denoising autoencoder, which contains two stages. Stage1 describes the unsupervised pre-training stage, that deals with the extraction of features, stage 2 describes the supervised fine-tuning stage, that deals with the prediction of samples by the SoftMax function. The dataset is randomly sampled and the model outperforms the other machine learning classifiers with a DR of 96.85%.

The DAE and SoftMax classifier are blended to design an IDS model (Khan *et al.*, 2019) on the UNSW-NB15 dataset. The model consists of two stages 1. pre-training stage, where the features extraction task is carried out by the DAE 2. fine-tuning stage, where the attack classification task is done by the SoftMax classifier, using the features extracted from the pre-training stage and the model achieved a good performance score.

For the prediction of multisource heterogeneous attack categories in the network intrusion dataset IoTID, an intrusion detection algorithm with a combination of DAE and ELM, a type of ANN, is proposed (Wei *et al.*, 2021). Highly robust, abstract low dimensional integration of features are extracted by the DAE. The supervising learning task of recognizing

the category of network attack is done by the ELM, a fast learner algorithm. The SDA-ELM algorithm improves the accuracy and detection rate of the minority category intruders.

To enhance the standard of detection effectiveness, intrusion detection framework combining unsupervised learning based DAE and supervised learning based DNN are introduced by Lopes *et al.* (2022). The unsupervised learning is used to obtain the compressed low-dimensional representation of the network security dataset CIC-IDS, and the supervised learning is used to classify the multiclass attacks. Our proposed hybrid approach outperforms other relevant methods by maintaining 99.6% DR.

From the literature study, we come to a point that the DAE is the suitable dimensionality reduction technique for our security system. Since the network traffic is high-dimensional with noise and corruptions, DAE is the suitable feature extraction strategy to reduce the distortions and high-dimensionality of the network traffic by learning and extracting hidden patterns from the traffic and also to be noted that convolutional and recurrent network structures are mainly suited for image and sequential data types i.e. unstructured data types, whereas the architecture is not best suited for our network security datasets, that are in CSV types i.e. structured data types.

As far as, different supervised machine learning classifiers are studied in this literature summary, instead of a single classifier, an ensemble strategy that consists of a group of classifiers show better performance improvement. Over the ensemble classifiers, the gradient boosting ensemble strategy, outperforms all the other machine learning classifiers. Thus, we take LightGBM, an advanced, lighter, optimized and improved variant of the gradient

boosting ensemble strategy as the classifier for its efficiency and fastness. The classifier has various regularization techniques (leaf wise strategy, histogram-optimized strategy, sub-sampling technique, etc.) that can outperform the other ML classifiers. Thus, we finalized that LightGBM is the suitable ML classifier for our NIDS model design to classify the samples of all our cyber-security datasets.

It is to be noted from this survey, rather than a single technique, a hybrid model (Peddabachigari *et al.*, 2007) combining more than a single strategy gives better prediction score. Especially an efficient feature extraction strategy and an effective classifier can be combined and well-organized to develop a hybrid model, that can extract the patterns and classify the samples in a better way by securing a good prediction score in short training time. Thus, we combine DAE, an unsupervised learning technique and LightGBM, a supervised learning technique, together to develop a novel hybrid model that can train the network traffic samples in a faster manner and give a better detection score in short training time.

The major set-backs that exist in all the aforementioned models are most primarily they are vulnerable to noise and corruptions in the higher-dimensional network traffic by which the models could not properly learn and extract meaningful patterns and lead to wrong classification results. Besides, the methods are sensitive to the imbalanced nature of the high-dimensional datasets and could not successfully predict the samples of the minority category. By increasing the learning and predictive capacity, the model can able to correctly classify the minority samples. Firstly, DAE is proposed which is robust to noise and corruptions. The DAE is well-versed in reducing the distortions by extracting the lower-dimensional useful patterns in the network traffic. Secondly, LightGBM, which is very much lighter and faster is proposed

as a classifier, which can classify the samples with good performance. The LightGBM is well-versed in improving the predictive capacity of the model. The combined learning and predictive efficiency of those two strategies are very sufficient to deal with the imbalanced nature of the datasets. The DAE and LightGBM combined strategy can give a robust, stable, lighter, yet stronger IDS.

CHAPTER 3

RESEARCH MEHTODOLOY

3.1 Enhanced IDS Design

Figure 3.1 shows an overview of our proposed network-based IDS. The datasets samples are first pre-processed (Barreca, 2001) using one-hot encoding and normalization techniques. Subsequently, an enhanced denoising autoencoder is used to remove the deviations in the latent structure and reduce the high dimensional features by extracting salient, primitive and descriptive patterns i.e. core-structure. Finally, the LightGBM is exploited to classify the samples using the extracted optimal subset of low-dimensional features.

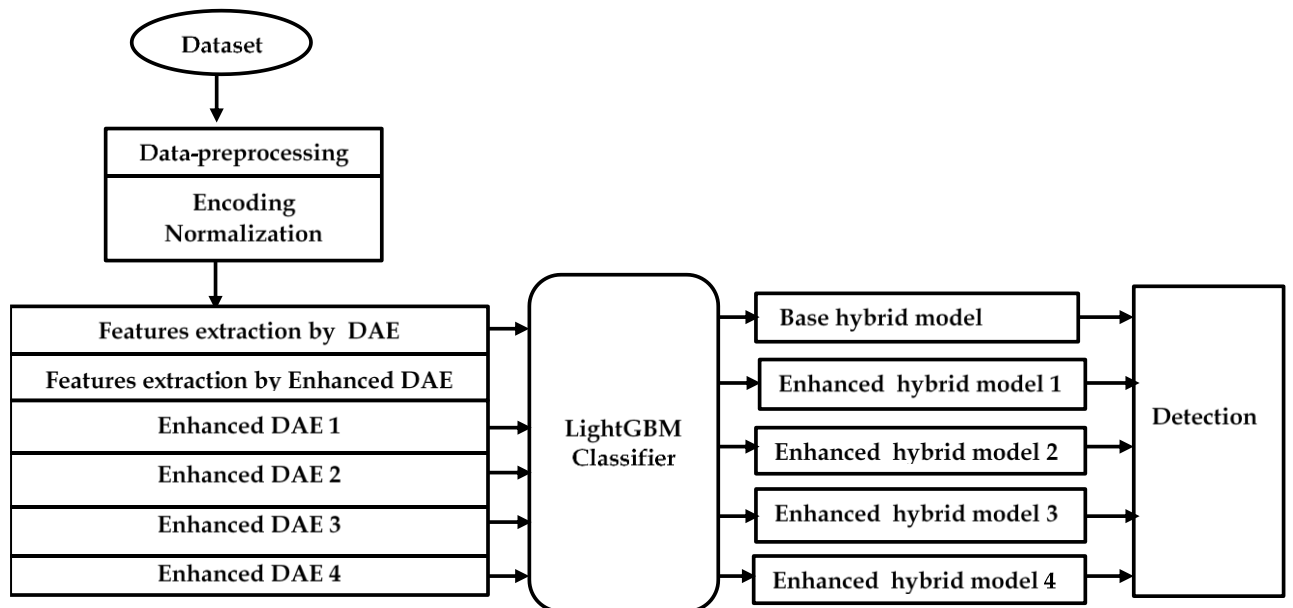


Figure 3.1: The Overall Design of the Enhanced Hybrid IDS Models

3.1.1 Data-Pre-processing

One-hot encoding and normalization are the two standard steps of the data pre-processing stage.

One-hot encoding

As the datasets consist of nominal features, the categorical values of the features are converted to numerical values using one-hot encoding by assigning different integer I to each category value of the attribute and converting it into a binary vector with the number of different categories as length. The position of the integer value I have a value one where all other positions are set as zero. E.g. The categorical values of the attribute destination port in the dataset UNSW-NB15 are shown in Table 3.1.

Table 3.1: The categorical values of the attribute

Attribute - Destination port
UDP
TCP
ICMP

The numericalized values of the categorical attribute are shown in the Table 3.2.

Table 3.2: The encoded values of the categorical attribute

ID	UDP	TCP	ICMP
1	1	0	0
2	0	1	0
3	0	0	1

Min-max normalization

The range of numerical values can vary between a minimum and a maximum, min-max normalization is used to normalize the feature values in the range of [0, 1] as follows:

$$X_n = (x_i - x_{min}) / (x_{max} - x_{min})$$

Where x_i is the feature value, X_n is the normalized feature value, x_{min} , x_{max} are the minimum and maximum feature values.

3.1.2 Performance metrics

Various criteria that are generally considered to measure the performance (Labatut & Sheriff, 2012) of the designed security models are as follows:

Confusion matrix

Confusion matrix is a $N*N$ matrix that gives the holistic view of the model performance, where N is the number of target class. The matrix relates the actual target value with those of predicted value. E.g. for a binary classification problem, the model has 2 output category and a $2*2$ matrix is formed that gives 4 values as shown in Table 3.3.

Table 3.3: Confusion matrix for a binary classification problem

Predicted values	Actual values	
	Positive (P)	Negative (N)
Positive (p)	TP	FN
Negative (N)	FP	TN

True Negative (TN): The benign network traffic is correctly classified as benign.

True Positive (TP): The intrusive network traffic is correctly classified as intrusive.

False Negative (FN): The intrusive traffic is misclassified as benign and the traffic is actually intrusive.

False Positive (FP): The benign traffic is misclassified as intrusive and the traffic is actually benign.

Based on the above four criteria, the performance metrics used to evaluate the proposed IDS models are as follows:

Detection rate: It is also known as sensitivity and recall. It is the ratio of intrusive traffic records that are correctly classified to the total intrusive traffic records in the input dataset, which is our aim. It is a very important metric in evaluating the performance of a model. The detection rate is measured as follows:

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

Accuracy: It indicates how an IDS can accurately classify the network traffic records in the dataset. Accuracy is measured as follows:

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

Precision: It is the proportion of correctness in the predicted intrusive traffic records. Precision is measured as follows:

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

F1-score: It is a harmonic mean between precision and recall. F1-score is measured as follows:

$$F1 - score = \frac{2 \times (Recall \times Precision)}{(Recall + Precision)}$$

Classification loss: The classification loss computes the difference between the actual output and predicted output and estimates the performance of the model on test set samples. Usually the cross-entropy is chosen. It is another very important criteria in measuring the detection performance. Generally, when the classification loss is low, the detection rate would be high. The model should have high detection rate close to 1.

3.2 Proposed enhanced DAE Models

Though the gaussian noise in the input layer of the DAE has the capability to remove distortions and extract robust clean meaningful patterns, the DAE could not completely remove it due to the small ratio of gaussian noise used in the DAE. It gives only partial regularization and robustness on the encoder activation of the DAE. Still there are variations in the latent space of the DAE due to the hidden unremoved distortions. As a result, the similar data points are distant apart i.e. there are gaps among the similar data points and the DAE could not completely group similar data points in each clusters of the latent structure. This led to the deviations in the latent structure which affects the quality of the patterns extracted and the predictive capacity of the model. More importantly, the usage of higher proportion of gaussian noise i.e. corrupting larger proportion of input samples collapse the entire latent structure and degrade the performance. Thus, some other additional constraints can be added on the encoder side of the DAE to suppress the deviations in the latent space. In the following ways, the deviations in the latent structure can be suppressed. They are:

1. By minimizing the larger partial derivatives in the encoder activation.
2. By minimizing the magnitude of the encoder weight matrix activation to be sparser
3. By directly transforming and projecting similar data points and filtering the dissimilar points in the encoder activation.
4. By enforcing the generated distribution on the encoder activation to be an approximation i.e. maximum likelihood of the standard normal distribution, which can minimize the deviations by enforcing continuous and complete latent space.

The research methodology deals with the description of theoretical, mathematical, algebraic properties of the additional novel strategies that are to be added to the encoder activation layer of the DAE that give four enhanced DAE models which can minimize the rate of change in the latent manifold, by removing the distortions and enhance the feature learning capacity of the DAE.

All the four models have their own unique mathematical and algebraic properties in suppressing the deviations in the latent structure of the DAE and enhancing the feature learning capacity of the DAE. Among the four enhanced models, the fourth model, enhanced DAE 4 otherwise named as generative DAE, slightly outperforms the other three enhanced models in improving the features learning capacity of the model. The fourth model, generative DAE, as the name suggests that it can extract the entire probability distribution of the network traffic in the latent space, which is continuous and complete in nature, unlike the other three enhanced DAE and traditional DAE models, which can output discrete real vector values in the latent structure, which may be discontinuous and incomplete in nature. The generative DAE due its generative nature can generate meaningful sampled points and outperforms the other models, since the generative models always outperform the stochastic and deterministic models by generating the quality enriched patterns. The four enhanced DAE models are described and derived as below:

3.2.1 Enhanced DAE model 1

By minimizing the larger partial derivatives on the encoder activation of the DAE, the deviations in the latent space can be suppressed. To enforce the partial derivatives to be smaller, the jacobian gradient norm is inserted on the encoder activation of the DAE.

The overall design of the DAE with jacobian gradient norm is visualized as shown in Figure 3.2.

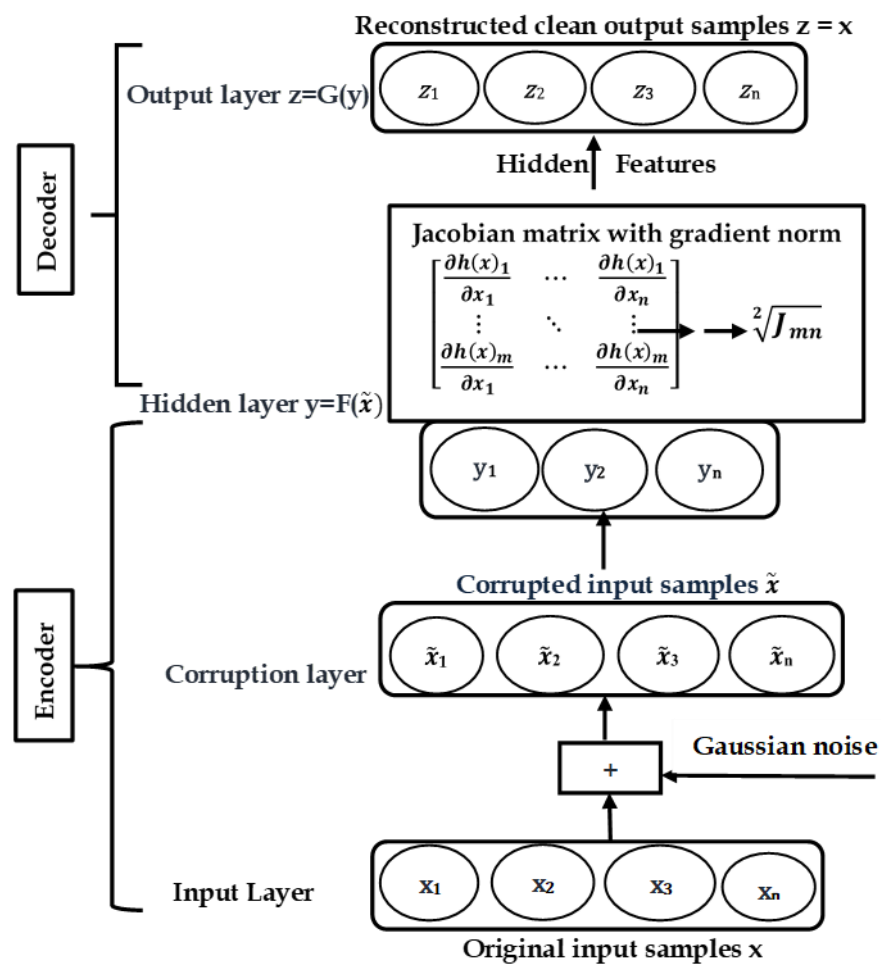


Figure 3.2: The Architecture of the Enhanced DAE 1

The jacobian matrix ([Ford, 2015](#)) computed at the encoder activation of the DAE contains the partial derivatives of the extracted hidden patterns corresponding to the actual input data patterns. The partial derivatives denote the gradients generally refer to the rate of change of an output function from the corresponding input function. That is what it refers to the rate of change of a hidden layer representation WRT to the corresponding input patterns.

The higher-order partial derivatives denote the variations in the latent manifold region from the original input manifold. The distorted representation makes the network unstable. The smaller partial derivatives denote the less deviation or no deviation.

By minimizing the larger partial derivatives, important patterns, compact representations and descriptive structure can be captured and transformed from the original high-dimensional space and well projected in the non-linear latent subspace, by removing the gaps among the similar data points that are distant apart and grouping of similar data points that belong to identical classes in each distinctive cluster labels of the latent space.

If the partial derivative value is closer to zero, then no major change is observed in the mapped latent representation WRT original representation, whereas if the value is closer to one, then a major change is observed in that mapped latent representation. Thus, the data points are not properly mapped and deviations are observed in the latent structure.

The gradient norm ([Rego and Lupu, 2021](#)) denotes the square root of the sum of squares of all the partial derivatives in the encoder layer i.e. latent data points WRT input data

points. The gradient vector norm has the important property of being invariant (Goodfellow *et al.* 2009), and hence we are utilizing its algebraic and mathematical property in DAE.

The gradient norm applied on the jacobian matrix enforces squishing pressure on the higher-order partial derivatives i.e. magnitude close to 1, shrink it to smaller singular values i.e. magnitude close to 0. Hence the space among the similar data points gets squished and the similar data points of the identical classes are clustered together in their respective clusters due to the pressure applied by the gradient norm. By squishing and reducing the gaps among the data points that are distant apart, large number of similar data points that belong to identical classes are grouped together to form distinctive clusters in the latent manifold which suppress the deviations in the latent manifold. Thus, the latent manifold extracts the intrinsic and primitive patterns of the original high-dimensional network traffic, which enriches the quality and structure of the features extracted by the model and enhances the features learning and predictive capacity of the model.

The jacobian matrix is thus computed as follows:

$$\text{Jacobian Matrix: } J_h(x) = \frac{\partial h(x)_m}{\partial x_n} = \begin{bmatrix} \frac{\partial h(x)_1}{\partial x_1} & \dots & \frac{\partial h(x)_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h(x)_m}{\partial x_1} & \dots & \frac{\partial h(x)_m}{\partial x_n} \end{bmatrix}$$

which denotes the partial derivatives of the hidden features' WRT to the input features.

$$J_h(x) = \frac{\partial(h_{i=1, \dots, m})}{\partial(x_{i=1, \dots, n})} = \frac{\partial h_m}{\partial x_n}$$

$$\text{where } J_{mn} = \left(\frac{\partial h_m}{\partial x_n} \right)^2$$

Where $i = 1$ to m denotes the *input* (x) - summed over i input data points

$j= 1$ to n denotes the *hidden features (h)* - summed over j hidden points

Consider $h(x)$ as a vector-valued function, each hidden unit has a separate gradient vector.

The gradient norm thus computed is shown as follows:

$$\text{Gradient norm of the jacobian matrix } \lambda F_{mn} = \sqrt[2]{J_{mn}}$$

where λ represents denotes the weight of the gradient norm applied.

Parameter alpha is the tuning hyperparameter that denotes the strength of the gradient vector norm of the Jacobian matrix. It denotes the quantity of contracting pressure, that is to be applied on the larger partial derivatives. It denotes the magnitude and direction of the pressure applied on the gradients. The magnitude is the proportion of the contracting pressure applied, whereas the direction denotes reaching the bottom of the steepest descent to maximize the objective function. The contraction ratio maximizes up to the point of saturation of the activation units.

The jacobian matrix on the encoder layer of the enhanced DAE is thus calculated as

$$z_m = w_n x_n$$

$$h_m = \phi(z_m)$$

where ϕ denotes the non-linearity in activation function.

The dot product of the n^{th} attribute value and its associated weight gives the m^{th} hidden unit.

$$\frac{\partial h_m}{\partial x_n} = \frac{\phi(z_m)}{\partial x_n}$$

$$\text{Jacobian norm} = \sum_{mn} \left(\frac{\partial h_m}{\partial x_n} \right)^2$$

The first strategy suppresses the deviations by minimizing the larger partial derivatives on the encoder activation of the DAE.

3.2.2 Enhanced DAE Model 2

The second strategy focuses on suppressing the deviations by enforcing the weight matrix of the encoder activation to be sparser. i.e. By minimizing the magnitude of the encoder activation weights, the deviation can be suppressed. To enforce the weight matrix to be sparser, iterative thresholding function is inserted on the encoder activation of the DAE. Figure 3.3 shows the architecture of the DAE with ITF.

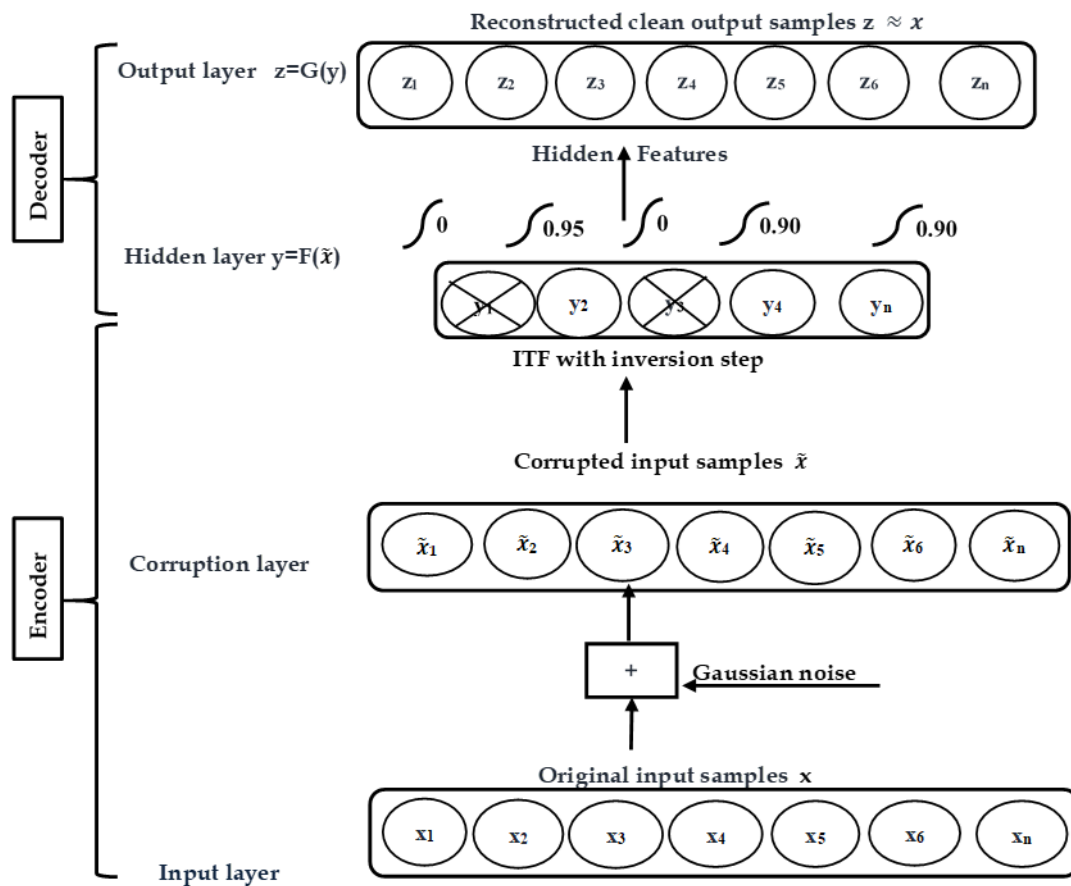


Figure 3.3: The Architecture of the Enhanced DAE 2

By inserting ITF i.e. iterative thresholding function in the encoder layer activates only n -number of hidden units with largest activation values randomly at every iteration and deactivate the rest of the nodes in both forward and backward phase to learn and extract the

patterns, which enforce the weight matrix to be sparse and regulate the activation weights of the encoder layer. By taking the highest activation nodes in the encoder layer guarantee sparse representation for each input data values and achieve exact sparsity in the encoder activation.

The ITF ([Fornasier and Rauhut, 2008](#)) algorithm generally involves two steps:

1. The first step involves a dictionary that sparsely denotes the input data.
2. The second step involves transforming the original input space to a new latent feature space.

Let E is an estimated support set, E^c is the complement of the support set. W^c is the inverse of W (weight), sup_n is the function that contains the indexes of the n -largest magnitude coefficients of input features, h_E is the hidden vector obtained by penalizing the values of h hidden units to the indices of E and W_E is the weight matrix got by penalizing the columns of W to the indices of E .

In the feed-forward activation phase, after calculating the hidden vector code $h = W^i x$, the n -highest nodes are identified by adjusting the threshold values of the encoder activation units using the ITF strategy and sorting the activation values of the RELU hidden units with thresholds that are adaptively adjusted by the ITF, until the n -highest activation nodes are identified. This gives a vector of activations by the support set of $sup_n(W^i x)$ and the support set is defined by $\alpha sup_n(W^i x)$ where it is further weighted by the regularization operator alpha.

For a fixed input data vector x and its associated weights W , starting from $h^0 = 0$ (hidden units), ITF iteratively obtains the sparse representation of $x = Wh$ that follows the below stages:

- Perform the feedforward phase of the DAE and then calculate the hidden layer activation units. It is shown in equation $h = W^i x$
- The support set $E = \text{sup}_n(W^i x)$ is estimated primarily by the ITF algorithm.
- Find the n-highest activated nodes (activations) in the encoder layer and set rest of the nodes to zero.
- Let E denotes a support set that contains the n-largest activation node values and E^c denotes the pseudo-inverse set of E contains the rest of the activation values.

The ITF algorithm learns a dictionary that satisfies $x = Wh$ using the estimated support set $E = \text{sup}_n(W^i x)$. After computing the support set of h as E , we constraint W (weight matrix) to the indices contained in the support set E and obtain W_E . The pseudo-inverse of W_E are used to evaluate and obtain the non-zero values by minimizing $(x - W_E h_E)^2$. In the inversion step, after estimating the support set, $E_h = \text{sup}_n(W^i x)$, the elements of the dictionary are updated to obtain the corresponding non-zero values in the support set. Using the support set, the non-zero, i.e. strongest nodes are sorted and obtained as $h_1, h_2, h_3, \dots, h_n$.

Compute the outcome and reconstruction cost using the n-highest encoder units. Back-propagate the gradient cost using the n-highest hidden units used in the feed-forward phase, that is defined by the $\text{sup}_n(W^i x)$ and iterate.

Extract the features $h = W^i x$ supported by $E_h = \text{sup}_n \alpha(W^i x)$ using the αn highest nodes and rest of the nodes are set as zero, that supports sparse representation for each input data. As a last step, the support estimation is refined and repeated till convergence and achieves exact sparsity in latent space representation

$$h_{(E)}^c = 0 \quad \text{where } E = \text{sup}_{\alpha n}(h)$$

The conventional sparse coding techniques involve complex matrix operations and do not have provision to control sparsity level in the encoder activation nodes and a large number of dead nodes are formed. The traditional sparse penalty constraints the hidden layer nodes to be dead for most of the time. The conventional sparse constraint can only partially assure the sparsity in the hidden activation nodes and no exact sparsity is achieved in the encode activation layer and there is no guarantee that every input data achieves sparse representation. The sparse coding using ITF ([Blumensath and Davies, 2008](#)) is the latest trend in unsupervised features learning, whereas it involves only simple matrix operations and the sparsity level in the encoder activation layer can be controlled using the parameter ‘n’. In the enhanced DAE, after several iterations, using the support set, we use the αn largest hidden units to obtain the sparse features where α denotes the strength of the sparsity level.

The node is considered highly active only if its activation value is close to 1, whereas the node is considered inactive if its activation value is close to 0. The n-highest nodes are retained by the ITF function to impose sparsity in the encoder activation and achieve sparse representation for each input training point and penalize the weight matrix to be sparse and regulate the activation weights.

The recent improved, optimized and controlled sparse effect can be enforced and combined with DAE to suppress the deviations on the latent manifold and extract more quality meaningful patterns and descriptive structure from the high-dimensional network traffic. The extracted features are used to train the LightGBM classifier on the top of the encoder layer.

The resulting representations achieve state-of-the-art classification results, solely by enforcing n- sparsity in the hidden units.

The ITF ([Gregor and LeCun, 2010](#)) adjusts and set the threshold of activation values of only the n-nodes to the higher values closer to 1 and penalizes the activation values of the rest of the nodes to lower values closer to zero. The n-number of units with the high level of activation values generally have higher capacity in extracting primitive, salient patterns and in-depth-core structure of the network traffic.

By constraining the weights of the encoder activation using the ITF, the rate of change in the latent space can be suppressed. This enables a large number of similar data points that belong to identical class labels are grouped in each distinctive cluster of the latent space, that enhance the feature learning capacity of the DAE, which further boost the predictive performance of the classifier.

3.2.3 Enhanced DAE Model 3

By reducing the space among the similar data points that are distant apart and grouping of larger number of similar data points that belong to identical classes and filtering dissimilar data points in each distinctive cluster, suppress the deviations in the latent space of the DAE. This can be achieved by using the data-pairwise similarity distance weight. By enforcing strong bonding among the data points using the data-pairwise similarity distance weight, the feature extraction capacity of the DAE can be improved.

Considering strong relationships among the data points and filtering out the overemphasized i.e. weak relations are the major highlight in this research work. The strong relation is termed as the pairwise-similarity distance weight ([Shimada et al., 2021](#); Fred and Jain, 2006) and it estimates the relation between the data pairs. The relation weight S_{ij} between the data pairs (x_j, x_i) is measured by the pairwise similarity distance formula as shown in

$$S_{ij} = \frac{1}{\|x_j - x_i\|^2}$$

First of all, construct a relation set for the data points in each dimension of the latent manifold. The data points with similarity weight close to 1 are included in the relation set, whereas the data points with weight close to 0 are not included in the relation set and are being filtered by the scalar parameter alpha. The pairwise-distance weight varies between [0-1] in the step rate of 0.02.

On the latent manifold, data relation is finetuned based on the pair-wise similarity distance weight defined on the latent representation and the relation sets are determined based on several iterations. The architecture of the enhanced DAE 3 otherwise known as relational DAE is shown in Figure 3.4.

Relational-denoising autoencoder with weighted reconstruction cost
 $S_{ij} \| x_j - x'_i \|^2 \dots S_{ik} \| x_k - x'_i \|^2$

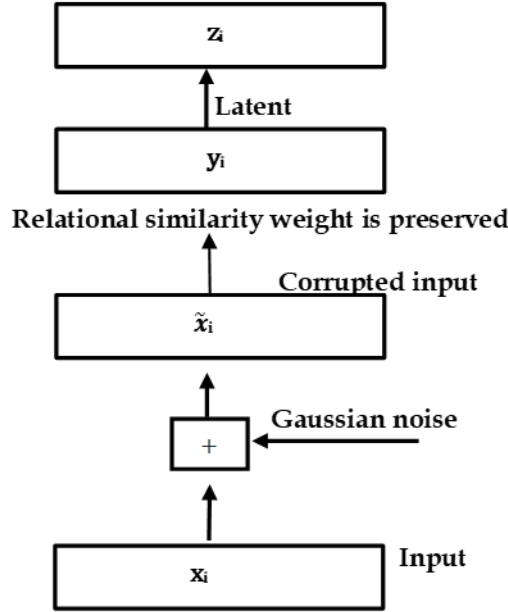


Figure 3.4: The Architecture of the Enhanced DAE 3

The figure is explained as follows: In the traditional DAE, the data point x_i is only involved in the reconstruction of itself and the reconstruction cost is measured to be the distance between x_i and x'_i i.e. $\| x_i - x'_i \|^2$. In the proposed DAE, data point x_i is used to reconstruct the set of neighboring similar data points in the relation set, thus minimizing the reconstruction cost to a possible extent. For e.g. the reconstruction cost between the data points x_j and x'_i is measured to be the weighted distance i.e. $S_{ij} \| x_j - x'_i \|^2$.

The data relation ([Wang and Sun, 2015](#)) is initialized by measuring the data pair-wise similarity distance between the data points in the latent space and constructing a relation set for each feature dimensions. A relation set consists of N-nearest neighbours (Indyk and Motwani, 1998).

A relation set otherwise can be called as reconstruction set is constructed for each dimension and each relation set consists of identical data points that correspond to the similar class label which can be projected in each dimensional distinctive clusters, by reducing the space among the similar data points that are distant apart, such that the latent manifold contains distinctive clusters with identical data points through which the intrinsic structure, prominent behaviours and characteristics of the original high-dimensional input data structure can be extracted. Maintaining mutual data relations further enhance the quality of the patterns extracted.

The proposed model filters the least weighted data points in each dimension using the scalar parameter alpha defined in the RELU activation function. Alpha is denoting the scalar parameter to control and filter the weak similar or dissimilar data points. The parameter alpha that is inserted in the activation function RELU varies in the range between [0-1] in step size of 0.02 as like the similarity weight. The value close to 1 denotes the strong relation among the data points i.e. they are similar data points. The value close to 0 denotes the weak relation among the data points i.e. they are dissimilar data points and it is filtered from being included in the relation set by the parameter alpha.

To model the data relation, the decoder reproduces a set of identical data points defined as $\Omega_i = \{j, k, \dots\}$ with specific weights $\{s_{ij}, s_{ik}\}$ for a data point x_i defined through one reconstructing the others in a relation set, Thus the weighted reconstruction cost C is given below.

$$S_{ij} \| x_j - x'_i \|^2 \dots \dots S_{ik} \| x_k - x'_i \|^2$$

The weighted reconstruction cost is back-propagated to update and finetune the weights to minimize the reconstruction cost and information loss i.e. impurity, where impurity is defined as the ratio between number of similar data points i.e. data points that correspond to identical class labels to the number of dissimilar data points i.e. data points that corresponds to other non-identical class labels. This shows the significance of considering data relation on the latent manifold during the iterative learning, which is the major motivation of this work. The proposed methods can discover more complex structure by iteratively exploring the data relation.

Each data point is used to reconstruct a set of identical samples in the relation set rather than itself due to the similarity weight, thus minimizing the reconstruction and information loss and so the optimal solution can be reached. The relational DAE can find large number of similar data points in each distinctive cluster around the lower-dimensional manifold by reducing the gaps and suppressing the rate of change in the extracted patterns, which results in a meaningful compact structure. On continuous iterations, the proposed DAE can find large number of similar data points forming distinctive clusters in the latent manifold. These features contribute to the LightGBM classifier to achieve good prediction score.

3.2.4 Enhanced DAE Model 4

By enforcing the latent space distribution to be an approximation i.e. maximum likelihood of the standard normal distribution, through which continuous and complete latent space is obtained by suppressing the deviations in the latent space i.e. reducing the gaps among the similar data points that are distant apart and grouping large number of similar data points that

belong to identical classes in each clusters of the latent manifold, thus enhancing the features learning capacity of the model.

The traditional DAE encodes discrete set of data points i.e. discrete real vector values in the latent representation, which gives an incomplete and discontinuous latent structure, that lacks information and useful patterns. To have a full complete and continuous latent space, the DAE is enhanced to be an enhanced DAE 4 otherwise known as generative DAE ([Im et al., 2018](#)). The architecture of the generative DAE is shown in Figure 3.5.

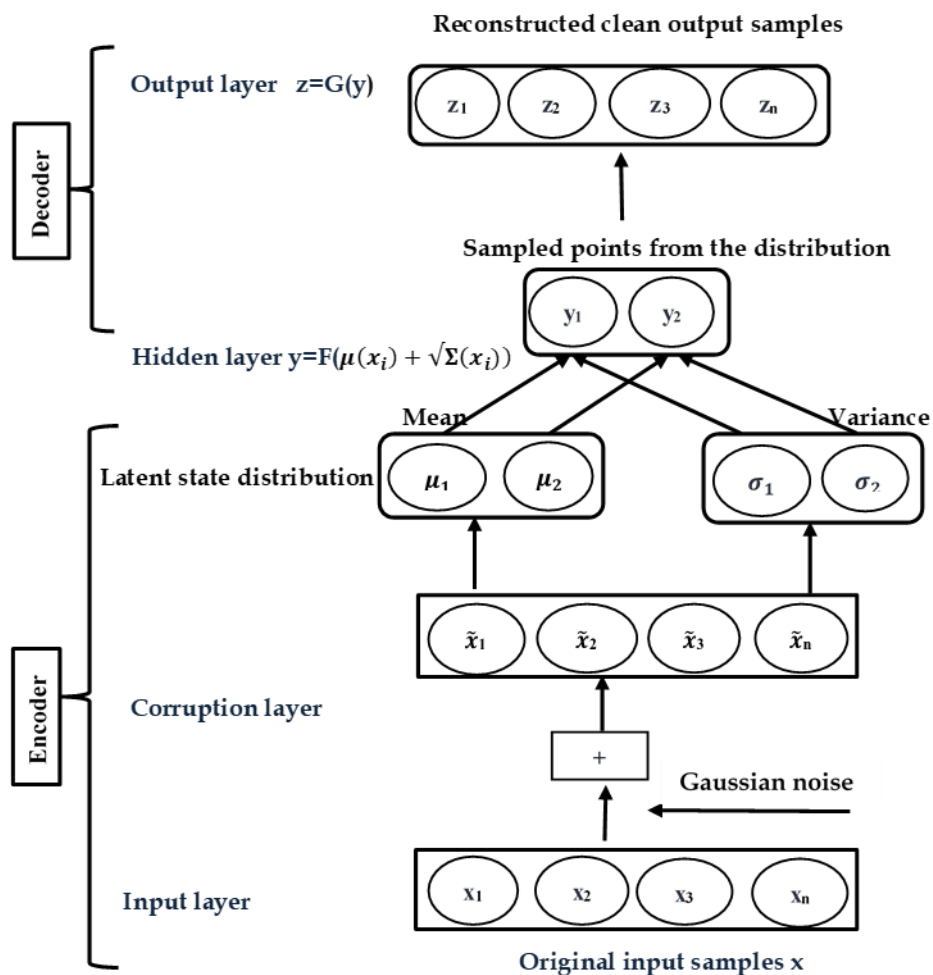


Figure 3.5: Architecture of Generative DAE

The generative DAE, while transforming the data points, the original input data distribution of the cyber security datasets is enforced to be a standard normal distribution i.e. zero mean and unit variance. The mean and variance of each feature in the input dimension is transformed into a normal distribution by using the formula as shown below

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

$f(x)$ - Probability density function

x - Actual input distribution

μ - Mean, σ - Standard deviation

The value is standardized by taking the z-score of the feature value i.e. normal distribution value as shown in the formula

$$z = \frac{x-\mu}{\sigma}$$

The encoder instead of outputting a single real discrete value for each dimension to describe each latent feature, it outputs a probability distribution for each latent feature. (i.e. for x_i) it outputs two encoder values that is close to a standard normal distribution. 1. mean 2. variance that describes the probability distribution ([Sugiyama et al., 2019](#)) for each dimension of feature in the latent manifold.

It is a probabilistic encoder with parameters μ and $\sqrt{\Sigma}(\sigma)$ and it is shown in the equation below, thus each latent feature has a continuous probability distribution of the network traffic which is also regular and complete in nature, from which the data points are generated and sampled, that give the meaningful core structure of the entire network traffic,

$$z = \mu(x_i) + \sqrt{\Sigma}(x_i)$$

We change the log variance to the standard deviation for the numerical stability by taking the log exponent of the variance function. we can draw samples from $z()$, that gives continuous latent structure from which the data points can be sampled (Dawid, 2011).

The sampled points generated and drawn from the $z()$ gives a continuous and complete latent structure. In order to ensure the generated distribution and the sampled points to be approximation or maximum likelihood of the original input distribution and the data points, the bayesian inference strategy is used. (i.e.) Infer good values of z , given observed data x .

Bayesian inference

The bayesian inference strategy ([KingRobert and Eckersley, 2019](#)) is used to approximate the latent space i.e. generated distribution, to the standard normal i.e. prior distribution. The mean and variance of the latent space distribution are approximated to the prior standard normal distribution by the bayesian inference strategy and the approximation difference is measured by the KL-divergence term and it is almost closer to zero. The inference strategy is shown as follows:

- The prior distribution $p(x)$ -The standard normal input distribution - Expected

$$N(0, I)$$

- The posterior distribution $p(z)$ - The generated latent distribution - Extracted

$$N(\mu(x), \sigma(x)I)$$

- $P(z/x)$ - Maximum likelihood - Given the prior distribution, the generated distribution to be an approximation of the original distribution.
- Let $p(z)$ generated from $p(x)$ have maximum likelihood distribution and it is represented using the formulae

$$p(z) = \int \dots \int p(z|x)p(x) dz$$

- To measure the divergence between the original prior distribution, $\mu = 0, \sigma = 1$ and the likelihood approximated distribution $x \sim N(\mu, \sigma^2)$, the KL-metric is used to estimate the divergence and it is always close to zero. The KL-divergence metric is given in the equation as:

$$KL(z||x) \stackrel{z}{=} p(z) - p(x)$$

For each dimension in the latent space, mean close to zero enforces the generated sampled similar data points of the identical class to be centred around zero for each clusters in the latent structure, while variance close to 1 enforces the subsequent generated similar sampled similar data points of the identical class to spread and distributed uniformly from the centred point in each distinctive clusters of the latent space, thus giving a continuous and complete latent structure form which the primitive, descriptive core-structure of high-dimensional network traffic can be extracted. The generated distribution is quite tighter and the latent space is tightly bounded. This maintains continuity and completeness among the data points in the latent space and enhances the features learning capacity of the DAE.

The patterns extracted from the enhanced DAE models are passed to the LightGBM classifier to classify the samples and the performance is measured in terms of standard quality performance metrics. The continuous probability distribution and complete latent representation is shown in Figure 3.6.

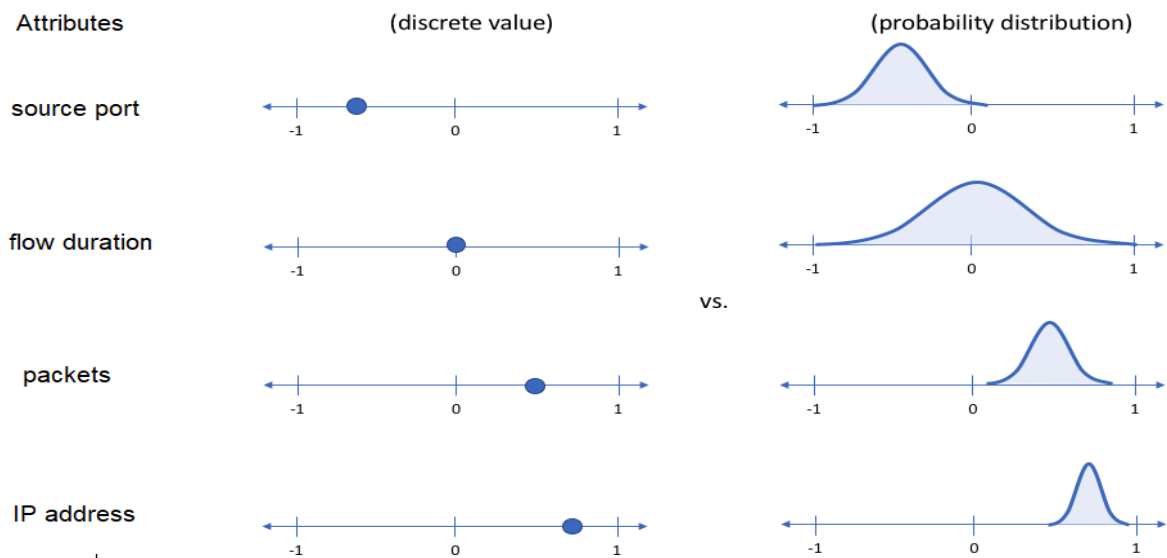


Figure 3.6: The Continuous Latent Representation of the Generative DAE

3.3 Hyperparameter Tuning

The hyperparameters (Bergstra *et al.* 2011) are the tuning parameters that can be finetuned to get the reliable results. The experimentations are iterated several times to finetune and fix the parameters to give the improved performance score. Various hyperparameters are fine-tuned to improve the feature extraction and classification processes as follows

Hidden layer and Neurons: The DAE with a single hidden layer is used to extract the features of different datasets. However, the number of neurons differ for different datasets and it depends on the input features.

Weight and Bias: In all our experiments, the weights are randomly initialized while the bias are set to zero.

Activation Function: The RELU and sigmoid activation functions are used to transform the non-linear input signals to the processed format (latent encoded signals).

Optimizer: Optimizer is important to minimize the reconstruction loss by updating the weights and learning rate. It is used within the SGD, to optimize and accelerate the back-propagation process. Nadam is the optimizer used in all our datasets.

Gaussian Noise: Gaussian noise in DAE is used to remove noise and corruptions in the network traffic. 10% - 20% of gaussian noise is used for eight different benchmark datasets.

Batch Size: Instead of training a single record at each iteration in an epoch, a cluster of records are grouped together as a batch to increase the training speed of the SGD.

Epoch: The entire dataset i.e. each record in a training dataset is both forward and back-propagated to complete those two passes in a network once to finetune and update the weights to minimize the loss. An epoch is comprised of a large number of batches.

Three most hyperparameters considered in the LightGBM are described as follows

Number of trees: It denotes the number of base learners or DT required to learn the patterns and classify the samples.

Depth of trees: It denotes the height i.e. branches of the DT that can be extended to solve the classification problem.

Learning rate: It denotes the step size of the DT to learn the patterns and classify the samples with minimum prediction loss.

CHAPTER 4

RESULTS & DISCUSSIONS

4.1 DATASETS

To evaluate the effectiveness of the proposed IDS models, the following datasets are used. These are the standard public benchmarking datasets that are used by most of the machine learning researchers to evaluate their different IDS models.

ISCX-TOR 2016 ([Mamun et al., 2016](#)): The tor network is known for providing privacy to users by concealing the user's identity. The users are allowed to communicate while keeping their internet activities unmonitored. The tor network is often misused by hackers for illegal activities. ISCX-TOR was created to capture tor traffic which downgrades the security, confidentiality and privacy of internet users. Various real-world network categories of web browsing, email, chat, video and audio streaming, file transfer, VoIP and P2P have been included as a set of tasks in this dataset. The dataset consists of 152,028 samples and 28 features with an output label. In total, there are 129,478 non-tor traffic samples and 22,552 tor traffic samples.

UNSW-NB15 ([Moustafa et al., 2015](#)): UNSW-NB15 was created by the cyber range lab of the Australian Centre for Cyber Security (ACCS). The dataset contains 47 features and an output class label. The features in the data set are classified into six categories as follows: table flow features, basic features, content features, time features, additional generated features, labelled features and they are extracted using network flow analyzers such as argus and bro-IDS tools.

There are in total 2.54 million samples with two class labels as normal or attack. It consists of 2,217,764 normal class samples and 321,283 attack class samples. Through random sampling, 221,776 normal class samples and 96,385 attack class samples are considered for our research.

ISCX-URL2016 ([Lashkari et al., 2017](#)): ISCX-URL2016 was discovered by the UNB-Canadian Institute of Cybersecurity (UNB-CIC) for the harmful URL's detection in web-related activities, since the deadly harmful URL's on the web is the main platform for online illegal activities. The dataset contains 114,400 samples and 79 features with an output class label. The target class label consists of benign class (35,000 samples) and four attack classes including spam URL (12,000 samples), phishing URL (10,000 samples), malware URL (11,500 samples) and defacement URL (45,450 samples). The URL samples in the dataset are collected from the following sources: over 35,000 benign URL's from the alexa top websites, around 12000 spam URL's from the WEbspam-UK2007 dataset, around 10,000 phishing URL's from the open phishing website, more than 11,500 malware URL's from the DNS-BH malware websites, and more than 45,450 defacement URL's from the hermitic web crawler.

CIDDS-01 ([Ring et al., 2017](#)): CIDDS-01 is a network security dataset containing the network traffic samples of both legitimate and intrusive behaviors of denial of service, brute force, port scan, and ping scan attacks. It is developed in an open stack virtual environment with different servers namely email server, web server, HTTP server. CIDDS-001 contains 28 million samples, but many samples are duplicated. The duplicated samples are removed. Through random sampling, 225,000 normal samples, 23,464 port scan samples, 146,800 DoS samples, 6,090 ping scan samples and 7,440 brute force samples are taken for evaluation. The dataset consists of 12 features with an output class label.

CIDDS-002 ([Ring et al., 2017](#)): CIDDS-002 is a network security dataset containing two weeks of network traffic samples. This dataset is taken specifically for the port scan attack detection. The dataset contains 16 million samples, but many samples are duplicated. Through random sampling, 481,322 port scan attack samples and 611,970 normal samples are taken for evaluation. Similar to CIDDS-01, CIDDS-02 consists of 12 features with an output class label. The testbed architecture of both datasets consists of three subnets reflecting organizational structure namely 1. server subnet 2. developer subnet 3. office subset. The developer subnet contains various clients such as nmap tools that exhibit port scan behavior. The office subnet consists of window clients. The server subnet includes three main servers namely file, email and web servers, which provide service to the clients.

BoT-IoT ([Koroniotis et al., 2018](#)): The BoT-IoT dataset was developed by the UNSW Canberra Centre for Cyber to detect various malicious activities in the IoT network traffic. The dataset possesses the following characteristics: 1. realistic testbed configuration 2. realistic traffic 3. labelled data 4. IoT traces 5. diverse attack scenarios 6. full packet capture 7. new generated features. The dataset contains various IoT attacks including DoS, distributed DoS, reconnaissance, theft and normal network traffic. The testbed architecture consists of three components namely 1. network platform 2. simulated IoT services 3. forensics analytics. The network platform contains virtual machines to generate network traffic. The IoT devices include devices such as weather stations etc. The forensic analytics include tools namely argus to extract the features. It contains over 3.6 million records with 43 attributes and an output class label. There are 1,926,624 DDoS samples, 1,650,260 DoS samples, 91,082 reconnaissance samples, 70 theft samples, and 478 normal samples.

IoTID-2020 ([Ullah et al., 2020](#)): The advancement of IoT devices led way for the intruders to launch cyber-attacks against IoT devices. The IoTID-2020 dataset was developed to detect malicious activities in the IoT networks. The testbed architecture contains various interconnected IoT devices that form smart home environment. The dataset consists of 80 network and flow-based features and an output class label. It contains 625,783 samples with five class labels (one normal class and four attack classes). There are 40,073 normal samples, 415,677 mirai samples, 75,265 scan samples, 59,391 DoS samples and 35,377 MITM ARP spoofing samples.

CIC-IDS-2017 ([Sharafaldin et al., 2018](#)) was created by the Canadian Institute of Cybersecurity. It contains the abstract behavior of users based on various protocols like HTTP, HTTPS, FTP, SSH, email protocols and satisfies the eleven characteristics namely anonymity, attack diversity, complete capture, complete interaction, complete network configuration, available protocols, complete traffic, feature set, metadata, heterogeneity, and labelling. The wednesday captured network traffic dataset that consists of 692,703 samples is taken for our evaluation. The CIC-IDS dataset totally contains 5 days of network traffic. We have taken only the wednesday network traffic and the other four days network traffic have been skipped since the attack types given in those are similar as in the other seven benchmarking datasets taken for evaluation. The wednesday network traffic contains the various DoS attack varieties, which are missed in the other aforementioned datasets. We skip the four days network traffic, since we don't want to evaluate the same attack categories given in the other aforementioned benchmarking datasets.

It includes the results of network traffic based on the time stamp, source and destination IPs, source and destination ports, protocols and attacks. The B-Profile system has been used in this dataset to profile the naturalistic real-time network traffic and the abstract behavior of humans. The testbed architecture contains modem, firewall, switches, routers, and a variety of operating systems such as windows, ubuntu and mac OS X. There are 78 numerical features with an output class label in total. The output class label consists of benign traffic along with six attack types (i.e. 440,031 benign samples, 5,796 DoS Slow Loris samples, 5,499 DoS Slow HTTP Test samples, 231,073 DoS Hulk samples, 10,293 DoS Golden Eye samples and 11 Heartbleed samples). This dataset is taken specifically for the DoS and heart bleed detection.

The datasets are partitioned into 80% training set to train our model and 20% test set to validate our model performance. From all the aforementioned datasets, the following attacks (Ahmad *et al.*, 2009) are successfully predicted by our IDS models:

Heartbleed: Heartbleed is a major attack in the open SSL cryptographic open-source code library where the confidentiality and privacy of user data will be leaked out.

Denial of service (DoS) attack: The intruder blocks the network traffic by injecting continuous malicious requests and preventing legitimate users from accessing it.

Botnet attack: Botnets are network of computers infected by malware that allow the intruders to control them.

Distributed denial of service (DDoS) attack: DDoS is same as DoS but multiple intruders from multiple sources are involved.

Port scan attack: The intruders scan numerous server ports to find an active port to launch an attack on the network services available on a host machine.

Defacement URL: The URL modifies the visual appearance and original contents of legitimate social websites and spoils their reputation.

Phishing URL: The phishing URL sends fake, dangerous emails to benign users, i.e. it pretends to be from authorized sources to steal their personal information such as PIN, passwords etc.

Spam URL: The spammed URL spreads irrelevant, junk messages to a large number of legitimate users to spread fake advertisements, false messages etc.

Malicious URL: The malicious URL pretends to be safer, but contains malicious source codes and voluntarily direct the legitimate users to malware websites in a back doorway with aim of possessing a threat to the system.

Reconnaissance: It is a type of scanning attack, that scans the victim ports on the system intending to launch a threat on the network system (through the vulnerable ports).

Theft: It is a common attack in the network traffic that the intruder hacks the legitimate user's password, PIN's etc to possess fraudulent activities and transactions.

Brute force attack: The attackers hack the passwords of an authorized user by trying all combinations of passwords or passphrases with the hope of eventually guessing it correctly.

Ping scan attack: The intruders ping many hosts at the same time to identify whether the hosts belong to the same network and use their IP addresses to launch an attack.

ARP spoofing: ARP spoofing is a man in the middle attack where the attackers send forged ARP messages on behalf of the sender to hijack the information sent in between two users.

Scan: Various smart devices in the IoT network are scanned in an unauthorized way by intruders to exploit the weak points. The different types of scanning attacks in the IoT environment are IP address scanning, port scanning, and version scanning etc.

Mirai: Mirai is a type of botnet that attacks smart IoT devices such as IP cameras, home routers and ARC processors. It alternates the data flow from the authorised clients or users to the malware-infected device.

The eight standard benchmark datasets taken for the evaluation contains modern, realistic network traffic of real-world category with updated attack types. Out of these eight datasets, three datasets, namely CIDDS-002, ISCX-Tor2016 and UNSW-NB15, are utilized for performing binary classification task. The rest of six datasets are used to perform multi-classification task. All the benchmarking datasets contain the traffic patterns of the entire network system. On evaluating the statistical analysis of all the datasets, (Ring *et al.*, 2019) (Moustafa *et al.*, 2019) the network traffic of those are developed from the common background and has similarity in the network configuration, network profile system, client/server set up, testbed scenario, operating system, types of protocols used, abstract behaviour of users scenario, traffic diversity, feature set, labelled data, network environment, extracted flow traffic, network-flow generators, and extractors. Moreover, all the datasets are non-linear and structured type (CSV format). Both DAE and LightGBM can handle and perform well on the structured datasets. The recent datasets that are developed between 2015 to 2020 are taken for evaluation since it contains the modern network traffic of the real-world scenarios and updated attack diversity. The datasets developed in recent years are the updated version of the previous years, thus over-coming the shortcomings of the previous ones and contain the similar structure, patterns of the previous ones in addition with the new features (Hindy *et al.*, 2020). There are minor variations among these datasets, mainly in the attack diversity, which are acceptable. Each different dataset is adjusted with different training parameters to achieve reliable detection score.

4.2 Experimental setup

The software and hardware setup for the proposed models are shown in Tables 4.1 and 4.2.

Software setup

Table 4.1: The software setup for the proposed models

Operating System	Windows 10 64 bit
Code Editor	Jupyter Notebook
Machine Learning Packages	LightGBM, Pandas, Matplotlib, SciPy, Sklearn, Seaborn, Keras with TensorFlow, NumPy
Implementation Environment	Anaconda Software
Scripting Language	Python 2.6

Hardware setup

Table 4.2: The hardware setup for the proposed models

Processor	Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 2.00 GHz
System Type	64-bit operating system
RAM	8GB RAM

ML Packages

The ML packages and scripting languages (Pedregosa *et al.*, 2011) used in our models are briefed below:

Python: Python is the main scripting language, contains many built-in libraries, useful for data science, data mining, and machine learning applications. Many machine learning libraries are scripted by the python scripting language. Some of the libraries included are TensorFlow, Keras, Scikit-learn. The python programming language is more flexible, powerful and easy to implement different machine learning models.

Anaconda: The anaconda is free and open-source software built on python language, and useful for various data science and machine learning applications. It consists of over 1400

machine learning packages. The major advantage is that any required machine learning packages can be installed on it.

Keras: Keras is a high-level open-source library used to frame different neural network architectures. It contains source codes for building different blocks of NN such as layers, optimizers, cost function, activation etc. It is user-friendly, modular, and extensible.

Numpy: Numpy is a python library that consists of high-level mathematical functions that supports large multi-dimensional arrays and matrices.

Scikit: Scikit-learn is a free open-source library, contains built-in script codes to perform various machine learning operations such as classification, regression, clustering, dimensionality reduction etc. It can interoperate with other python libraries namely NumPy, SciPy etc.

Matplotlib: The matplotlib is a powerful library in python useful for plotting graphs and visualizing any pictorial and graphical representations. The different kinds of plots, histograms, bar charts, scatterplots, etc., can be generated using this library.

Jupyter notebook: Jupyter notebook supports interactive computing and web-based applications. It supports the whole computation process of scripting, debugging codes, and execution of results. It functions as an IDE to end-users. The two main functions of jupyter notebook are:

- As a web application: The jupyter- notebook IDE is integrated with web-browser and supports interactive computing of documents which includes explanatory text, mathematical computations etc.
- As a notebook document: The end-user can interact with web applications through jupyter notebook code editor and view the inputs and outputs of the computations,

explanatory text, mathematical functions, images, graphs and rich media representations of objects.

It helps us to script, edit each piece of code and test for any bugs by integrating the code editor with any browser. The end-user can view the results of computations.

Pandas: Pandas is a python library that contains inbuilt mathematical functions, which supports different data manipulation and data analysis operations on the data frames.

LightGBM: It is a machine learning library built on the python platform. The advanced light gradient boosting library is used to solve the classification problems.

4.3 Results and Discussions

At first, the initial experiment setup that comprises our base hybrid model consisting of DAE with LightGBM is evaluated. Based on the results, later the DAE is enhanced to be the enhanced DAE models with the insertion of four proposed additional novel strategies and are associated with LightGBM and develop four enhanced hybrid IDS models to satisfy our objectives and also to estimate which one of the four strategy is very best in satisfying our objective based on the detection loss and detection rate obtained. Finally, all the proposed models are experimented and evaluated on all the eight benchmark datasets by the standard performance quality metrics and the results obtained by the proposed models are shown below

Tables 4.1a and 4.2a present the optimal hyperparameters used for our experiments.

Table:4.1a: The hyperparameter tuning of DAE for different datasets for all the proposed models

	Datasets	Hyperparameters		
		Neurons	Learning rate	Gaussian noise
1	IoTID	30	0.0002	20
2	BoT-IoT	30	Default	20
3.	CIC-IDS	30	0.0002	20
4.	CIDDS-001	8	0.0002	10
5.	CIDDS-002	8	0.0002	10
6	ISCX-URL	30	Default	20
7	ISCX-TOR	13	0.0002	15
8	UNSW-NB	25	0.0002	20

#Epoch is set to 100, activation function is set as RELU, 10% -20% of gaussian noise is used for the benchmark datasets and a single hidden layer is used respectively.

Table:4.2a: The hyperparameter tuning of LightGBM for different datasets for all the proposed models

	Datasets	Hyperparameters	
		Trees	Learning rate
1	IoTID	100	0.1
2	BoT-IoT	100	0.01
3.	CIC-IDS	150	0.5
4.	CIDDS-001	100	0.05
5.	CIDDS-002	100	0.05
6	ISCX-URL	100	0.5
7	ISCX-TOR	100	0.5
8	UNSW-NB	100	0.5

Depth of the tree is set to 25 respectively

4.3.1 Results and Discussions for the Base Hybrid Model

The datasets are pre-processed as mentioned in Section 3.1.1. Uniformly 10-20% proportion of gaussian noise is applied in the input layer for all the datasets to remove the perturbations. As this is a stochastic process, different proportion of noise insertion is experimented. The proportion of noise is varied for each dataset input features. i.e. 10% for CIDDS-001, 20% for

IoTID, 20% for BoT-IoT, 20% for CIC-IDS, 10% for CIDDS-002, 20% for ISCX-URL, 15% for ISCX-TOR and 20% for UNSW-NB respectively. The original experimental input data for the benchmark datasets are listed in Appendix section. Applying gaussian noise artificially corrupts a portion of the samples in the input layer i.e. input nodes with those corrupted portions are deactivated. This gaussian noise forces the DAE to filter the distortions and group the robust clean similar data points in each distinctive clusters of the latent space and extract the compressed hidden patterns from the high-dimensional network traffic.

The number of hidden features i.e. patterns required differ for each dataset depends on the model performance. i.e. 8 hidden features for both CIDDS datasets, 13 for ISCX-TOR2016, 25 for UNSW-NB15, 30 for CIC-IDS2017, ISCX-URL2016, BoT-IoT2018 and IoTID2020 datasets respectively. Since the extracted patterns are in encoded format, it is not easily understandable and interpretable by the humans and thus the efficiency and quality of the extracted patterns can be checked and evaluated by passing it through any machine learning classifier and measuring the performance metrics.

In the same way, the patterns from the DAE are passed to our proposed LightGBM classifier, where using those patterns, the classifier now classifies the input samples. The LightGBM instead of evaluating every single feature value to perform the classification, using its histogram binning strategy, it discretizes all feature values into histogram bins. It sorts the features values with higher to lower gradients and those feature histograms with larger gradients i.e. higher information gain are used to classify the samples, by filtering the feature values with lower gradients, that boost the predictive capacity of the model. Table 4.3 shows the results for the base hybrid model containing DAE and LightGBM classifier

Table 4.3: The classification performance of the base hybrid model

Datasets	Detection rate	Detection loss	Precision	Accuracy	F1-score
IoTID	97.43%	0.10%	97.42%	97.43%	97.42%
BoT-IoT	99.91%	0.35 %	99.92%	99.91%	99.91%
CIC-IDS	99.86%	0.15%	99.84%	99.85%	99.84%
CIDDS-001	99.60%	0.20 %	99.58%	99.60%	99.59%
CIDDS-002	99.90%	0.42 %	99.90%	99.89%	99.89%
ISCX-URL	97.76%	0.20 %	97.73%	97.72%	97.72%
ISCX-TOR	97.00%	0.11%	97.11%	96.96%	97.00%
UNSW-NB	96.11%	0.96 %	95.96%	96.00%	95.99%

On notifying the performance of all the benchmark datasets in Table 4.3. the detection loss of the model decreases and converges to a minimum level within the range between 0.10% to 0.96% over the epochs. i.e. 0.10% for IoTID, 0.35% for BoT-IoT, 0.15% for CIC-IDS, 0.20% for CIDDS-001, 0.42% for CIDDS-02, 0.20% for ISCX-URL, 0.11% for ISCX-TOR, and 0.96% for UNSW-NB respectively.

The performance graph that illustrates the detection loss of the base hybrid model for all the benchmark datasets are given below in Figure 4.1.

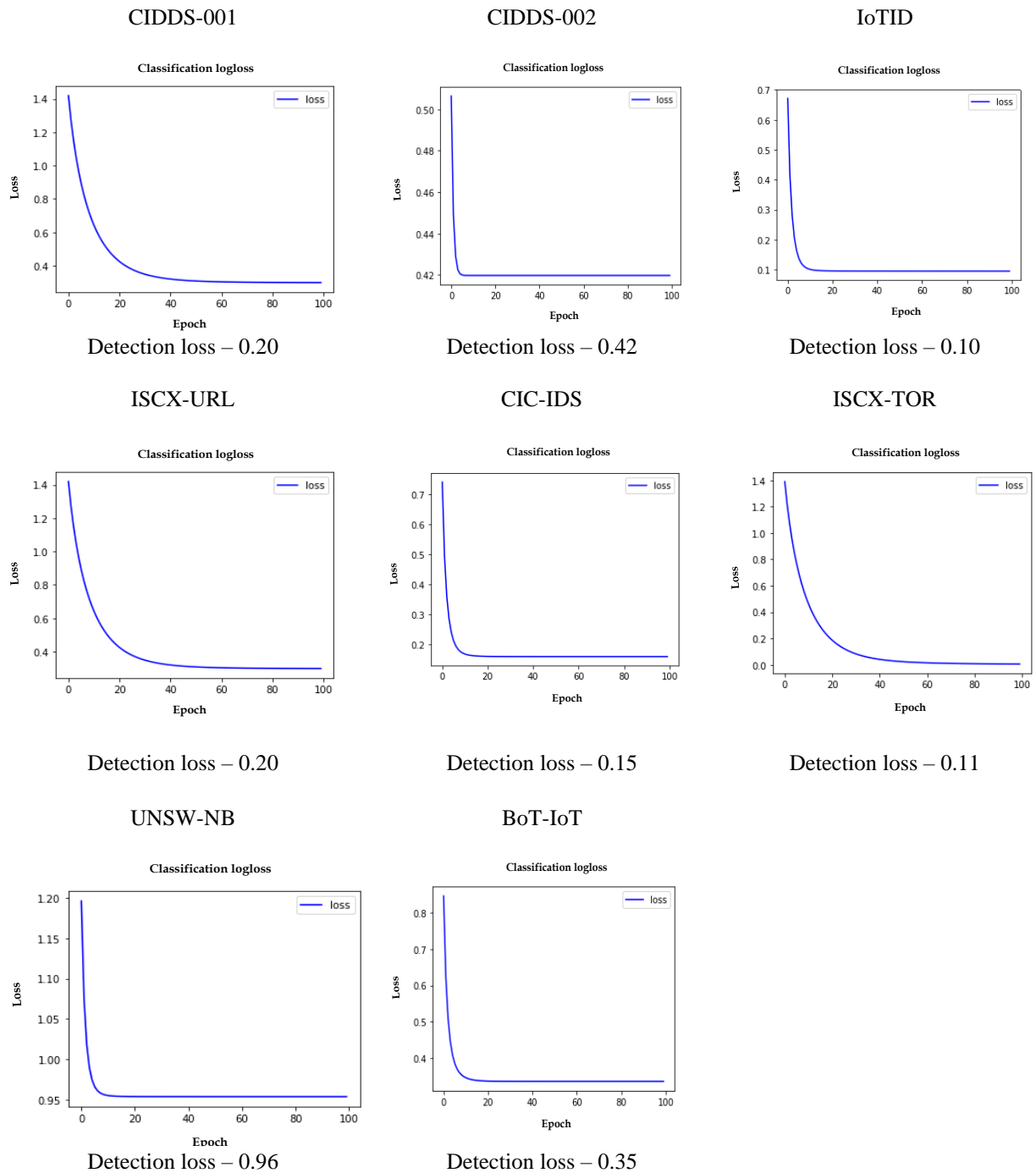


Figure 4.1: The detection loss of the base hybrid model for the benchmark datasets

The bar graph that illustrates detection loss of the base hybrid model for the benchmark datasets are shown in Figure 4.2.

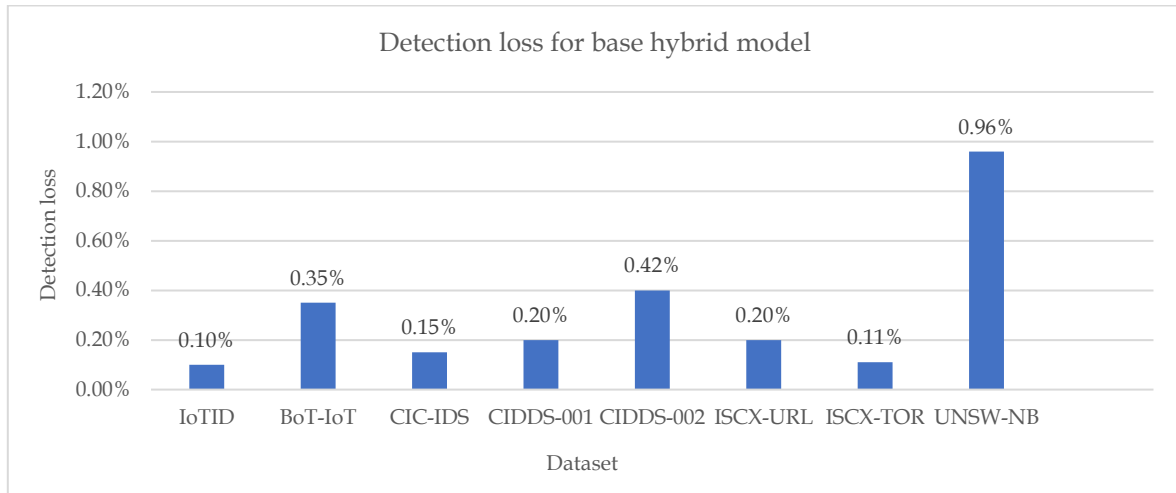


Figure 4.2: The detection loss of the base hybrid model is shown in graphical bar

On notifying the performance of all the benchmark datasets in Table 4.3. the detection rate of the model increases and converges to a maximum level within the range between 96.11% to 99.91% over the epochs. i.e. 97.43% for IoTID, 99.91% for BoT-IoT, 99.86% for CIC-IDS, 99.60% for CIDDS-001, 99.90% for CIDDS-002, 97.76% for ISCX-URL, 97% for ISCX-TOR, 96.11% for UNSW-NB.

The bar graph that illustrates detection rate of the base hybrid model for the benchmark datasets are shown in Figure 4.3.

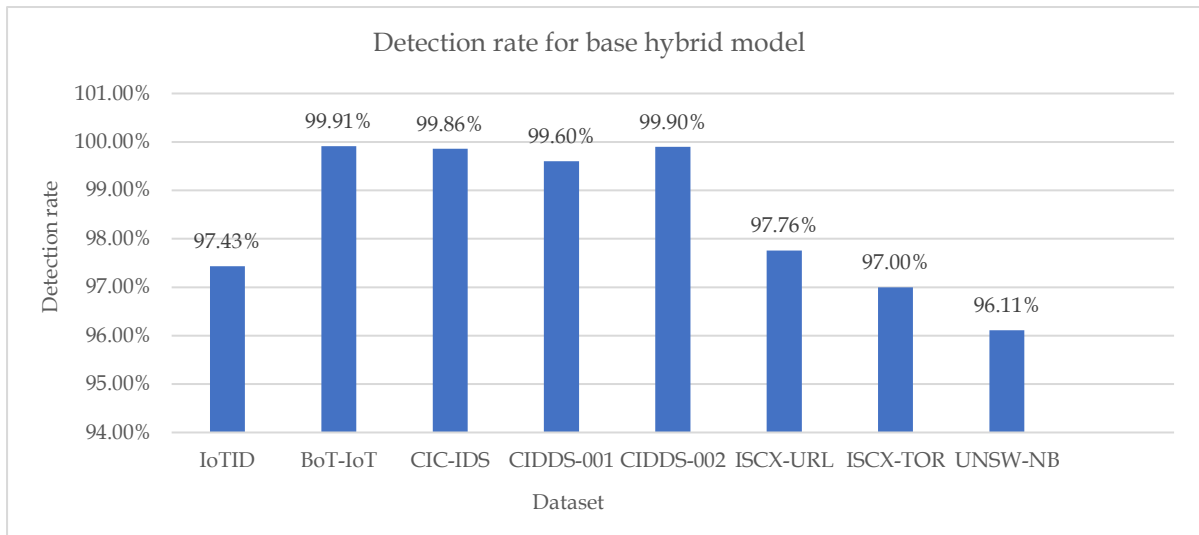


Figure 4.3: The detection rate of the base hybrid model is shown in graphical bar

It is clear the distortions are greatly reduced and the proposed model gives minimal DL with better DR. More importantly, it is to be noticed that the proposed hybrid model can detect all classes including any minority class with a smaller number of samples. i.e. “DoS”, “MITM” and “Scan” are the minority intrusive classes of the IoTID. “Reconnaissance”, and “Theft” are the minority intrusive classes of the BoT-IoT, “DoS Gold Eye”, “DoS Slow Http Test”, “DoS Slow Loris”, and “Heartbleed” are the minority intrusive classes of the CIC-IDS, “Brute Force”, “Port Scan”, and “Ping Scan” are the minority intrusive classes of the CIDDS-001, “Port Scan” is the minority intrusive class of the CIDDS-002, “Malware”, “Phishing” and “Spam” are the minority intrusive classes of the ISCX-URL, “Tor” is the minority intrusive class of the ISCX-TOR, and “Attack” is the minority intrusive class of the UNSW-NB respectively. The class-wise performance of the benchmark datasets is shown Table 4.4.

Table 4.4: The class-wise performance of the base hybrid model

Dataset	#Sample	Class	Detection rate	Precision
IoTID	59,391	DoS	99.29%	99.92%
	35,377	MITM	93.61%	94.45%
	415,677	Mirai	97.85%	97.57%
	40,073	Normal	98.00%	99.15%
	75,265	Scan	95.10%	95.04%
BoT-IoT	1,926,624	DDoS	99.91%	99.93%
	1,650,260	DoS	99.91%	99.92%
	478	Normal	92.36%	94.39%
	91,082	Reconnaissance	99.94%	99.95%
	70	Theft	91.44%	91.64%
CIC-IDS	440,031	Benign	99.88%	99.90%
	10,293	DoS Goldeneye	98.72%	99.24%
	231,073	DoS Hulk	99.91%	99.78%
	5,499	DoS Slowhttptest	98.88%	99.53%
	5,796	DoS Slowloris	98.93%	99.28%
	11	Heartbleed	96%	96.21%
CIDDS-001	225,000	Benign	99.45%	99.88%
	7,440	Brute Force	99.87%	98.14%
	146,800	DoS	99.93%	99.21%
	23,464	Port Scan	99.52%	99.71%
	6,090	Ping Scan	97.52%	98.65%
CIDDS-002	611,970	Benign	99.92%	99.91%
	481,322	Port scan	99.51%	99.64%
ISCX-URL	45,500	Defacement	97.43%	97.55%
	35,000	Benign	99.74%	99%
	11,900	Malware	96.05%	96.26%
	10,000	Phishing	92.70%	94.25%
	12,000	Spam	96.46%	97.53%
ISCX-TOR	22,552	Tor	92.64%	92.38%
	129,478	Non-Tor	97.78%	97.94%
UNSW-NB	96,385	Attack	95.75%	95.85%
	221,776	Normal	96.86%	96.19%

If DAE is not involved in the feature extraction task, the presence of deviations in the performance graph (i.e. features extraction) of one of the benchmark datasets, namely CIDDS-001 is shown below in Figure 4.4.

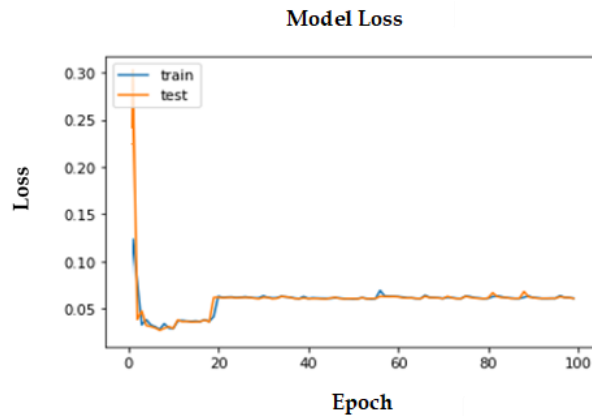


Figure 4.4: The presence of distortions in the performance graph of a benchmark dataset

It is clear from the graph; the training vs validation loss has many ups and downs i.e. spikes and ridges and has no flatness i.e. uniformity in the graph. Moreover, we observe major variations between the two. This shows the distortions are present in the network traffic and due to it, the model stuck in several underfitting/overfitting issues, and could not learn and extract the intrinsic patterns with core structure properly, which would give lower detection rate and higher detection loss.

The base hybrid model combining DAE and LightGBM gives a better DR and DL. Though DAE is a good feature extractor that reduces the perturbations as much as possible, it gives only partial robustness and regularization on the encoder side in learning and extracting patterns. On viewing the latent structure of the DAE for one of the benchmarking datasets, (for e.g.) CIDDS-001, Still there are deviations in the latent space, that lead to gaps among the similar data points which belong to identical class in each distinctive cluster labels such that the identical data points are distant apart due to the hidden unremoved distortions in the latent space, which affects the quality of the patterns extracted and so the DAE could not completely transform and group similar data points in each distinctive clusters of the latent space, which

makes the LightGBM classifier to misclassify the samples, that affects the predictive capacity of the model.

The scatter plot graph that shows the presence of hidden distortions in one of the benchmark datasets, namely CIDDSS-001, after being processed by the DAE is shown in Figure 4.5.

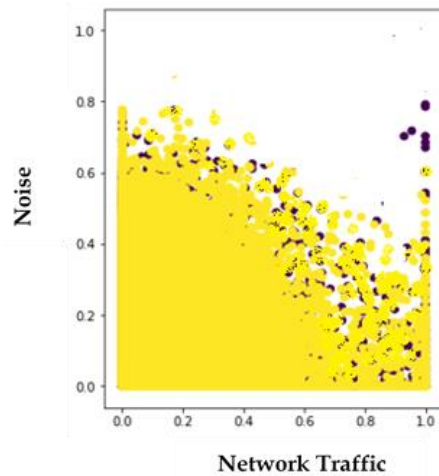


Figure 4.5. The presence of hidden distortions in a benchmark dataset

The deviations i.e. (deviated latent structure) in the latent structure of the DAE for one of the benchmark datasets, namely CIDDSS-001 are shown below in Figure 4.6.

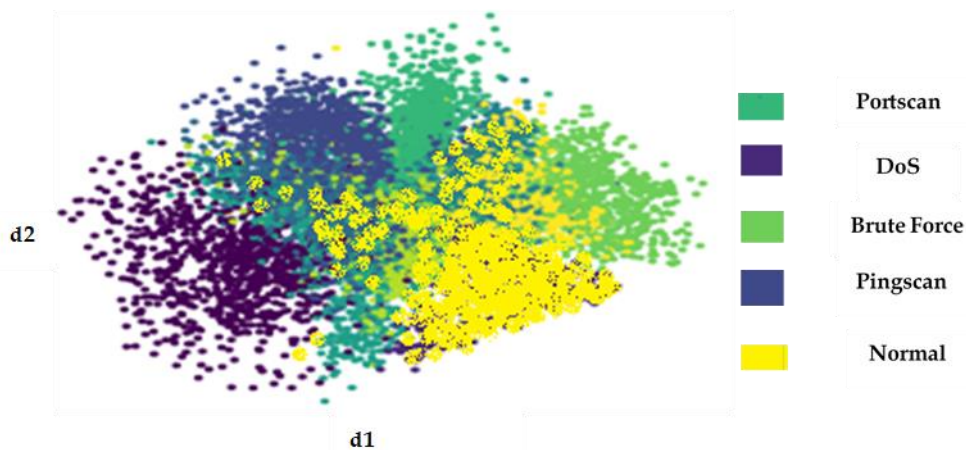


Figure 4.6: The deviated latent structure of a benchmark dataset

The smaller proportion of gaussian noise used in the input layer of the DAE gives only partial robustness and regularization on the encoder side of the DAE in extracting the hidden patterns. The higher proportion of usage of gaussian noise in the input layer of the DAE lead to corruption of larger proportion of input data samples in the input layer nodes which entirely collapse the latent structure and degrade the network performance.

In order to suppress the deviations in the latent structure and eradicate the gaps among the similar data points that are distant apart and completely project and group the similar data points that belong to identical class in each distinctive clusters of the latent space, we propose some additional novel strategies on the encoder activation of the DAE. As machine learning is a stochastic process, we test nearly four strategies and develop four enhanced DAE models i.e. Enhanced DAE 1, Enhanced DAE 2, Enhanced DAE 3, Enhanced DAE 4. Mainly, all the four strategies are simple and efficient in removing the deviations and enhancing the features learning capacity of the model. The patterns extracted from the enhanced DAE models are associated with LightGBM classifier and develop four enhanced hybrid models that can classify the datasets samples with higher prediction performance. The investigations and results of the proposed enhanced hybrid models are discussed below

4.3.2 The Results and Discussions for the Enhanced Hybrid Model 1

This model uses the jacobian gradient vector norm in the encoder layer of the DAE for the feature extraction task i.e. in removing the deviations. For discussion, let us take one of the benchmark datasets, namely CIDDS-001, which contains the real-world updated network traffic categories. The original experimental input data for the benchmark datasets are listed in Appendix section.

The jacobian matrix computes the partial derivatives of all the extracted patterns i.e. 8 latent features WRT to the original input patterns i.e. 32 input features. The extracted encoded patterns and the input patterns vary for each benchmark datasets. i.e. 28 input features and 13 extracted patterns for the ISCX-TOR, 47 input features and 25 extracted patterns for the UNSW-NB, 79 input features and 30 extracted patterns for the ISCX-URL, 12 input features and 8 extracted patterns for the both CIDDS datasets, 43 input features and 30 extracted patterns for the BoT-IoT, 43 input features and 30 extracted patterns for the IoTID, 78 input features and 30 extracted patterns for the CIC-IDS respectively. The partial derivative denotes the rate of change. The value close to 1 denotes the larger partial derivatives, and it represents the deviations in the latent manifold. The value close to 0 denotes the smaller partial derivatives and it represents the lesser or no deviations in the latent manifold.

The gradient norm with weight $10e^{-3}$ is applied on the larger partial derivatives of jacobian matrix. i.e. The proportion of gradient norm weight varies for each dataset input features. i.e. $10e^{-3}$ for CIDDS-001, CIDDS-002 and ISCX-TOR , whereas $10e^{-4}$ for IoTID, BoT-IoT, CIC-IDS, ISCX-URL and UNSW-NB datasets respectively. The gradient norm is the sum square of partial derivatives of all the extracted patterns WRT the input patterns. It enforces squishing pressure on all the larger partial derivatives i.e. values closer to 1 contract to smaller singular values closer to 0. Thus, it eradicates the deviations by reducing the gaps among the similar data points and group those data points in each distinctive cluster and grouping large number of similar data points of the identical class in each distinctive cluster contain the meaningful salient in-depth core-structure of the network traffic.

The extracted patterns from the enhanced DAE 1 are used by the LightGBM classifier to classify the datasets samples with higher DR and minimum DL. The same experimental strategy is followed and repeated for the rest of the other datasets. Table 4.5 shows the results for the enhanced hybrid model 1 containing enhanced DAE 1 and LightGBM classifier.

Table 4.5: The classification performance of the enhanced hybrid model 1

Dataset	Detection rate	Detection loss	Precision	Accuracy	F1-score
IoTID	98.58%	0.03%	98.56%	98.52%	98.57%
BoT-IoT	99.95%	0.30%	99.91%	99.93%	99.93%
CIC-IDS	99.91%	0.12%	99.89%	99.91%	99.90%
CIDDS-001	99.80%	0.12%	99.78%	99.80%	99.79%
CIDDS-002	99.96%	0.33%	99.93%	99.92%	99.94%
ISCX-URL	97.97%	0.12%	97.93%	97.95%	97.95%
ISCX-TOR	98.12%	0.04%	98.05%	98.03%	98.08%
UNSW-NB	97.36%	0.78%	97.31%	97.32%	97.33%

On notifying the performance of all the benchmark datasets in Table 4.5. the detection loss of the model decreases and converges to a minimum within the range between 0.03 to 0.78% i.e. 0.03% for IoTID, 0.30% for BoT-IoT, 0.12% for CIC-IDS, 0.12% for CIDDS-001, 0.33% for CIDDS-002, 0.12% for ISCX-URL, 0.04% for ISCX-TOR, and 0.78% for UNSW-NB respectively.

The performance graph that illustrates the detection loss of the enhanced hybrid model1 for all the benchmark datasets are given below in Figure 4.7.

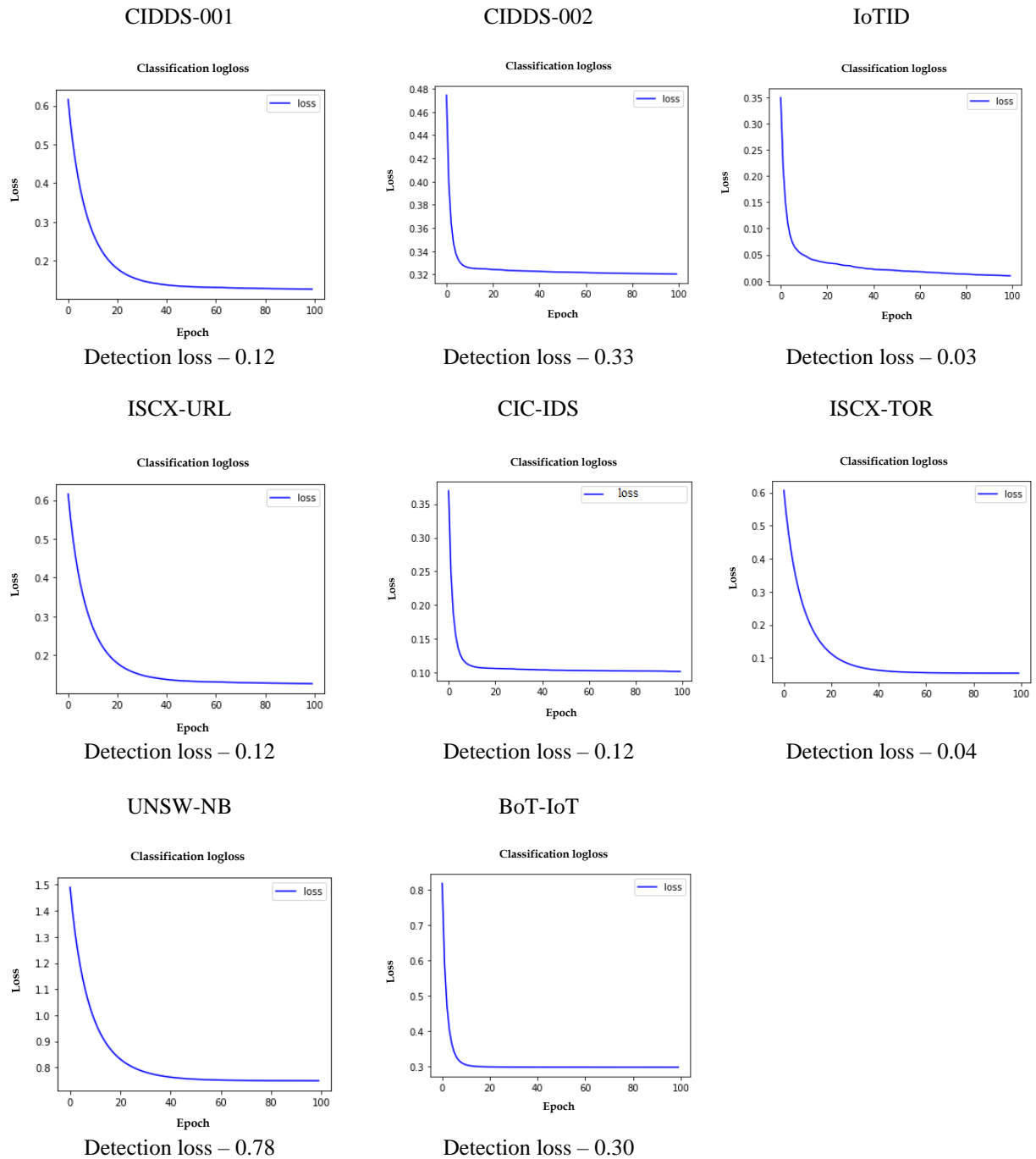


Figure 4.7: The detection loss of the enhanced hybrid model1 for all the benchmark datasets

The bar graph that illustrates detection loss of the enhanced hybrid model 1 for the benchmark datasets are shown in Figure 4.8.

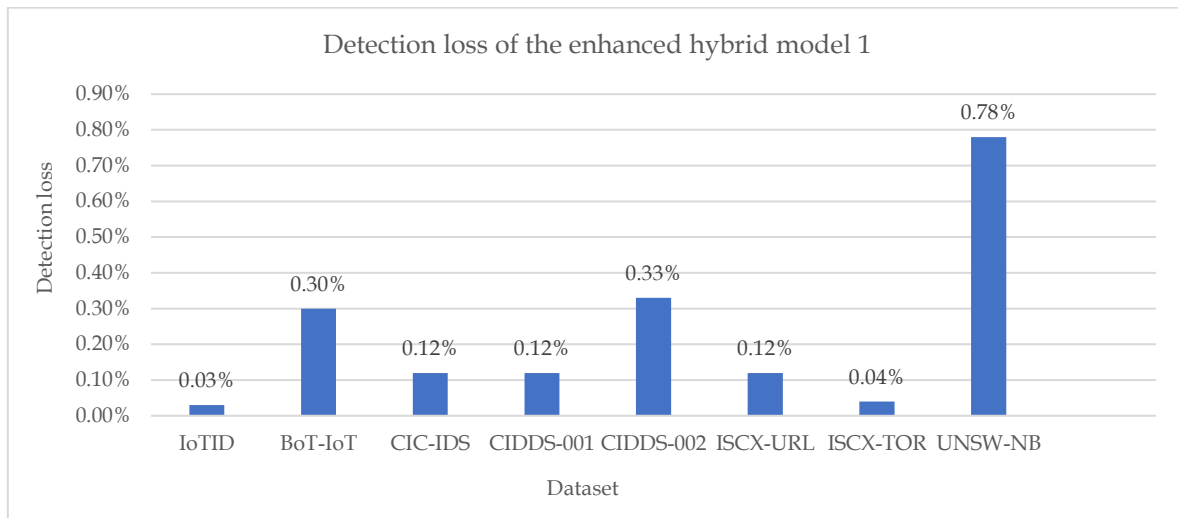


Figure 4.8: The detection loss of the enhanced hybrid model 1 is shown in graphical bar

On notifying the performance of all the benchmark datasets in Table 4.5. the detection rate of the model increases and converges to a maximum level within the range between 97.36% to 99.96% over the epochs. i.e. 98.58% for IoTID, 99.95% for BoT-IoT, 99.91% for CIC-IDS, 99.80% for CIDDS-001, 99.96% for CIDDS-002, 97.97% for ISCX-URL, 98.12% for ISCX-TOR, and 97.36% for UNSW-NB respectively. This shows that proposed enhanced hybrid model 1 outperforms the base hybrid model and other existing systems as given in Tables (4.42 – 4.49).

The bar graph that illustrates the detection rate of the enhanced hybrid model 1 for the benchmark datasets are shown in Figure 4.9.

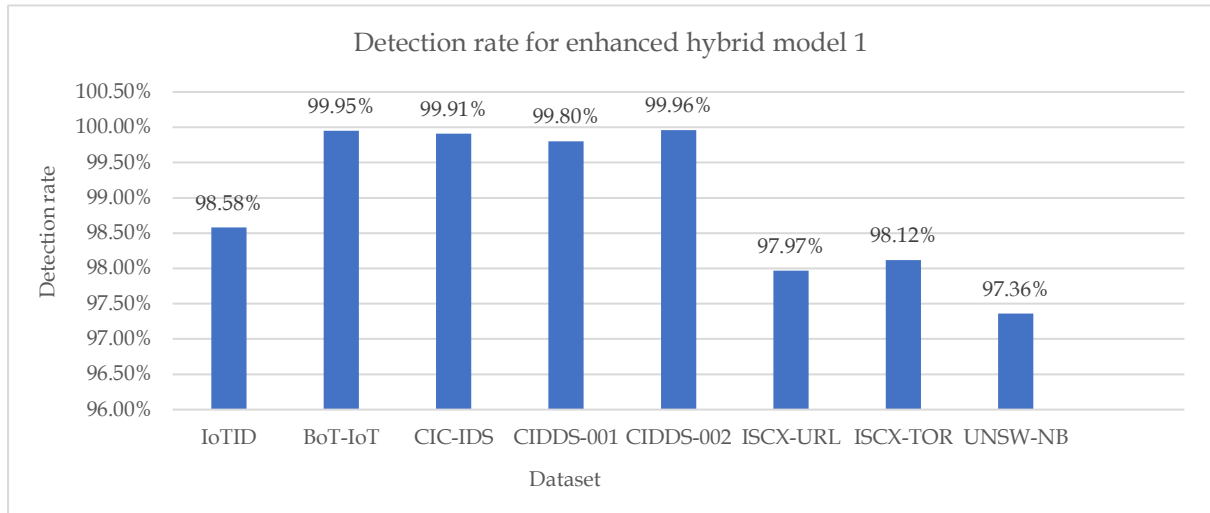


Figure 4.9: The detection rate of the enhanced hybrid model 1 is shown in graphical bar .

More importantly, on measuring the classwise performance of the benchmark datasets as shown in Table 4.6. the model scored good detection rate for the minority category samples

Table 4.6: The class-wise performance of the enhanced hybrid model 1

Dataset	#Sample	Class	Detection rate	Precision
IoTID	59,391	DoS	99.94%	99.88%
	35,377	MITM	95.30%	95.38%
	415,677	Mirai	98.87%	98.82%
	40,073	Normal	99.45%	99.35%
	75,265	Scan	97.13%	97.13%
BoT-IoT	1,926,624	DDoS	99.97%	99.92%
	1,650,260	DoS	99.93 %	99.90%
	478	Normal	95.39 %	95.39%
	91,082	Reconnaissance	99.91 %	99.91%
	70	Theft	93.33%	93.33%
CIC-IDS	440,031	Benign	99.93%	99.91%
	10,293	DoS Goldeneye	98.77%	99.51%
	231,073	DoS Hulk	99.97%	99.90%
	5,499	DoS Slowhttptest	98.94%	99.73%
	5,796	DoS Slowloris	98.98%	99%
	11	Heartbleed	96.24%	100%
CIDDS-001	225,000	Benign	99.92%	99.93%
	7,440	Brute Force	99%	99%
	146,800	DoS	99.68%	99.65%
	23,464	Port Scan	99.70%	99.70%
	6,090	Ping Scan	98.91%	98.81%
CIDDS-002	611,970	Benign	99.99%	99.95%
	481,322	Port scan	99.53%	99.32%
ISCX-URL	45,500	Defacement	98%	97.97%
	35,000	Benign	99.36%	99.26%
	11,900	Malware	97%	97%
	10,000	Phishing	95.18%	95.18%
	12,000	Spam	97.12%	97.12%
ISCX-TOR	22,552	Tor	93.66%	93.48%
	129,478	Non-Tor	98.89%	98.85%
UNSW-NB	96,385	Attack	97.12%	97.12%
	221,776	Normal	97.86%	97.71%

It is clear that the proposed strategy increases the learning and predictive capacity of the model, which shows higher performance for the minority category samples without any oversampling technique.

4.3.3 Results and Discussions for the Enhanced Hybrid Model 2

This model uses the iterative thresholding function in the encoder layer of the DAE for the feature extraction task i.e. in removing the deviations. The original experimental input data for the benchmark datasets are listed in Appendix section. The extracted encoded patterns and the input patterns vary for each benchmark datasets. i.e. 28 input features and 13 extracted patterns for the ISCX-TOR, 47 input features and 25 extracted patterns for the UNSW-NB, 79 input features and 30 extracted patterns for the ISCX-URL, 12 input features and 8 extracted patterns for the both CIDDs datasets, 43 input features and 30 extracted patterns for the BoT-IoT, 43 input features and 30 extracted patterns for the IoTID, 78 input features and 30 extracted patterns for the CIC-IDS respectively.

For discussion, let us take one of the benchmark datasets, namely CIDDs-001, which contains the real-world updated network traffic categories. Firstly, the ITF converges the activation values of all the encoder neurons i.e. 8 neurons to nearly zero i.e. it deactivates the hidden neurons. Secondly, it adaptively adjusts the threshold values of only the n - encoder neurons to nearly one. i.e. 4 neurons. The value of ' n ' varies for each dataset input features and are determined by several investigations. i.e. $n=4$ for CIDDs-001 and 002, $n=7$ for ISCX-TOR, $n=15$ for IoTID, BoT-IoT, CIC-IDS, UNSW-NB, and $n=20$ for ISCX-URL datasets respectively. The ' n ' value is further strengthened and weighted by the parameter α , that varies for each dataset. i.e. $\alpha = 2$ for datasets IoTID, BoT-IoT, CIC-IDS, ISCX-URL, ISCX-TOR and UNSW-NB, whereas $\alpha= 3$ for datasets CIDDs-001 & 002 respectively.

For CIDDs-001, instead of activating all eight encoder nodes at every iteration, only 4 encoder nodes are randomly involved at every iteration, by deactivating the rest of the nodes.

The sum of weight matrix computation is restricted only for the n -strongest encoder nodes, which enforce the weight matrix to be sparser. The sparser weight matrix consists of only the smaller magnitude elements that can regulate the activation weights. i.e. it minimizes the magnitude of the encoder weight matrix. By minimizing the values of the encoder weight matrix by activating only n -strongest neurons using the ITF, remove the deviations by grouping similar data points of the identical class in each corresponding clusters of the latent space and enhance the features learning capacity of the model.

It is noted in the support estimation set maintained by the ITF. The set consists of indexes of the n -number of strongest encoder nodes and its enforced weights in the weight matrix. On the other hand, it also contains the complement (inverse) of it. i.e. indexes of inactive nodes and its restricted weights. The information is periodically updated in the estimation set at each iteration. i.e. the encoder weight matrix is obtained by restricting the columns of w to indices of the support estimation set and thus the hidden vector values i.e. latent patterns are obtained by restricting the columns of h to the indexes of the support estimation set. Thus, it has a chance of obtaining sparse representation for each input values and achieves exact sparsity in the encoder activation at controlled level.

The proposed model eradicates the deviations and improves the structure and quality of the patterns by adjusting and penalizing the encoder weight activation at controlled sparsity level. By penalizing the activation weights of the encoder at controlled level generally suppress the deviations in the latent space of the DAE by reducing the gaps among the similar data points and group those data points in each corresponding clusters and grouping large number of similar data points of the identical class in each distinctive clusters contain the meaningful

salient in-depth core-structure of the network traffic and enhance the quality of the features patterns extracted.

The extracted patterns from the enhanced DAE 2 are used by the LightGBM classifier to classify the samples and the classifier classify the datasets samples with higher detection rate and minimal detection loss. The same experimental strategy is followed and repeated for the rest of the other datasets. Table 4.7 shows the results for the enhanced hybrid model 2 containing enhanced DAE 2 and LightGBM classifier.

Table 4.7: The classification performance of the enhanced hybrid model 2

Dataset	Detection rate	Detection loss	Precision	Accuracy	F1-score
IoTID	98.57%	0.05%	98.54%	98.52%	98.55%
BoT-IoT	99.95%	0.30%	99.91%	99.92%	99.93%
CIC-IDS	99.92%	0.11%	99.90%	99.89%	99.91%
CIDDS-001	99.79%	0.13%	99.79%	99.78%	99.79%
CIDDS-002	99.97%	0.32%	99.95%	99.92%	99.96%
ISCX-URL	97.97%	0.12%	97.95%	97.90%	97.96%
ISCX-TOR	98.12%	0.04%	98%	98.07%	98.06%
UNSW-NB	97.34%	0.74%	97.32%	97.33%	97.33%

On notifying the performance of all the benchmark datasets in Table 4.7. the detection loss of the model decreases and converges to a minimum within the range between 0.05% to 0.74%. i.e. 0.05 % for IoTID, 0.30% for BoT-IoT, 0.11% for CIC-IDS, 0.13% for CIDDS-001, 0.32% for CIDDS-002, 0.12% for ISCX-URL, 0.04% for ISCX-TOR, and 0.74 % for UNSW-NB respectively.

The performance graph that illustrates the detection loss of the enhanced hybrid model 2 for all the benchmark datasets are given below in Figure 4.10.

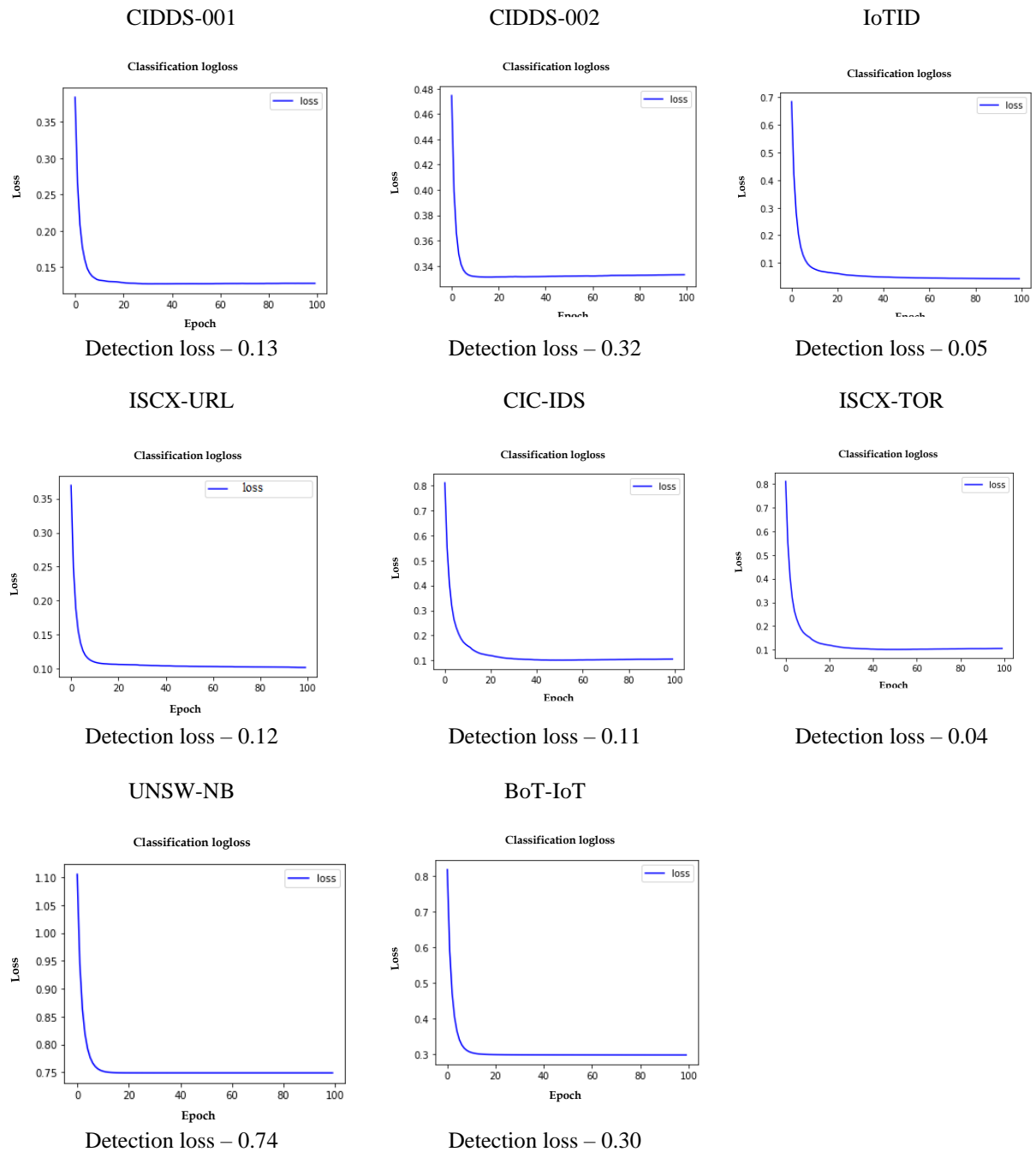


Figure 4.10: The detection loss of the enhanced hybrid model 2 for the benchmark datasets

The bar graph that illustrates detection loss of the enhanced hybrid model 2 for the benchmark datasets are shown in Figure 4.11.

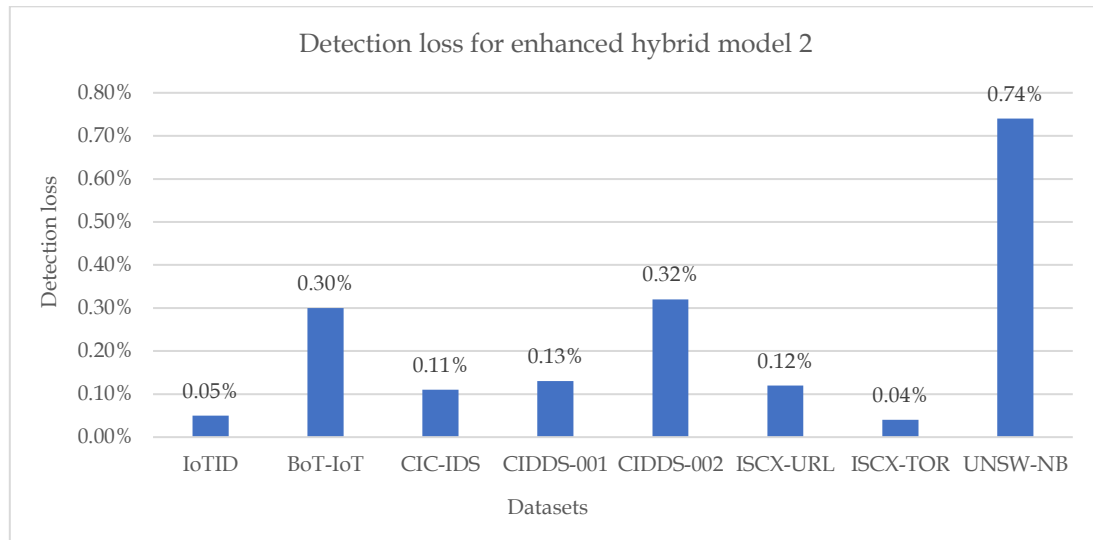


Figure 4.11: The detection loss of the enhanced hybrid model 2 is shown in graphical bar

On notifying the performance of all the benchmark datasets in Table 4.7. the detection rate of the model increases and converges to a maximum level within the range between 97.34% to 99.97% over the epochs. i.e. 98.57% for IoTID, 99.95% for BoT-IoT, 99.92% for CIC-IDS, 99.79% for CIDDS-001, 99.97% for CIDDS-002, 97.97% for ISCX-URL, 98.12% for ISCX-TOR, 97.34% for UNSW-NB datasets respectively.

The bar graph that illustrates the detection rate of the enhanced hybrid model 2 for the benchmark datasets are shown in Figure 4.12.

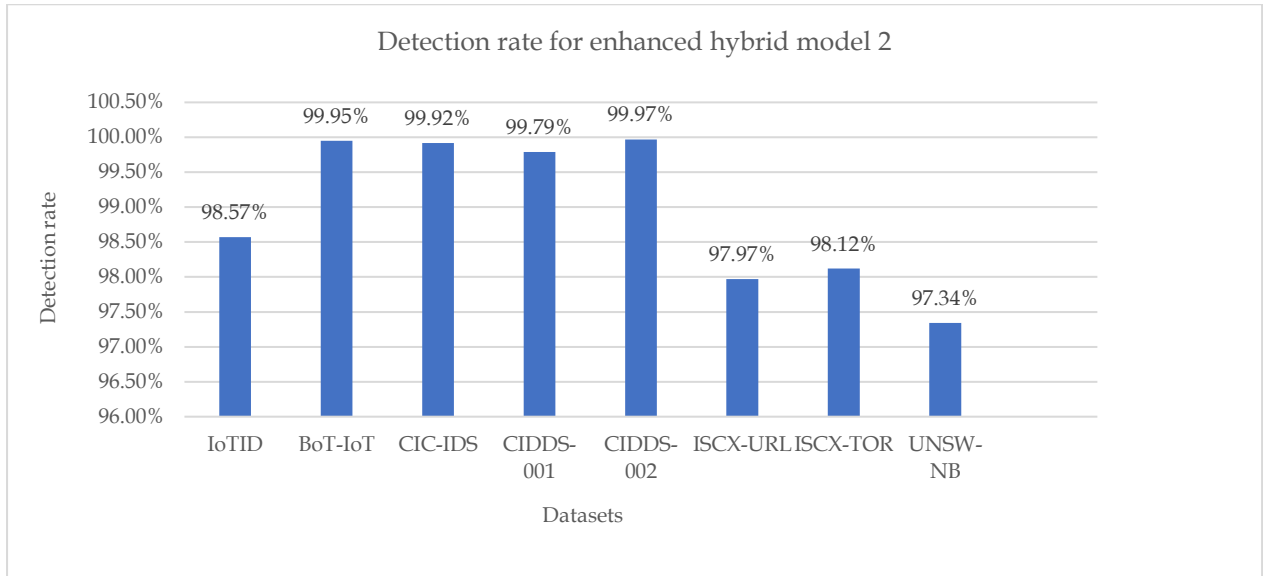


Figure 4.12: The detection rate of the enhanced hybrid model 2 are shown in graphical bar

This shows that proposed enhanced hybrid model 2 outperforms the base hybrid model and other existing systems as given in Tables (4.42 - 4.49). More importantly, on evaluating the class-wise performance of benchmark datasets as shown in Table 4.8. the model gives good detection rate for the minority category samples without any oversampling technique.

Table 4.8: The class-wise performance of the enhanced hybrid model 2

Dataset	#Sample	Class	Detection rate	Precision
IoTID	59,391	DoS	99.88%	99.80%
	35,377	MITM	95.38%	95.38%
	415,677	Mirai	98.81%	98.80%
	40,073	Normal	99.35%	99.35%
	75,265	Scan	97.13%	97.13%
BoT-IoT	1,926,624	DDoS	99.97%	99.92%
	1,650,260	DoS	99.93 %	99.90%
	478	Normal	95.39 %	95.39%
	91,082	Reconnaissance	99.91 %	99.91%
	70	Theft	93.33%	93.33%
CIC-IDS	440,031	Benign	99.92%	99.90%
	10,293	DoS Goldeneye	99.51%	99.51%
	231,073	DoS Hulk	99.95%	99.94%
	5,499	DoS Slowhttptest	99.73%	99.73%
	5,796	DoS Slowloris	99.60%	99%
	11	Heartbleed	96.35%	100%
CIDDS-001	225,000	Benign	99.93%	99.93%
	7,440	Brute Force	99%	99%
	146,800	DoS	99.68%	99.68%
	23,464	Port Scan	99.70%	99.70%
	6,090	Ping Scan	98.81%	98.81%
CIDDS-002	611,970	Benign	99.98%	99.97%
	481,322	Port scan	99.72%	99.38%
ISCX-URL	45,500	Defacement	98%	97.98%
	35,000	Benign	99.36%	99.31%
	11,900	Malware	97%	97%
	10,000	Phishing	95.18%	95.18%
	12,000	Spam	97.12%	97.12%
ISCX-TOR	22,552	Tor	93.66%	93.37%
	129,478	Non-Tor	98.89%	98.81%
UNSW-NB	96,385	Attack	97.12%	97.12%
	221,776	Normal	97.80%	97.74%

It is clear that the proposed strategy has no overfitting/underfitting issues and is not biased towards any majority class and optimizes the generalization capacity of the model, which gives higher performance for the minority category samples.

4.3.4 Results and Discussions for the Enhanced Hybrid Model 3

This model uses the data-pairwise similarity distance weight on the encoder-activation of the DAE. The original experimental input data for the benchmark datasets are listed in Appendix section. For discussion, let us take one of the benchmark datasets, namely CIDDS-001, which contains the real-world updated network traffic categories. The dataset has 32 input dimensions and 8 encoder latent dimensions with 81759 data points.

The extracted encoded patterns data with and the input patterns vary for each benchmark datasets. i.e. 28 input features and 13 extracted patterns with 35406 data points for the ISCX-TOR, 47 input features and 25 extracted patterns with 63632 data points for the UNSW-NB, 79 input features and 30 extracted patterns with 22880 data points for the ISCX-URL, 12 input features and 8 extracted patterns with 218658 data points for CIDDS-002 dataset, 43 input features and 30 extracted patterns with 733703 data points for the BoT-IoT, 43 input features and 30 extracted patterns with 125156 data points for the IoTID, 78 input features and 30 extracted patterns with 138541 data points for the CIC-IDS respectively.

By observing strong relationship among the data points, the deviation can be suppressed. We initialize the data relation by computing the data pairwise similarity distance weight among the data points in each dimension and determine a relation set for each dimension. We set similarity to be the weight of each pairwise relationship. For each dimensional data point i.e. x_i , data points with relational weight close to 1 are included in the relation set and those with relational weight close to zero are not included in the relation set and are being filtered by the alpha scalar parameter that is inserted in the encoder activation function. Both the similarity weight and the scalar parameter varies in the range between 0 to 1 in the step size of 0.02.

Consider a relation set for each dimension. For each dimensional data point, e.g. let x_i be a data point, let x_j, x_k, x_l, x_m, x_n be the set of other data points. Let us compute the data pairwise similarity weight among the data points.

- Relational similarity distance weight s_{ij} for the data point x_j is 0.82
 x_i ----- x_j (i.e. (x_i, x_j) they are similar data points)
- Relational similarity distance weight s_{ik} for the data point x_k is 0.02
 x_i ----- x_k (i.e. (x_i, x_k) they are not similar data points)
- Relational similarity distance weight s_{il} for the data point x_l is 0.71
 x_i ----- x_l i.e. (x_i, x_l) they are neighboring data points)
- Relational similarity distance weight s_{im} for the data point x_m is 0.33
 x_i ----- x_m i.e. (x_i, x_m) they are not neighboring data points).
- Relational similarity distance weight s_{in} for the data point x_n is 0.90
 x_i ---- x_n i.e. (x_i, x_n) they are neighboring data points).

As per the similarity weight principle, data points (x_i, x_l, x_n) with similarity weight close to 1 are similar i.e. most weighted data points, whereas data points (x_k, x_m) with similarity weight close to 0 are not considered as similar i.e. least weighted data points. The most weighted data points are included as neighbors and the least weighted data points are not included as neighbors in a relation set that is constructed for each dimensional cluster. Thus, the data points (x_i, x_l, x_n) are included in the relation set and rest of the data points (x_k, x_m) are not included in the relation set.

For each dimensional cluster, the set can be composed of M similar neighbors. i.e. the latent space consists of 29120 similar data points in DoS class cluster, 1527 similar data points

in Brute Force class cluster, 5652 similar data points in Port Scan class cluster, 1160 similar data points in Ping Scan class cluster, and 45130 similar data points in benign class cluster.

It minimizes the pairwise distance in the projected sub-space weighted by the corresponding distance in the original space. Minimizing the pairwise distance in the projected sub-space weighted by the corresponding distance in the original space reduces the rate of change in the latent structure by grouping similar data points in each respective clusters of the latent space.

Thus, large number of similar data points are grouped together in each corresponding clusters of the latent space which give more meaningful, useful patterns and improves the structure and quality of the features extracted by the model. Thus, the similarity weight is preserved in the latent manifold.

The extracted patterns from the enhanced DAE 3 are used by the LightGBM classifier to classify the samples with higher detection rate and minimal detection loss. The same experimental strategy is followed and repeated for the rest of the other datasets.

Table 4.9 shows the results for the enhanced hybrid model 3 containing enhanced DAE 3 and LightGBM classifier:

Table 4.9: The classification performance of the enhanced hybrid model 3

Dataset	Detection rate	Detection loss	Precision	Accuracy	F1-score
IoTID	98.58%	0.03%	98.52%	98.50%	98.55%
BoT-IoT	99.96%	0.28%	99.93%	99.91%	99.94%
CIC-IDS	99.92%	0.11%	99.90%	99.90%	99.91%
CIDDS-001	99.80%	0.12%	99.77%	99.79%	99.78%
CIDDS-002	99.97%	0.32%	99.94%	99.93%	99.95%
ISCX-URL	97.99%	0.10%	97.93%	97.90%	97.96%
ISCX-TOR	98.11%	0.05%	98.10%	98.07%	98.10%
UNSW-NB	97.35%	0.75%	97.33%	97.31%	97.34%

More importantly, on evaluating the classification performance of the proposed enhanced hybrid model 3 as shown in Table 4.9. the detection loss of the model decreases and converges to a very minimum within the range between 0.03% to 0.75% i.e. 0.03 % for IoTID, 0.28% for BoT-IoT, 0.11% for CIC-IDS, 0.12% for CIDDS-001, 0.32% for CIDDS-002, 0.10% for ISCX-URL, 0.05% for ISCX-TOR, and 0.75% for UNSW-NB datasets respectively.

The performance graph that illustrates the detection loss of the enhanced hybrid model 3 for all the benchmark datasets are given below in Figure 4.13.

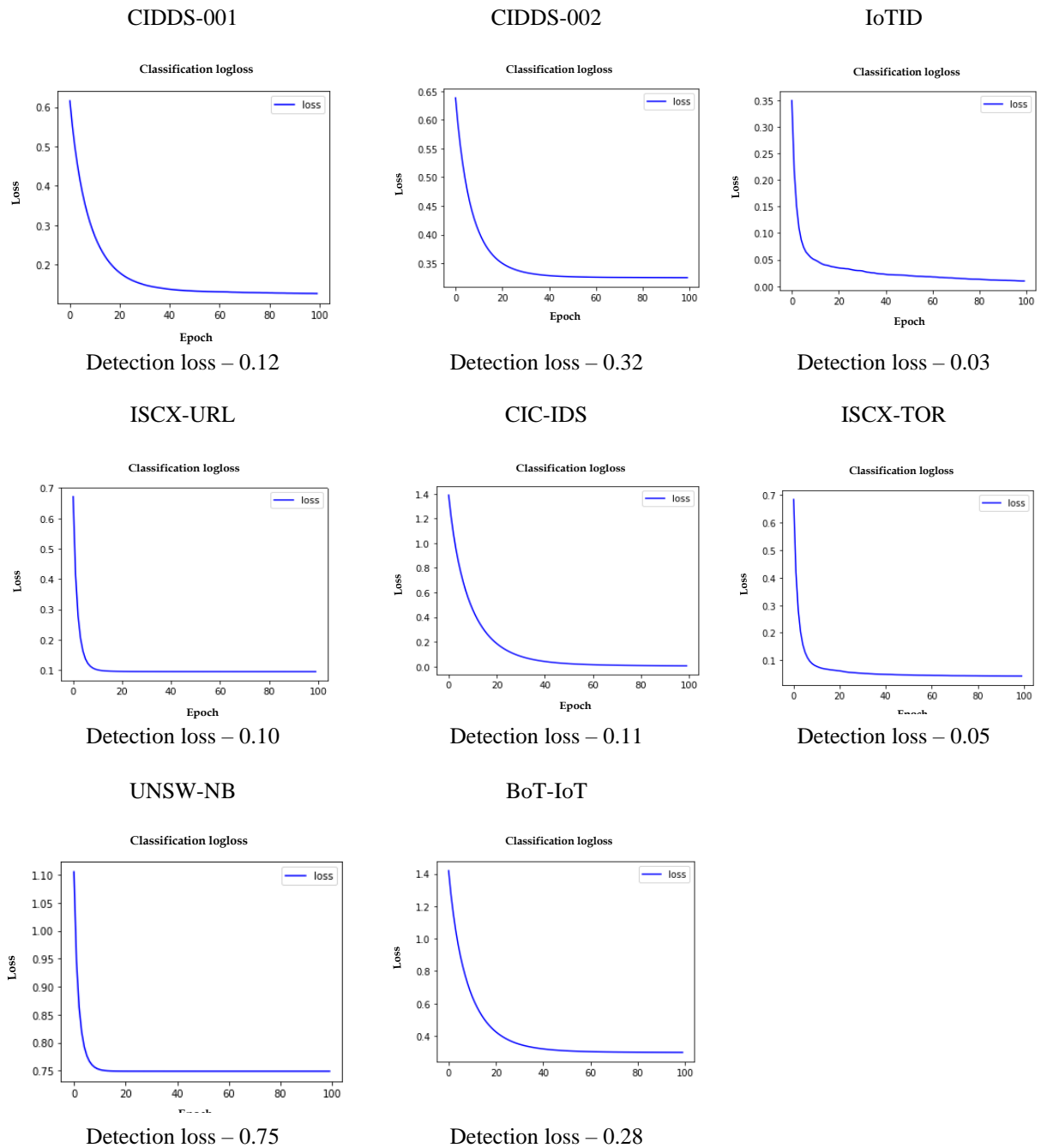


Figure 4.13: The detection loss of the enhanced hybrid model 3 for the benchmark datasets

The bar graph that illustrates detection loss of the enhanced hybrid model 3 for the benchmark datasets are shown in Figure 4.14.

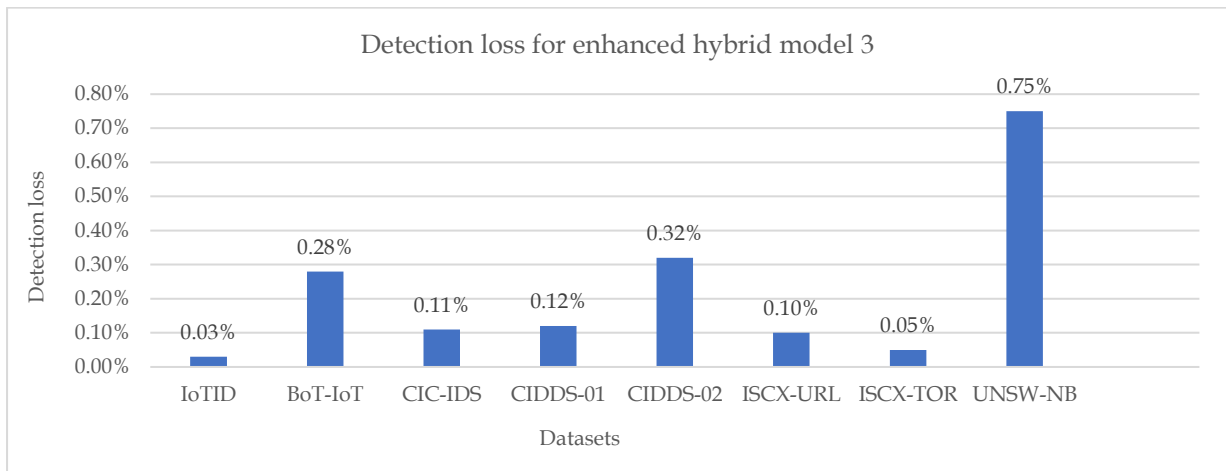


Figure 4.14: The detection loss of the enhanced hybrid model 3 are shown in graphical bar

More importantly, on evaluating the classification performance of the proposed enhanced hybrid model 3 as shown in Table 4.9. the detection rate of the model increases and converges to a maximum level within the range between 97.35% to 99.97% over the epochs. i.e. 98.58% for IoTID, 99.96% for BoT-IoT, 99.92% for CIC-IDS, 99.80% for CIDDS-001, 99.97% for CIDDS-002, 97.99% for ISCX-URL, 98.11% for ISCX-TOR, 97.35% for UNSW-NB, respectively.

The bar graph that illustrates detection rate of the enhanced hybrid model 3 for the benchmark datasets are shown in Figure 4.15.

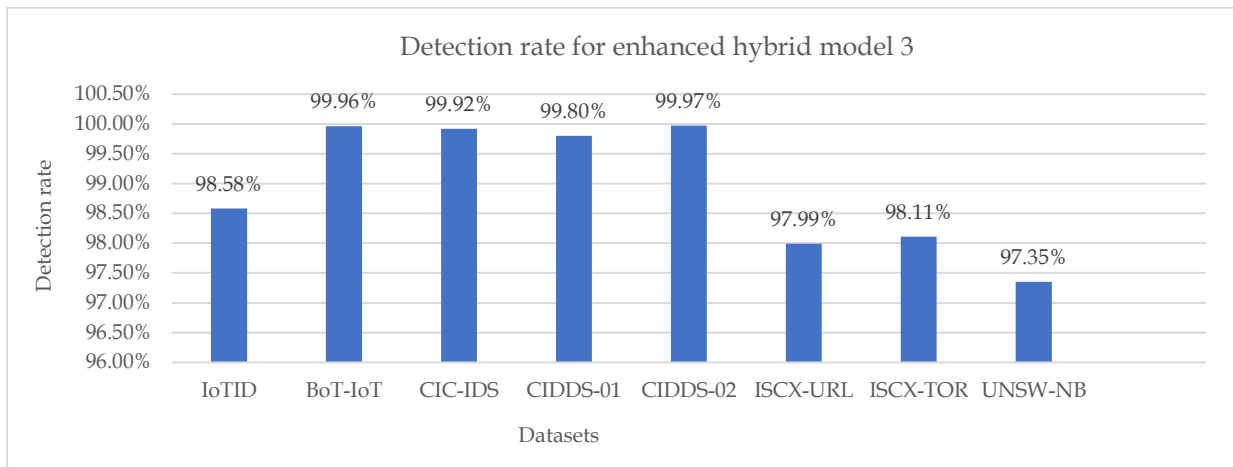


Figure 4.15: The detection rate of the enhanced hybrid model 3 are shown in graphical bar

This shows that proposed enhanced hybrid model 3 outperforms the base hybrid model and other existing systems as given in Tables (4.42 - 4.49). More importantly, on evaluating the class-wise performance of the benchmark datasets as shown in Table 4.10. the model gives good detection rate for the minority category without any oversampling technique.

Table 4.10: The class-wise performance of the enhanced hybrid model 3

Dataset	#Sample	Class	Detection Rate	Precision
IoTID	59,391	DoS	99.94%	99.78%
	35,377	MITM	95.30%	95.38%
	415,677	Mirai	98.86%	98.79%
	40,073	Normal	99.45%	99.35%
	75,265	Scan	97%	97.04%
BoT-IoT	1,926,624	DDoS	99.97%	99.94%
	1,650,260	DoS	99.95%	99.93%
	478	Normal	95.47%	95.39%
	91,082	Reconnaissance	99.93%	99.91%
	70	Theft	93.33%	93.33%
CIC-IDS	440,031	Benign	99.94 %	99.92%
	10,293	DoS Goldeneye	99.58%	99.51%
	231,073	DoS Hulk	99.97 %	99.94%
	5,499	DoS Slowhttpstest	99.72%	99.72%
	5,796	DoS Slowloris	99.64%	99.56%
	11	Heartbleed	96.35%	96.35%
CIDDS-001	225,000	Benign	99.92%	99.93%
	7,440	Brute Force	99%	99%
	146,800	DoS	99.68%	99.65%
	23,464	Port Scan	99.90%	99.70%
	6,090	Ping Scan	98.91%	98.81%
CIDDS-002	611,970	Benign	99.98%	99.96%
	481,322	Port scan	99.72%	99.35%
ISCX-URL	45,500	Defacement	98%	97.97%
	35,000	Benign	99.36%	99.26%
	11,900	Malware	97%	97%
	10,000	Phishing	95.4%	95.18%
	12,000	Spam	97.12%	97.12%
ISCX-TOR	22,552	Tor	93.59%	93.52%
	129,478	Non-Tor	98.90%	98.90%
UNSW-NB	96,385	Attack	97.11%	97.12%
	221,776	Normal	97.83%	97.77%

It is clear that the proposed strategy optimizes the generalization capacity of our model and is free from any overfitting/underfitting issues, which gives higher performance for the minority category samples.

4.3.5 Results and Discussions for the Enhanced DAE Model 4

This model uses the inference based approximated standard normal distribution on the encoder activation of the DAE. The original experimental input data for the benchmark datasets are listed in Appendix section. For the discussion, let us take one of the benchmark datasets CIDDS-001, which contains the real-world updated network traffic categories. The dataset has 32 original input patterns and we extract 8 hidden patterns from it.

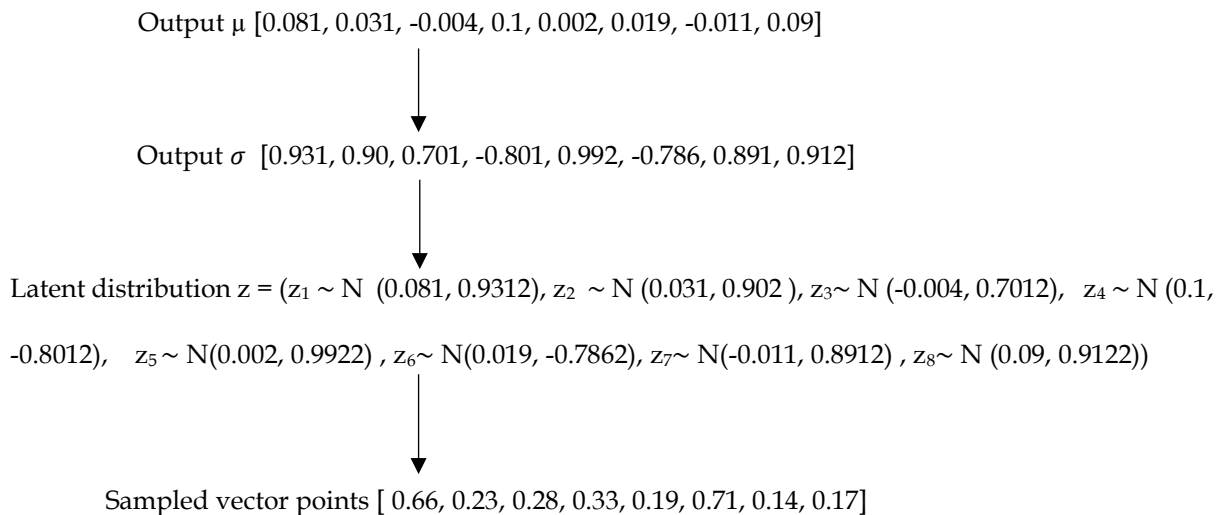
The extracted encoded patterns and the input patterns vary for each benchmark datasets. i.e. 28 input features and 13 extracted patterns for the ISCX-TOR, 47 input features and 25 extracted patterns for the UNSW-NB, 79 input features and 30 extracted patterns for the ISCX-URL, 12 input features and 8 extracted patterns for the both CIDDS datasets, 43 input features and 30 extracted patterns for the BoT-IoT, 43 input features and 30 extracted patterns for the IoTID, 78 input features and 30 extracted patterns for the CIC-IDS respectively.

We enforce the distribution of each input feature dimension to be the standard normal distribution i.e. zero mean and unit variance. The values are standardized by taking the z-score of the normal distribution values. It is set as the prior distribution.

The enhanced DAE 4, for each dimension, instead of computing a single real vector value for each dimension, it outputs two vector values 1. mean 2. variance that are close to the standard normal distribution i.e. mean close to zero, variance close to one and it gives an entire probability distribution for each and every features dimension of the network traffic, i.e. it makes the latent space continuous and complete in nature, by reducing the gaps among the similar data points that are distant apart and grouping the similar data points that belong to

identical class in each various clusters of the latent space, giving the salient abstract core-structure lying in the high-dimensional network traffic, which eradicates the deviations in the latent space and enhances the quality of the patterns extracted. It is the generated posterior distribution, from which the data points are generated and sampled, that gives the meaningful, salient and useful structure of the network traffic.

In order to ensure the generated distribution to be an approximation/maximum likelihood of the original prior distribution, we can use the inference strategy. In this investigation 8 encoder latent dimensions with 413794 sampled data points are generated and extracted. The encoded latent representation values are given below



All the encoded generated dimensions i.e. 8 gives the entire probability distribution of the input network traffic, i.e. 8 latent dimensions give the approximated standard normal continuous probability distribution, which give continuous and complete structure by removing the deviations and grouping large number of similar data points of identical class in each distinctive cluster labels and enhancing the quality and structure of the features extracted.

The extracted patterns from the enhanced DAE 4 are used by the LightGBM classifier to classify the samples and the classifier classify the datasets samples with highest detection rate and lowest detection loss. The same experimental strategy is followed and repeated for the rest of the other datasets.

Table 4.11. shows the results for the enhanced hybrid model 4 containing enhanced DAE 4 and LightGBM classifier:

Table 4.11: The classification performance of the enhanced hybrid model 4

Dataset	Detection rate	Detection loss	Precision	Accuracy	F1-score
IoTID	98.61%	0.01%	98.58%	98.57%	98.59%
BoT-IoT	99.98%	0.20%	99.95%	99.96%	99.96%
CIC-IDS	99.94%	0.09%	99.91%	99.92%	99.92%
CIDDS-001	99.85%	0.07%	99.80%	99.85%	99.82%
CIDDS-002	99.98%	0.30%	99.97%	99.96%	99.97%
ISCX-URL	98.06%	0.06%	98%	98.03%	98.04%
ISCX-TOR	98.14%	0.02%	98.12%	98.12%	98.13%
UNSW-NB	97.39%	0.70%	97.36%	97.35%	97.34%

On notifying the performance of all the benchmark datasets in Table 4.11. the detection loss of the model decreases and converges to a very minimum within the range between 0.01% to 0.70%. i.e. 0.01% for IoTID, 0.20% for BoT-IoT, 0.09% for CIC-IDS, 0.07% for CIDDS-001, 0.30% for CIDDS-002, 0.06% for ISCX-URL, 0.02% for ISCX-TOR and 0.70% for UNSW-NB respectively.

The performance graph that illustrates the detection loss of the enhanced hybrid model 4 for all the benchmark datasets are given below in Figure 4.16.

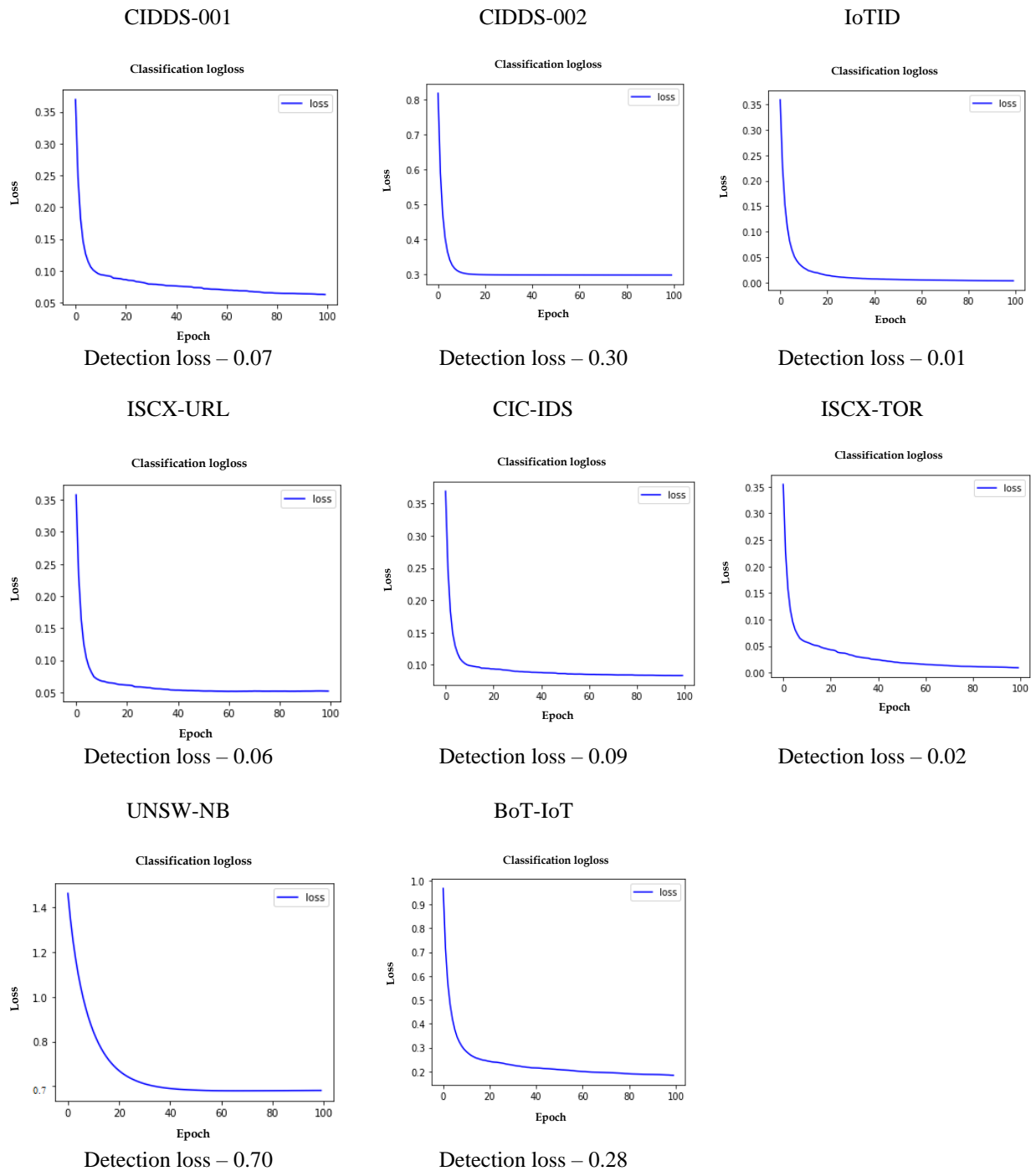


Figure 4.16: The detection loss of the enhanced hybrid model 4 for all the benchmark datasets

The bar graph that illustrates detection loss of the enhanced hybrid model 4 for the benchmark datasets are shown in Figure 4.17.

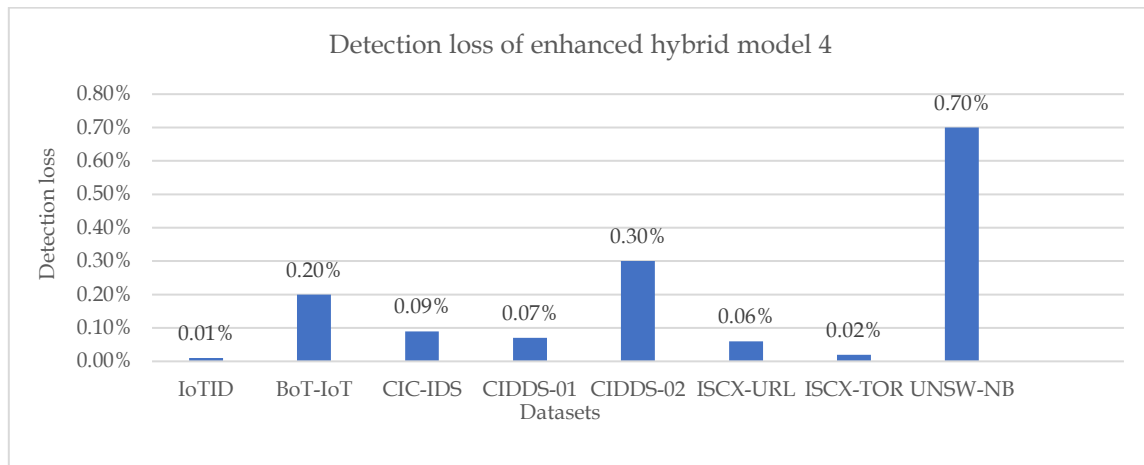


Figure 4.17: The detection loss of the enhanced hybrid model 4 is shown in graphical bar

On notifying the performance of all the benchmark datasets in Table 4.11. the detection rate of the model increases and converges to a very maximum level within the range between 97.39% to 99.98% over the epochs. i.e. 98.61% for IoTID, 99.98 % for BoT-IoT, 99.94% for CIC-IDS, 99.85% for CIDDS-001, 99.98% for CIDDS-002, 98.06% for ISCX-URL, 98.14% for ISCX-TOR, 97.39% for UNSW-NB respectively.

The bar graph that illustrates detection rate of the enhanced hybrid model 4 for the benchmark datasets are shown in Figure 4.18.

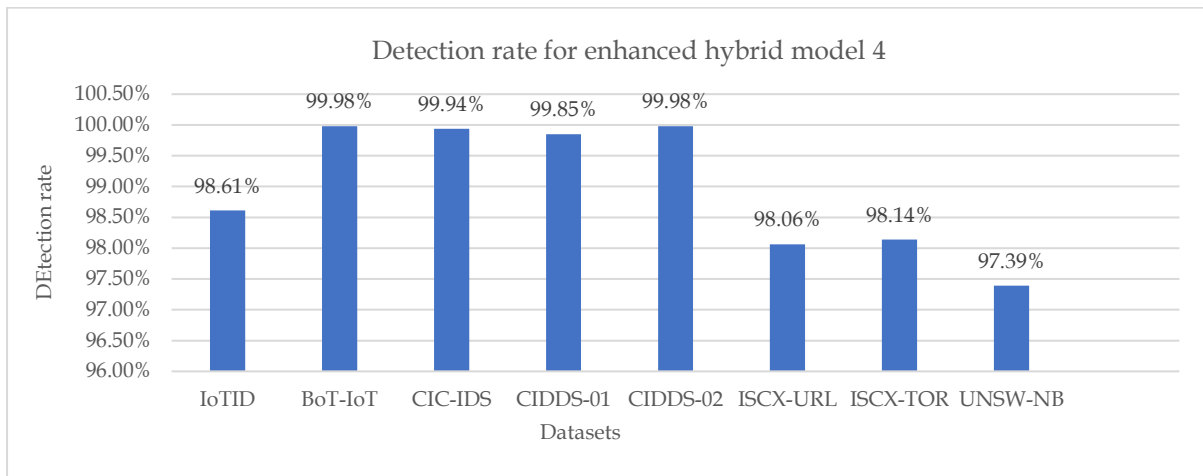


Figure 4.18: The detection rate of the enhanced hybrid model 4 is shown in graphical bar

This shows that proposed enhanced hybrid model 4 outperforms the other three enhanced hybrid models with the base hybrid model and other existing systems as given in Tables (4.42-4.49). More importantly, on evaluating the class-wise performance of the benchmark datasets as shown in Table 4.12. the model gives good detection rate for the minority category without any oversampling technique.

Table 4.12: The class-wise performance of the enhanced hybrid model 4

Dataset	#Sample	Class	Detection rate	Precision
IoTID	59,391	DoS	99.94%	99.90%
	35,377	MITM	95.46%	95.38%
	415,677	Mirai	98.87%	98.83%
	40,073	Normal	99.45%	99.45%
	75,265	Scan	97.13%	97.06%
BoT-IoT	1,926,624	DDoS	99.99%	99.97%
	1,650,260	DoS	99.97%	99.93%
	478	Normal	95.47%	95.47%
	91,082	Reconnaissance	99.98%	99.96%
	70	Theft	93.33%	93.33%
CIC-IDS	440,031	Benign	99.94%	99.92%
	10,293	DoS Goldeneye	99.52%	99.51%
	231,073	DoS Hulk	99.97%	99.94%
	5,499	DoS Slowhttptest	99.73%	99.73%
	5,796	DoS Slowloris	99.64%	99%
	11	Heartbleed	96.35%	96.35%
CIDDS-001	225,000	Benign	99.93%	99.92%
	7,440	Brute Force	99.14%	99%
	146,800	DoS	99.78%	99.68%
	23,464	Port Scan	99.90%	99.87%
	6,090	Ping Scan	98.95%	98.91%
CIDDS-002	611,970	Benign	99.98%	99.98%
	481,322	Port scan	99.90%	99.72%
ISCX-URL	45,500	Defacement	98%	98%
	35,000	Benign	99.84%	99.36%
	11,900	Malware	96.65%	97.14%
	10,000	Phishing	95.36%	95.36%
	12,000	Spam	97.70%	97.66%
ISCX-TOR	22,552	Tor	93.79%	93.66%
	129,478	Non-Tor	98.90%	98.89%
UNSW-NB	96,385	Attack	97.12%	97.12%
	221,776	Normal	97.96%	97.86%

It is clear that the proposed strategy has no overfitting/underfitting issues and is not biased towards any majority class. It has higher learning and predictive capacity that accelerates the generalization capacity of the model, which gives highest performance for the minority category samples.

Figure 4.19 represents the removed distortions in a benchmark dataset are shown in scatter plot graph

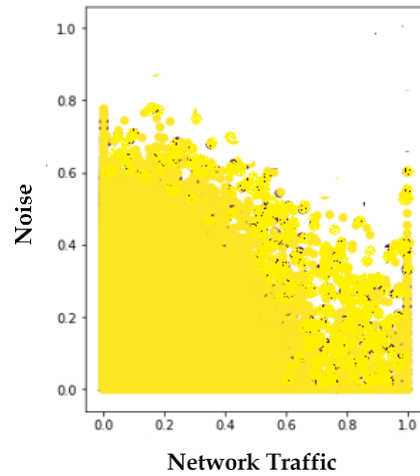


Figure 4.19: The removed distortions in a benchmark dataset

Figure 4.19 denotes the distortions are removed from a benchmark dataset CIDD5-001, i.e. the dataset has less distortions or no distortions.

The undeviated clustered latent structure of one of the benchmark datasets, is shown below in Figure 4.20.

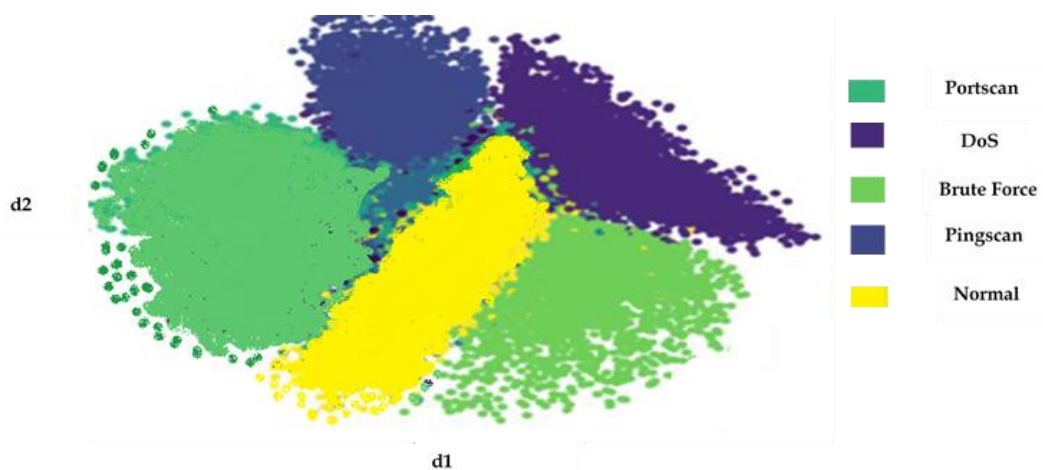


Figure 4.20: The undeviated latent structure by the Enhanced DAE

From the Figure 4.20, it is noticed that the deviations are removed and the similar data points are grouped in their corresponding class clusters of the latent space, which give a meaningful and descriptive structure of the network traffic.

4.3.6 The evaluation and interpretation of results of the benchmark datasets

The results obtained by the proposed models for the benchmark datasets are interpreted as below, where Base denotes the base hybrid model, M1 denotes the enhanced hybrid model 1, M2 denotes the enhanced hybrid model 2, M3 denotes the enhanced hybrid model 3, M4 denotes the enhanced hybrid model 4.

IoTID-2020

The confusion-matrix of IoTID for each class is shown below in Tables (4.13 – 4.17):

The confusion-matrix for the Mirai class is shown in Table 4.13.

Table 4.13: The confusion matrix for the Mirai class

Predicted values	Actual values												
		Mirai (P)						Other classes (N)					
		Base	M1	M2	M3	M4	Base	M1	M2	M3	M4		
Mirai (P)	(TP)	81055	82021	81967	82021	82021	(FN)	1899	933	987	933	933	
Other classes (N)	(FP)	2018	1001	993	978	962	(TN)	40185	41202	41210	41225	41241	

$$\text{Detection rate} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{Detection rate}$$

$$\text{Base} = 81055 / (81055 + 1899) = 81055 / 82954$$

$$\text{M1} = 82021 / (82021 + 933) = 82021 / 82954$$

$$\text{M2} = 81967 / (81967 + 987) = 81967 / 82954$$

$$\text{M3} = 82021 / (82021 + 933) = 82021 / 82954$$

$$\text{M4} = 82021 / (82021 + 933) = 82021 / 82954$$

As for the Mirai class is concerned, Among the 82954 total Mirai class samples in a test set, 81055 samples are correctly detected as Mirai by the base hybrid model, 82021 samples are

correctly detected by the enhanced hybrid models M1, M3 and M4 , whereas 81967 samples are correctly detected by the enhanced hybrid model M2. The detection rate of all the enhanced hybrid models is higher than our base hybrid model.

The confusion-matrix for the DoS class is shown in Table 4.14

Table 4.14: The confusion-matrix for the DoS class

Predicted values		Actual values										
		DoS (P)						Other classes (N)				
			Base	M1	M2	M3	M4		Base	M1	M2	M3
DoS (P)	(TP)	12029	12072	12065	12072	12077	(FN)	50	7	14	7	2
Other classes (N)	(FP)	9	14	24	26	7	(TN)	113069	113050	113040	113038	113064

Detection rate

$$\text{Base} = 12029 / 12029 + 50 = 12029 / 12079$$

$$\text{M1} = 12072 / 12072 + 7 = 12072 / 12079$$

$$\text{M2} = 12065 / 12065 + 14 = 12065 / 12079$$

$$\text{M3} = 12072 / 12072 + 7 = 12072 / 12079$$

$$\text{M5} = 12077 / 12077 + 72 = 12077 / 12079$$

As for the DoS class is concerned, Among the 12079 total DoS class samples in a test set, 12029 samples are correctly detected as DoS by the base hybrid model ,12065 samples are correctly detected by the enhanced hybrid model M2, 12072 samples are correctly detected by the enhanced hybrid models M1 and M3, whereas 12077 samples are correctly detected by the enhanced model M4. The detection rate of all the enhanced hybrid models is higher than our base hybrid model.

The confusion-matrix for the Scan class is shown in Table 4.15.

Table 4.15: The confusion-matrix for the Scan class

Predicted values		Actual values										
		Scan (P)						Other classes (N)				
			Base	M1	M2	M3	M4		Base	M1	M2	M3
Scan (P)	(TP)	14408	14710	14710	14696	14731	(FN)	736	434	434	448	413
Other classes (N)	(FP)	753	434	435	448	435	(TN)	109210	109579	109578	109565	109578

Detection rate

$$\text{Base} = 14408 / (14408 + 736) = 14408 / 15144$$

$$\text{M1} = 14710 / (14710 + 434) = 14710 / 15144$$

$$\text{M2} = 14710 / (14710 + 434) = 14710 / 15144$$

$$\text{M3} = 14696 / (14696 + 448) = 14696 / 15144$$

$$\text{M4} = 14731 / (14731 + 413) = 14731 / 15144$$

As for the Scan class is concerned, Among the 15144 total Scan class samples in a test set, 14408 samples are correctly detected as Scan by the base hybrid model, 14710 samples are correctly detected by the enhanced hybrid models M1 and M2, 14696 samples are correctly detected by the enhanced hybrid model M3, whereas 14731 samples are correctly detected by the enhanced hybrid model M4. The detection rate of the enhanced hybrid models is higher than our proposed base hybrid model.

The confusion-matrix for the MITM ARP Spoofing class is shown in Table 4.16.

Table 4.16: The confusion-matrix for the MITM ARP Spoofing class

Predicted values	Actual values											
		MITM (P)					Other classes (N)					
		Base	M1	M2	M3	M4		Base	M1	M2	M3	M4
MITM (P)	(TP)	6576	6640	6645	6640	6651	(FN)	391	327	322	327	316
Other classes (N)	(FP)	386	321	322	328	321	(TN)	117804	117869	117868	117862	117869

Detection rate

$$\text{Base} = 6576 / (6576 + 391) = 6576 / 6967$$

$$\text{M1} = 6640 / (6640 + 327) = 6640 / 6967$$

$$\text{M2} = 6645 / (6645 + 322) = 6645 / 6967$$

$$\text{M3} = 6640 / (6640 + 327) = 6640 / 6967$$

$$\text{M4} = 6651 / (6651 + 316) = 6651 / 6967$$

As for the MITM class is concerned, Among the 6967 total MITM class samples in a test set, 6576 samples are correctly detected as MITM by the base hybrid model, 6640 samples are correctly detected by the enhanced hybrid models M1 and M3, 6645 samples are correctly detected by the subsequent enhanced hybrid model M2, whereas 6651 samples are correctly

detected by the enhanced hybrid model M4. The detection rate of all the enhanced hybrid models is higher than our base hybrid model.

The confusion-matrix for the Normal class is shown in Table 4.17.

Table 4.17: The confusion-matrix for the Normal class

Predicted values	Actual values									
	Normal					Other classes				
	Base	M1	M2	M3	M4	Base	M1	M2	M3	M4
Normal	7881	7969	7961	7969	7969	132	44	52	44	44
Other classes	67	52	51	52	45	117077	117144	117093	117144	117099

Detection Rate

$$\text{Base} = 7881 / (7881 + 132) = 7881 / 8013$$

$$\text{M1} = 7969 / (7969 + 44) = 7969 / 8013$$

$$\text{M2} = 7961 / (7961 + 52) = 7961 / 8013$$

$$\text{M3} = 7969 / (7969 + 44) = 7969 / 8013$$

$$\text{M4} = 7969 / (7969 + 44) = 7969 / 8013$$

As for the Normal class is concerned, Among the 8013 total Normal class samples in a test set, 7881 samples are correctly detected as benign by the base hybrid model M1, 7969 samples are correctly detected by the enhanced hybrid models M1, M3 and M4, whereas 7961 samples are correctly detected by the enhanced model M2. The detection rate of all the enhanced hybrid models is higher than our base hybrid model.

Totally, out of 117144 intrusive samples in IoTID, 115480 samples are correctly detected by the enhanced hybrid model 4, 115429 samples are correctly detected by the enhanced hybrid models 1 and 3, 115387 samples are correctly detected by the enhanced hybrid model 2, 114068 samples are correctly detected by the base hybrid model. The enhanced hybrid models outperform the base hybrid model. The enhanced hybrid model 4 outperforms the other three enhanced models.

The bar graph that illustrates the classification performance of the proposed models for IoTID is shown in Figure 4.21.

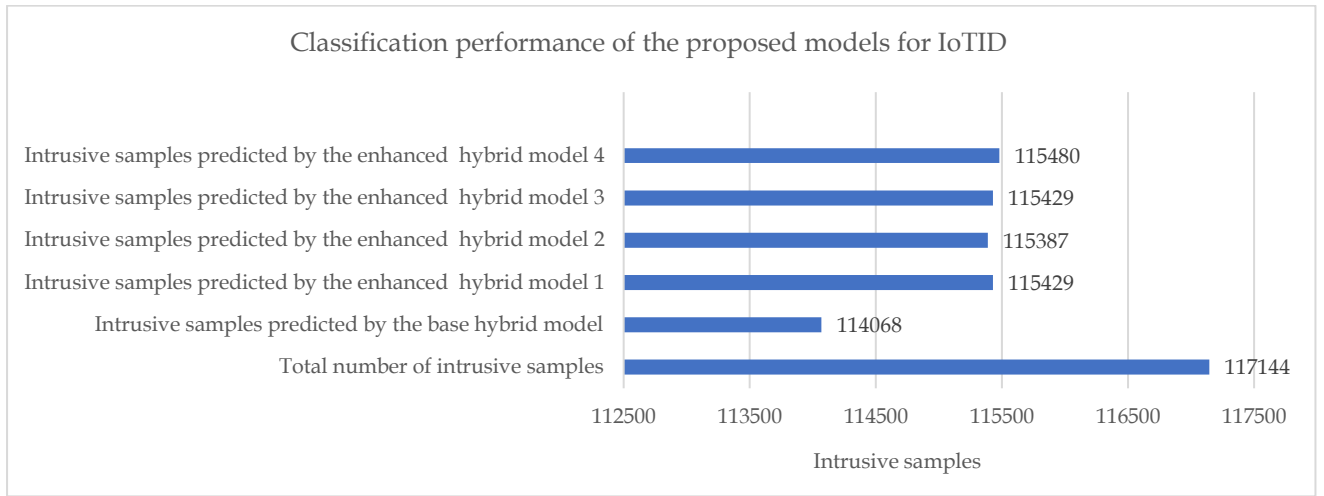


Figure 4.21: The classification performance of the proposed models for IoTID

BoT-IoT-2018

The confusion matrix of BoT-IoT for each class is shown below in Tables (4.18 – 4.21):

The confusion-matrix for the DDoS class is shown in Table 4.18.

Table 4.18: The confusion-matrix for the DDoS class

Predicted values	Actual values												
		DDoS (P)						Other classes (N)					
			Base	M1	M2	M3	M4		Base	M1	M2	M3	M4
DDoS (P)	(TP)	385000	385215	385215	385215	385307	(FN)	324	109	109	109	17	
Other classes (N)	(FP)	266	219	219	219	99	(TN)	348203	348250	348250	348250	348370	

Detection rate

$$\text{Base} = 385000/385000 + 324 = 385000/385324$$

$$\text{M1} = 385215/385215 + 109 = 385215/385324$$

$$\text{M2} = 385215/385215 + 109 = 385215/385324$$

$$\text{M3} = 385215/385215 + 109 = 385215/385324$$

$$\text{M4} = 385307/385307 + 17 = 385307/385324$$

There are 385324 DDoS samples in a dataset and 385000 of them are correctly classified as DDoS by the base hybrid model, whereas 385215 of them are correctly classified by the enhanced hybrid models M1, M2 and M3, whereas 385307 of them are correctly classified by the enhanced hybrid model M4. Larger number of DDoS samples are correctly classified by the four enhanced hybrid models than our base hybrid model.

The confusion-matrix for the DoS class is shown in Table 4.19

Table 4.19: The confusion-matrix for the DoS class

Predicted values	Actual values											
	DoS (P)						Other classes (N)					
		Base	M1	M2	M3	M4		Base	M1	M2	M3	M4
DoS (P)	(TP)	329740	329825	329825	329890	329955	(FN)	312	227	227	162	97
Other classes (N)	(FP)	262	317	317	222	217	(TN)	403479	403424	403424	403519	403524

Detection rate

$$\text{Base} = 329740/329740 + 312 = 329740/330052$$

$$\text{M1} = 329825/329825 + 227 = 329825/ 330052$$

$$\text{M2} = 329825/329825 + 227 = = 329825/ 330052$$

$$\text{M3} = 329890/329890 + 162 = 329890/ 330052$$

$$\text{M4} = 329955/329955 + 97 = 329955/330052$$

There are 330052 DoS samples in a dataset and 329740 of them are correctly classified as DoS, by the base hybrid model, 329825 of them are correctly classified by the enhanced hybrid models M1 and M2, 329890 of them are correctly classified by the enhanced hybrid model M3, 329955 of them are correctly classified by the enhanced hybrid model M4. Larger number of DoS samples are correctly classified by the four enhanced hybrid models than our base hybrid model.

The confusion-matrix for the Reconnaissance class is shown in Table 4.20.

Table 4.20: The confusion-matrix for the Reconnaissance class

Predicted values		Actual values										
		Reconnaissance (P)						Other classes (N)				
			Base	M1	M2	M3	M4		Base	M1	M2	M3
Reconnaissance (P)	(TP)	18285	18291	18291	18295	18304	(FN)	22	16	16	12	3
Other classes (N)	(FP)	9	16	16	16	16	(TN)	715477	715470	715470	715470	715470

Detection rate

$$\text{Base} = 18285 / 18285 + 22 = 18285/18307$$

$$\text{M1} = 18291/18291 + 16 = 18291/ 18307$$

$$\text{M2} = 18291/18291 + 16 = 18291/ 18307$$

$$\text{M3} = 18295 / 18295 + 12 = 18295/18307$$

$$\text{M4} = 18304/18304 + 3 = 18304/18307$$

There are 18307 Reconnaissance samples in a dataset and 18285 of them are correctly classified as Reconnaissance, by the base hybrid model, whereas 18291 of them are correctly classified by the enhanced hybrid models M1 and M2, 18295 of them are correctly classified by the enhanced hybrid model M3, whereas 18304 of them are correctly classified by the enhanced hybrid model M4. Larger number of Reconnaissance samples are correctly classified by the four enhanced hybrid models than our base hybrid model.

The confusion-matrix for the Theft class is shown in Table 4.21.

Table 4.21: The confusion-matrix for the Theft class

Predicted values		Actual values										
		Theft (P)					Other classes (N)					
			Base	M1	M2	M3	M4		Base	M1	M2	M3
Theft (P)	(TP)	13	14	14	14	14	(FN)	2	1	1	1	1
Other classes (N)	(FP)	1	1	1	1	1	(TN)	733777	733777	733777	733777	733777

Detection rate

$$\text{Base} = 13/13 + 2 = 13/15$$

$$M1 = \frac{14}{14} + 1 = \frac{14}{15} \quad M2 = \frac{14}{14} + 1 = \frac{14}{15}$$

$$M3 = \frac{14}{14} + 1 = \frac{14}{15} \quad M4 = \frac{14}{14} + 1 = \frac{14}{15}$$

There are 15 Theft samples in a dataset and 13 of them are of them are correctly classified as Theft, by the base hybrid model, whereas 14 of them are correctly classified by all the enhanced hybrid models M1, M2, M3, M4 and hence the detection rate of all the enhanced models is higher than our proposed base model.

The confusion-matrix for the Normal class is shown in Table 4.22

Table 4.22: The confusion-matrix for the Normal class

Predicted values	Actual values										
		Normal					Other classes				
		Base	M1	M2	M3	M4	Base	M1	M2	M3	M4
Normal	90	92	92	91	93	5	3	3	4	2	
Other classes	5	3	3	4	2	733693	733695	733695	733694	733696	

Detection Rate

$$\text{Base} = \frac{90}{90 + 5} = \frac{90}{95}$$

$$M1 = \frac{92}{92 + 3} = \frac{92}{95} \quad M2 = \frac{92}{92 + 3} = \frac{92}{95}$$

$$M3 = \frac{91}{91 + 4} = \frac{91}{95} \quad M4 = \frac{93}{93 + 2} = \frac{93}{95}$$

There are 95 benign samples in a dataset and 90 of them are of them are correctly classified as benign by the base hybrid model, 92 of them are correctly classified by the enhanced hybrid models M1 and M2, 91 of them are correctly by the enhanced hybrid model M3, whereas 93 of them are correctly classified by the enhanced hybrid model M4 and hence the detection rate of all the enhanced hybrid models is slightly higher than our proposed base hybrid model.

Totally, out of 733698 intrusive samples in BoT-IoT, 733580 samples are correctly detected by the enhanced hybrid model 4, 733414 samples are correctly detected by the enhanced hybrid models 1 and 3, 733345 samples are correctly detected by the enhanced hybrid

model 2 and 733038 samples are correctly predicted by the base hybrid model. The enhanced hybrid models outperform our base hybrid model and enhanced hybrid model 4 outperforms the other three enhanced hybrid models.

The bar graph that illustrates the classification performance of the proposed models for BoT-IoT are shown in Figure 4.22.

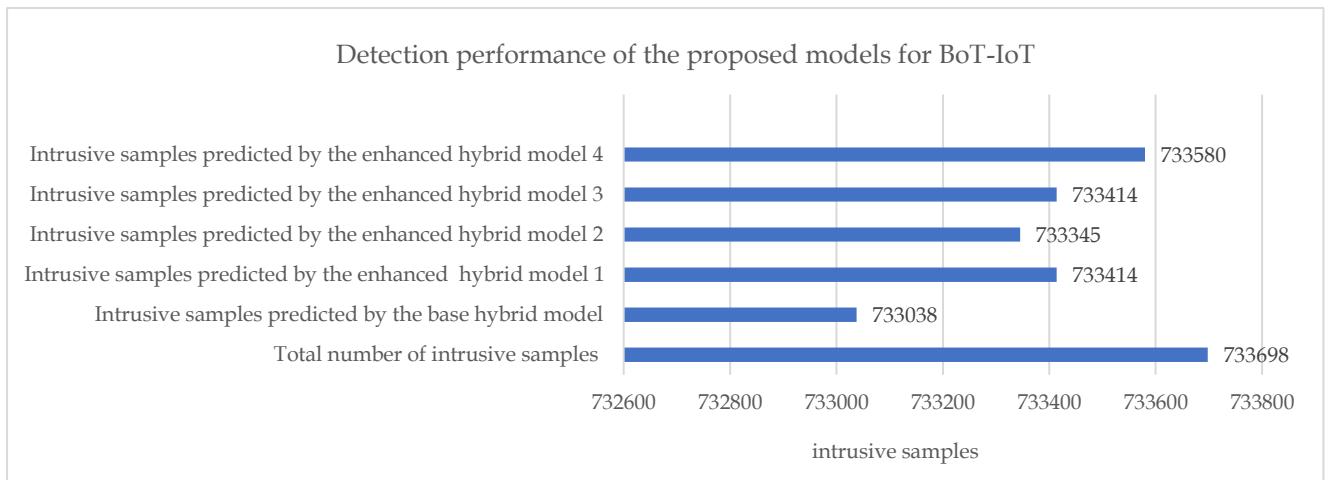


Figure 4.22: The classification performance of the proposed models for BoT-IoT

CIC-IDS-2017

The confusion matrix of CIC-IDS for each class are discussed below in Tables (4.23 – 4.28):

The confusion-matrix for the DoS Hulk class is shown in Table 4.23.

Table 4.23: The confusion-matrix for the DoS Hulk class

Predicted values	Actual values												
		DoS Hulk (P)						Other classes (N)					
			Base	M1	M2	M3	M4		Base	M1	M2	M3	M4
DoS Hulk (P)	(TP)	46109	46122	46131	46122	46133	(FN)	36	23	14	23	12	
Other classes (N)	(FP)	101	23	45	23	27	(TN)	85551	85629	85607	85629	85625	

Detection Rate

$$\text{Base} = 46109/46109 + 36 = 46109/46145$$

$$\text{M1} = 46122/46122 + 23 = 46122/46145 \qquad \text{M2} = 46131/46131 + 14 = 46131/46145$$

$$\text{M3} = 46122/46122 + 23 = 46122/46145 \qquad \text{M4} = 46133/46133 + 12 = 46133/46145$$

Among the 46145 DoS Hulk category samples in a dataset, 46109 samples are correctly predicted as DoS Hulk by the base hybrid model, whereas 46122 samples are correctly predicted by the enhanced hybrid models M1 and M3. 46131 samples are correctly predicted as DoS Hulk by the enhanced hybrid model M2, whereas 46133 samples are correctly predicted as DoS Hulk by the enhanced hybrid model M4. The detection rate of all the four enhanced hybrid models is higher than our base hybrid model.

The confusion-matrix for the DoS Golden Eye class is shown in Table 4.24.

Table 4.24: The confusion-matrix for the DoS Golden Eye class

Predicted values	Actual values											
		DoS Golden Eye (P)					Other classes (N)					
		Base	M1	M2	M3	M4		Base	M1	M2	M3	M4
		TP	1647	1669	1656	1669	1670	FN	30	8	21	8
Other classes (N)	FP	12	8	25	8	8	TN	130108	130112	130095	130112	130112

Detection Rate

$$\text{Base} = 1647/1647 + 30 = 1647/1677$$

$$\text{M1} = 1669/1669 + 8 = 1669/1677 \qquad \text{M2} = 1656/1656 + 21 = 1656/1677$$

$$\text{M3} = 1669/1669 + 8 = 1669/1677 \qquad \text{M4} = 1670/1670 + 7 = 1670/1677$$

Among the 1677 DoS Golden Eye category samples in a dataset, 1647 samples are correctly predicted as DoS Golden Eye by the base hybrid model, 1669 samples are correctly predicted by the enhanced hybrid models M1 and M3, 1656 samples are correctly predicted by the enhanced hybrid model M2, whereas 1670 samples are correctly predicted by the enhanced

hybrid model M4. Thus, the detection rate of all the four enhanced hybrid models is higher than our base hybrid model.

The confusion-matrix for the DoS Slow Loris class is shown in Table 4.25.

Table 4.25: The confusion-matrix for the DoS Slow Loris class

Predicted values		Actual values										
		DoS Slow Loris (P)						Other classes (N)				
			Base	M1	M2	M3	M4		Base	M1	M2	M3
DoS Slow Loris (P)	(TP)	1115	1121	1114	1121	1121	(FN)	10	4	11	4	4
Other classes (N)	(FP)	8	11	12	11	5	(TN)	130664	130661	130660	130661	130667

Detection rate

$$\text{Base} = 1115/1115 + 10 = 1115/1125$$

$$\text{M1} = 1121/1121 + 4 = 1121/1125$$

$$\text{M2} = 1114/1114 + 11 = 1114/1125$$

$$\text{M3} = 1121/1121 + 4 = 1121/1125$$

$$\text{M4} = 1121/1121 + 4 = 1121/1125$$

Among the 1125 DoS Slow Loris category samples in a dataset, 1115 samples are correctly predicted as DoS Slow Loris by the base hybrid model, 1114 samples are correctly predicted by the enhanced hybrid model M2, whereas 1121 samples are correctly predicted by the enhanced hybrid models M1, M2 and M4. Thus, the detection rate of all the four enhanced hybrid models is higher than our base hybrid model.

The confusion-matrix for the DoS Slow Http Test class is shown in Table 4.26.

Table 4.26: The confusion-matrix for the DoS Slow Http Test class

Predicted values		Actual values										
		DoS Slow Http Test (P)						Other classes (N)				
			Base	M1	M2	M3	M4		Base	M1	M2	M3
DoS Slow Http Test (P)	(TP)	1061	1074	1066	1074	1075	(FN)	16	3	11	3	2
Other classes (N)	(FP)	5	3	3	3	3	(TN)	130715	130717	130717	130717	130717

Detection rate

$$\text{Base} = 1061/1061 + 16 = 1061/1077$$

$$\text{M1} = 1074/1074 + 3 = 1074/1077$$

$$\text{M2} = 1066/1066 + 11 = 1066/1077$$

$$\text{M3} = 1074/1074 + 3 = 1074/1077$$

$$\text{M4} = 1075/1075 + 2 = 1075/1077$$

Among the 1077 DoS Slow HTTP Test category samples in a dataset, 1061 samples are correctly predicted as DoS Slow HTTP Test by the base hybrid model, whereas 1074 samples are correctly predicted by the enhanced hybrid models M1 and M3, 1066 samples are correctly predicted by the enhanced hybrid model M2, whereas 1075 samples are correctly predicted by the enhanced hybrid model M4. Thus the detection rate of all the four enhanced hybrid models is higher than our base hybrid model.

The confusion-matrix for the Heartbleed class is shown in Table 4.27.

Table 4.27: The confusion-matrix for the Heartbleed class

Predicted values		Actual values										
		Heartbleed (P)						Other classes (N)				
			Base	M1	M2	M3	M4		Base	M1	M2	M3
Heartbleed (P)	(TP)	5	5	5	5	5	(FN)	0	0	0	0	0
Other classes (N)	(FP)	0	0	0	0	0	(TN)	131792	131792	131792	131792	131792

Detection rate

$$\text{Base} = 5/5 + 0 = 5/5$$

$$\text{M1} = 5/5 + 0 = 5/5$$

$$\text{M2} = 5/5 + 0 = 5/5$$

$$\text{M3} = 5/5 + 0 = 5/5$$

$$\text{M4} = 5/5 + 0 = 5/5$$

Among the Heartbleed samples in a dataset, all the 5 samples are correctly predicted as Heartbleed by the proposed models. The detection rate is almost similar for all the proposed models.

The confusion-matrix for the Normal class is shown in Table 4.28.

Table 4.28: The confusion-matrix for the Normal class

Predicted values	Actual values										
		Normal					Other classes				
		Base	M1	M2	M3	M4	Base	M1	M2	M3	M4
Normal	81670	81703	81711	81703	81720	98	65	57	65	48	
Other classes	75	80	72	80	64	49954	49949	49957	49949	49965	

Detection rate

$$\text{Base} = 81670/81670 + 98 = 81670/81768$$

$$\text{M1} = 81703/81703 + 65 = 81703/81768$$

$$\text{M2} = 81711/81711 + 57 = 81711 /81768$$

$$\text{M3} = 81703/81703 + 65 = 81703/81768$$

$$\text{M4} = 81720/81720 + 64 = 81720/81768$$

Among the 81768 normal category samples in a dataset, 81670 samples are correctly predicted as normal by the base hybrid model, 81703 samples are correctly predicted by the enhanced hybrid models M1 and M3, 81711 samples are correctly predicted by the enhanced hybrid model M2, whereas 81720 samples are correctly predicted by the enhanced hybrid model M4. Thus the detection rate of the four enhanced hybrid models is higher than our base hybrid model.

Out of 50029 intrusive samples in CIC-IDS, 50,002 samples are correctly detected by the enhanced hybrid model 4, 49991 samples are correctly detected by the enhanced hybrid models M2 and M3, 49974 samples are correctly detected by the enhanced hybrid model M1 and 49937 samples are correctly detected by the base hybrid model. The enhanced hybrid models outperform the base hybrid model and the enhanced hybrid model 4 outperforms the other three enhanced hybrid models.

The bar graph that illustrates the classification performance of the proposed models for CIC-IDS are shown in Figure 4.23.

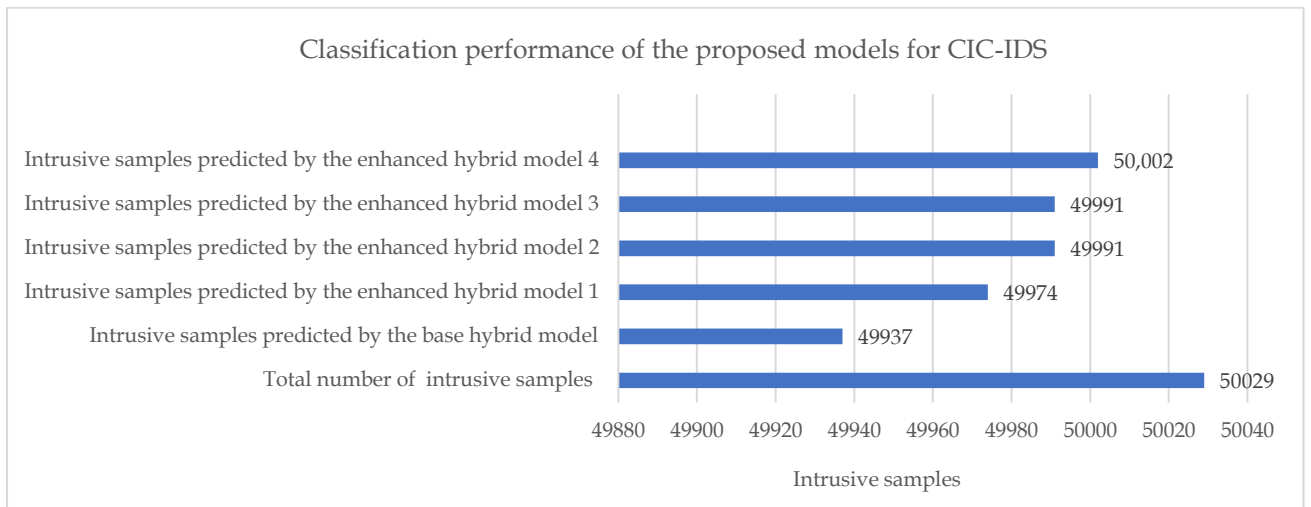


Figure 4.23: The classification performance of the proposed models for CIC-IDS

CIDDS-001

The confusion matrix of CIDDS-001 for each class are discussed below in Tables (4.29 – 4.33):

The confusion-matrix for the DoS class is shown in Table 4.29.

Table 4.29: The confusion-matrix for the DoS class

Predicted values	Actual values											
		DoS (P)					Other classes (N)					
			Base	M1	M2	M3	M4		Base	M1	M2	M3
DoS (P)	(TP)	29120	29150	29150	29150	29199	(FN)	93	63	63	63	14
Other classes (N)	(FP)	231	102	93	111	93	(TN)	53314	53443	53452	53434	53452

Detection rate

$$\text{Base} = \frac{29120}{29120 + 93} = \frac{29120}{29213}$$

$$\text{M1} = \frac{29150}{29150 + 63} = \frac{29150}{29213}$$

$$\text{M2} = \frac{29150}{29150 + 63} = \frac{29150}{29213}$$

$$\text{M3} = \frac{29150}{29150 + 63} = \frac{29150}{29213}$$

$$\text{M4} = \frac{29199}{29199 + 14} = \frac{29199}{29213}$$

There are 29213 DoS samples in a dataset, 29120 of them are correctly classified as DoS by the base hybrid model, 29150 of them are correctly classified as DoS by the enhanced hybrid models M1, M2, and M3, whereas 29199 of them are correctly classified as DoS by the enhanced hybrid model M4. Larger number of DoS samples are correctly classified by the four enhanced hybrid models than our base hybrid model.

The confusion-matrix for the Brute Force class is shown in Table 4.30.

Table 4.30: The confusion-matrix for the Brute Force class

Predicted values	Actual values											
		Brute Force (P)					Other classes (N)					
		Base	M1	M2	M3	M4		Base	M1	M2	M3	M4
Brute Force (P)	(TP)	1518	1527	1527	1529	1538	(FN)	24	15	15	13	4
Other classes (N)	(FP)	29	15	15	15	15	(TN)	81187	81201	81201	81201	81201

Detection rate

$$\text{Base} = 1518/1518 + 24 = 1518/1542$$

$$\text{M1} = 1527/1527 + 15 = 1527/1542 \quad \text{M2} = 1527/1527 + 15 = 1527/1542$$

$$\text{M3} = 1529/1529 + 13 = 1529/1542 \quad \text{M4} = 1538/1538 + 15 = 1538/1542$$

There are 1542 Brute Force samples in a dataset and 1538 of them are correctly classified as Brute Force by the base hybrid model, whereas 1527 of them are correctly classified by the enhanced hybrid models M1 and M2, 1529 of them are correctly classified by the enhanced hybrid model M3, whereas 1538 of them are correctly classified by the enhanced hybrid model M4. The detection rate of all the enhanced hybrid models is slightly higher than our proposed base hybrid model.

The confusion-matrix for the Port Scan class is shown in Table 4.31.

Table 4.31: The confusion-matrix for the Port Scan class

Predicted values	Actual values											
		Port Scan(P)					Other classes (N)					
		Base	M1	M2	M3	M4		Base	M1	M2	M3	M4
Port Scan (P)	(TP)	5631	5652	5646	5652	5663	(FN)	38	11	17	11	6
Other classes (N)	(FP)	16	11	17	11	17	(TN)	77073	77078	77072	77078	77072

Detection rate

$$\text{Base} = 5631/5631 + 38 = 5631/5669$$

$$\text{M1} = 5652/5652 + 11 = 5652/5669$$

$$\text{M2} = 5646/5646 + 17 = 5646/5669$$

$$\text{M3} = 5652/5652 + 11 = 5652/5669$$

$$\text{M4} = 5663/5663 + 9 = 5663/5669$$

There are 5669 Port Scan samples in a dataset and 5631 of them are correctly classified as Port Scan by the base hybrid model, 5652 of them are correctly classified as Port Scan, by the enhanced hybrid models M1 and M3, 5646 of them are correctly classified as Port Scan, by the enhanced hybrid model M2, whereas 5663 of them are correctly classified as Port Scan, by the enhanced hybrid model M4. The detection rate of all the four enhanced hybrid models is higher than our proposed base hybrid model.

The confusion-matrix for the Ping Scan class is shown in Table 4.32.

Table 4.32: The confusion-matrix for the Ping Scan class

Predicted values	Actual values											
		Ping Scan (P)					Other classes (N)					
		Base	M1	M2	M3	M4		Base	M1	M2	M3	M4
Ping Scan (P)	(TP)	1144	1160	1159	1160	1161	(FN)	29	13	14	13	12
Other classes (N)	(FP)	16	14	14	14	12	(TN)	81569	81571	81571	81571	81573

Detection rate

$$\text{Base} = 1144/1144 + 29 = 1144/1173$$

$$\text{M1} = 1160/1160 + 13 = 1160/1173$$

$$\text{M2} = 1159/1159 + 14 = 1159/1173$$

$$\text{M3} = 1160/1160 + 13 = 1160/1173$$

$$\text{M4} = 1161/1161 + 12 = 1161/1173$$

There are 1173 Ping Scan samples in a dataset and 1144 of them are correctly classified as Ping Scan by the base hybrid model, 1160 of them are correctly classified as Ping Scan by the enhanced hybrid models M1 and M3, 1159 of them are correctly classified as Ping Scan by the enhanced hybrid model M2, whereas 1161 of them are correctly classified as Ping Scan by the enhanced hybrid model M4. The detection rate of all the enhanced hybrid models is higher than our proposed base hybrid model.

The confusion-matrix for the Normal class is shown in Table 4.33.

Table 4.33: The confusion-matrix for the Normal class

Predicted values	Actual values									
	Normal					Other classes				
	Base	M1	M2	M3	M4	Base	M1	M2	M3	M4
Normal	44918	45130	45131	45130	45134	243	31	30	31	27
Other classes	53	31	30	31	35	37544	37566	37567	37566	37562

Detection rate

$$\text{Base} = 44918 / (44918 + 243) = 44918/45161$$

$$\text{M1} = 45130/(45130 + 31) = 45130/45161 \quad \text{M2} = 45131/(45131 + 30) = 45131/45161$$

$$\text{M3} = 45130/(45130 + 31) = 45130/45161 \quad \text{M4} = 45134/(45134 + 27) = 45134/45161$$

Among the 45161 normal category samples in a dataset, 44918 samples are correctly predicted as normal by the base hybrid model, whereas 45130 samples are correctly predicted by the enhanced hybrid models M1 and M3, 45131 samples are correctly predicted by the enhanced hybrid model M2, whereas 45134 samples are correctly predicted by the enhanced hybrid model M4. The detection rate all the four enhanced hybrid models is higher than our base hybrid model.

Out of 37597 intrusive samples in CIDDS-001, 37503 samples are correctly detected by the enhanced hybrid model 4, 37458 samples are correctly detected by the enhanced hybrid model M2, 37460 samples are correctly detected by the enhanced hybrid models M1 and M3,

37413 samples are correctly detected by the base hybrid model. The enhanced hybrid models outperform the base hybrid model and the enhanced hybrid model 4 outperforms the other three enhanced hybrid models.

The bar graph that illustrates the classification performance of the proposed models for CIDDS-001 are shown in Figure 4.24.

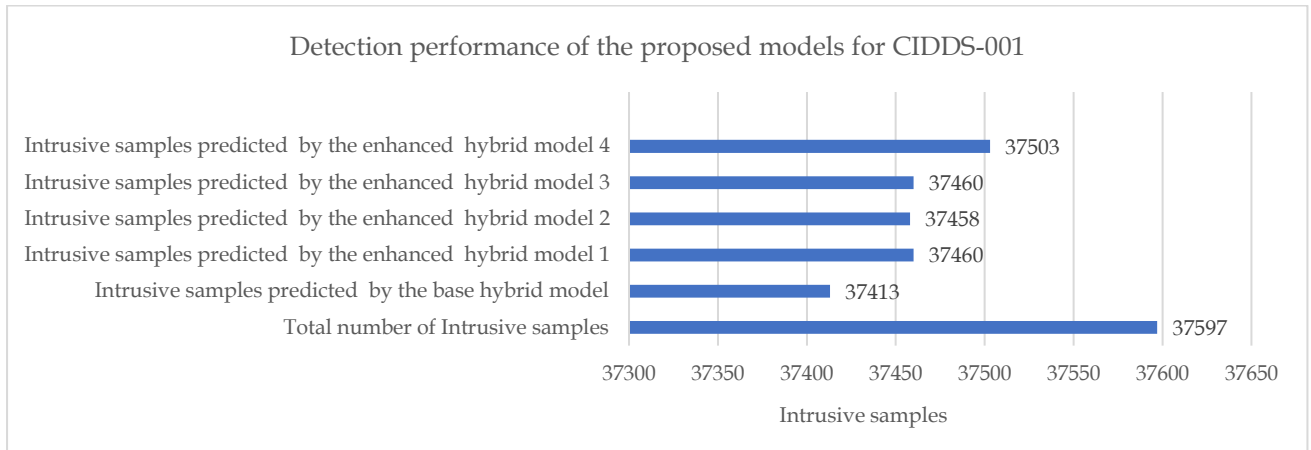


Figure 4.24: The classification performance of the proposed models for CIDDS-001

ISCX-URL 2016

The confusion matrix of ISCX-URL for each class is shown below in Tables (4.34 - 4.38)

The confusion-matrix for the Malware class is shown in Table 4.34.

Table: 4.34 The confusion-matrix for the Malware class

Predicted values	Actual values											
		Malware (P)					Other classes (N)					
			Base	M1	M2	M3	M4	(FN)	Base	M1	M2	M3
Malware (P)	(TP)	2309	2310	2310	2310	2312	(FN)	70	71	71	71	68
Other classes (N)	(FP)	90	71	71	71	68	(TN)	20410	20429	20429	20429	20432

Detection rate

$$\text{Base} = 2309/2309 + 71 = 2309/2380$$

$$\text{M1} = 2310/2310 + 70 = 2310/2380$$

$$\text{M2} = 2310/2310 + 70 = 2310/2380$$

$$M3 = 2310/2310 + 70 = 2310/2380$$

$$M4 = 2312/2312 + 68 = 2312/2380$$

Out of 2380 malware class samples in a test set, only 2309 samples are correctly predicted as malware by the base hybrid model, 2310 samples are correctly classified by the enhanced hybrid models M1, M2 and M3, whereas 2312 samples are correctly classified by the enhanced hybrid model M4. The detection rate of all the enhanced hybrid models is higher than the base hybrid model.

The confusion-matrix for the Defacement class is shown in Table 4.35.

Table 4.35: The confusion-matrix for the Defacement class

Predicted values	Actual values											
		Defacement (P)					Other classes (N)					
		Base	M1	M2	M3	M4	Base	M1	M2	M3	M4	
Defacement (P)	(TP)	8860	8918	8918	8918	8918	(FN)	240	182	182	182	182
Other classes (N)	(FP)	222	185	184	185	182	(TN)	13558	13595	13596	13595	13598

Detection rate

$$\text{Base} = 8860/8860 + 240 = 8860/9100$$

$$M1 = 8918/8918 + 182 = 8918/9100$$

$$M2 = 8918/8918 + 182 = 8918/9100$$

$$M3 = 8918/8918 + 182 = 8918/9100$$

$$M4 = 8918/8918 + 182 = 8918/9100$$

Out of 9100 Defacement class samples in a dataset, only 8860 samples are correctly predicted as Defacement by the base hybrid model, whereas 8918 samples are correctly classified by all the enhanced hybrid models M1, M2, M3 and M4. The detection rate of all the enhanced hybrid models is higher than the base hybrid model.

The confusion-matrix for the Spam class is shown in Table 4.36.

Table 4.36: The confusion-matrix for the Spam class

Predicted values	Actual values											
		Spam (P)					Other classes (N)					
		Base	M1	M2	M3	M4		Base	M1	M2	M3	M4
Spam (P)	(TP)	2325	2331	2331	2331	2345	(FN)	75	69	69	69	55
Other classes (N)	(FP)	59	69	69	69	56	(TN)	20421	20411	20411	20411	20424

Detection rate

$$\text{Base} = 2325/2325 + 75 = 2325/2400$$

$$\text{M1} = 2331/2331 + 69 = 2331/2400$$

$$\text{M2} = 2331/2331 + 69 = 2331/2400$$

$$\text{M3} = 2331/2331 + 69 = 2331/2400$$

$$\text{M4} = 2345/2345 + 55 = 2345/2400$$

Out of 2400 Spam class samples in a test set, only 2325 samples are correctly predicted as Spam by the base hybrid model, whereas 2331 samples are correctly classified by the enhanced hybrid models M1, M2 and M3, whereas 2345 samples are correctly classified by the enhanced hybrid model M4. The detection rate of the enhanced hybrid models is higher than the base hybrid model.

The confusion-matrix for the Phishing class is shown in Table 4.37.

Table 4.37: The confusion-matrix for the Phishing class

Predicted values	Actual values											
		Phishing (P)					Other classes (N)					
		Base	M1	M2	M3	M4		Base	M1	M2	M3	M4
Phishing (P)	(TP)	1886	1903	1903	1908	1908	(FN)	114	97	97	92	92
Other classes (N)	(FP)	115	97	97	97	93	(TN)	20765	20783	20783	20783	20787

Detection rate

$$\text{Base} = 1886/1886 + 114 = 1886/2000$$

$$\text{M1} = 1903/1903 + 97 = 1903/2000$$

$$\text{M2} = 1903/1903 + 97 = 1903/2000$$

$$\text{M3} = 1908/1908 + 92 = 1908/2000$$

$$\text{M4} = 1908/1908 + 92 = 1908/2000$$

Out of 2000 Phishing category samples in a test set, only 1886 samples are correctly predicted as phishing by the base hybrid model, 1903 samples are correctly classified by the enhanced hybrid models M1 and M2, whereas 1908 samples are correctly classified by the enhanced hybrid models M3 and M4. The detection rate of the enhanced hybrid models is slightly higher than the base hybrid model.

The confusion-matrix for the Normal class is shown in Table 4.38.

Table 4.38: The confusion-matrix for the Normal class

Predicted values	Actual values										
		Normal					Other classes				
		Base	M1	M2	M3	M4	Base	M1	M2	M3	M4
Normal	6930	6955	6955	6955	6956	70	45	45	45	44	
Other classes	70	52	48	52	45	15810	15828	15832	15828	15835	

Detection rate

$$\text{Base} = 6930 / (6930 + 70) = 6930 / 7000$$

$$\text{M1} = 6955 / (6955 + 45) = 6955 / 7000$$

$$\text{M2} = 6955 / (6955 + 45) = 6955 / 7000$$

$$\text{M3} = 6955 / (6955 + 45) = 6955 / 7000$$

$$\text{M4} = 6956 / (6956 + 44) = 6956 / 7000$$

Out of 7000 normal category samples in a test set, only 6930 samples are correctly predicted as normal by the base hybrid model, 6955 samples are correctly classified by the enhanced hybrid models M1, M2 and M3, whereas 6956 samples are correctly classified by the enhanced hybrid model M4. The detection rate of all the enhanced hybrid models is higher than the base hybrid model.

Out of 15800 intrusive samples in ISCX-URL, 15481 samples are correctly detected by the enhanced hybrid model 4, 15466 samples are correctly detected by the enhanced hybrid model M3, 15461 samples are correctly detected by the enhanced hybrid models M1 and M2, whereas 15,380 samples are correctly detected by the base hybrid model. The enhanced hybrid

models outperform the base hybrid model and the enhanced hybrid model 4 outperforms the other three enhanced hybrid models.

The bar graph that illustrates the classification performance of the proposed models for ISCX-URL are shown in Figure 4.25.

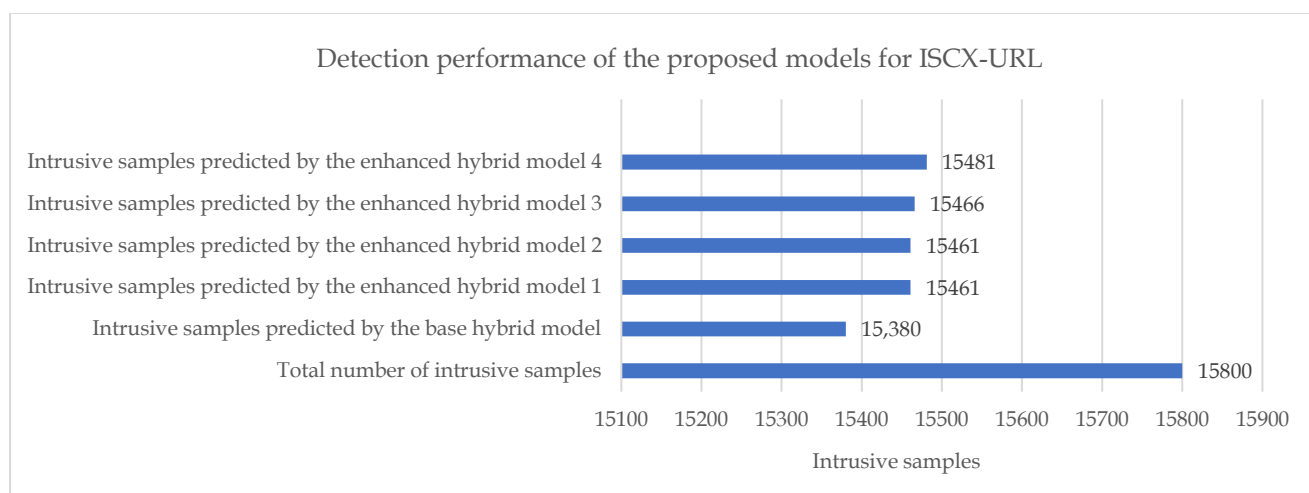


Figure 4.25: The classification performance of the proposed models for ISCX-URL

CIDDS-002

The confusion matrix of CIDDS-002 is shown Table 4.39.

Table 4.39: The confusion matrix of CIDDS-002

Predicted values		Actual values										
		Port Scan (P)						Other classes (N)				
			Base	M1	M2	M3	M4		Base	M1	M2	M3
Port Scan (P)	(TP)	7208	7231	7245	7245	7258	TN	57	34	20	20	7
Other classes (N)	(FP)	26	49	45	47	19	FN	202424	202401	202405	202403	202431

Detection rate

$$\text{Base} = 7208/7208 + 57 = 7208/7265$$

$$\text{M1} = 7231/7231 + 34 = 7231/7265 \quad \text{M2} = 7245/7245 + 20 = 7245/7265$$

$$\text{M3} = 7245/7245 + 20 = 7245/7265 \quad \text{M4} = 7258/7258 + 7 = 7258/7265$$

There are 7264 intrusive Port Scan samples in a dataset and 7208 of them are correctly classified as Port Scan by the base hybrid model, 7245 of them are correctly classified by the

enhanced hybrid models M2 and M3, 7231 of them are correctly predicted by the enhanced hybrid model M1, whereas 7258 of them are correctly classified by the enhanced hybrid model M4. The detection rate of all the enhanced hybrid models is higher than our base hybrid model. The detection rate of the enhanced hybrid model 4 is higher than the other three enhanced hybrid models.

The bar graph that illustrates the classification performance of the proposed models for CIDDS-002 are shown in Figure 4.26.

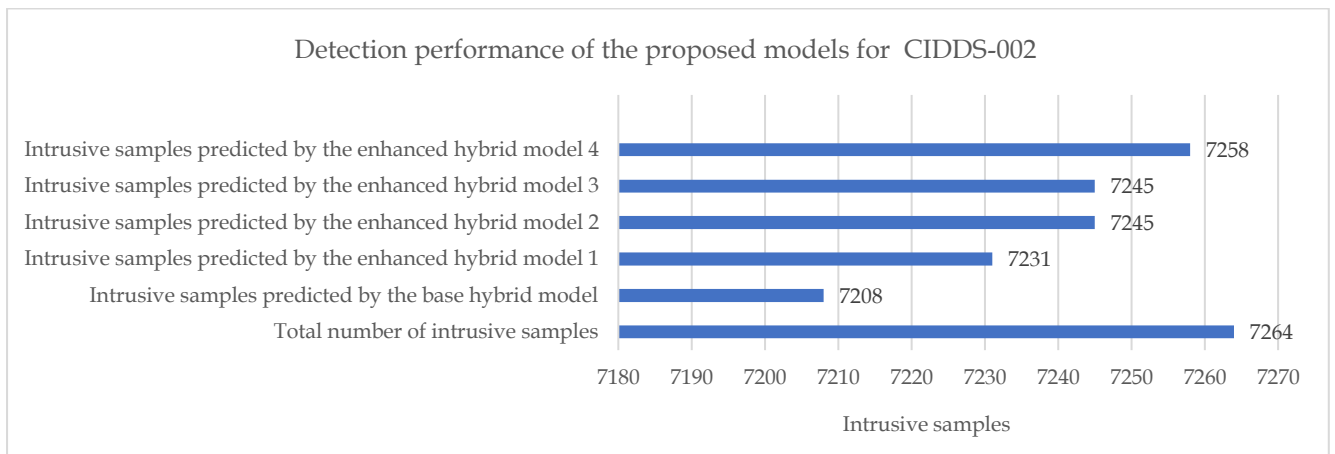


Figure 4.26: The classification performance of the proposed models for CIDDS-002

ISCX-TOR 2016

The confusion matrix of ISCX-TOR is shown in Table 4.40.

Table 4.40: The confusion matrix of ISCX-TOR

Predicted values	Actual values											
		Tor (P)					Non-Tor (N)					
			Base	M1	M2	M3	M4	TN	Base	M1	M2	M3
Tor (P)	(TP)	4145	4229	4229	4225	4235		370	286	286	290	280
Non-Tor (N)	(FP)	342	300	295	293	286	FN	25549	25591	25596	25598	25605

Detection Rate

$$\text{Base} = 4145/4145 + 370 = 4145/4515$$

$$\text{M1} = 4229/4229 + 286 = 4229/4515 \quad \text{M2} = 4229/4229 + 286 = 4229/4515$$

$$\text{M3} = 4225/4225 + 290 = 4225/4515 \quad \text{M4} = 4235/4235 + 280 = 4235/4515$$

The total number of Tor intrusive samples in the test set is 4515. Among them, 4145 samples are predicted correctly as Tor by the base hybrid model, 4229 samples are predicted correctly by the enhanced hybrid models M1 and M2. 4225 samples are predicted correctly by the enhanced hybrid model M3, whereas 4235 samples are predicted correctly by the enhanced hybrid model M4. The enhanced hybrid models give higher detection rate when compared with our base hybrid model. The enhanced hybrid model 4 gives higher DR than the other three enhanced hybrid models.

The bar graph that illustrates the classification performance of the proposed models for ISCX-TOR are shown in Figure 4.27.

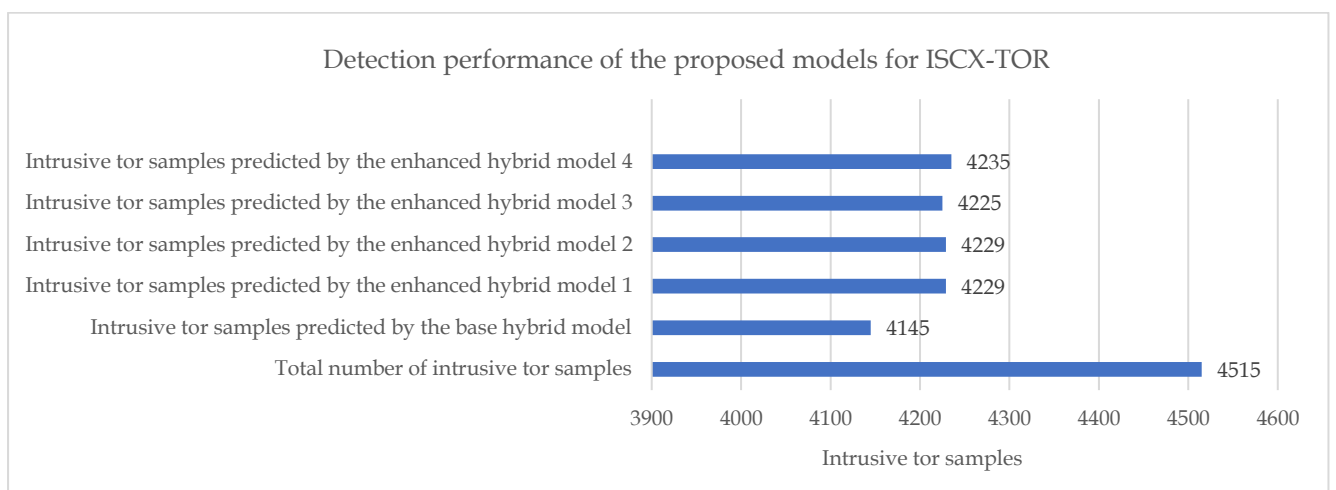


Figure 4.27: The classification performance of the proposed models for ISCX-TOR

UNSW-NB 15

The confusion matrix of UNSW-NB is shown in Table 4.41.

Table 4.41: The confusion matrix of UNSW-NB

Predicted values	Actual values												
		Attack (P)						Normal (N)					
		Base	M1	M2	M3	M4		Base	M1	M2	M3	M4	
	Attack (P)	TP	10238	10528	10522	10525	10539	TN	520	230	236	233	219
Normal (N)	FP	443	247	243	240	230	FN	21768	21964	21968	21971	21981	

Detection Rate

$$\text{Base} = 10238 / (10238 + 520) = 10238 / 10758$$

$$\text{M1} = 10528 / (10528 + 230) = 10528 / 10758 \quad \text{M2} = 10522 / (10522 + 236) = 10522 / 10758$$

$$\text{M3} = 10525 / (10525 + 233) = 10525 / 10758 \quad \text{M4} = 10539 / (10539 + 219) = 10539 / 10758$$

The total number of intrusive samples in the test set is 10758. Among them, 10238 samples are predicted correctly as intrusive by the base hybrid model, 10528 samples are predicted correctly by the enhanced hybrid model M1, 10522 samples are predicted correctly by the enhanced hybrid model M2, 10525 samples are predicted correctly by the enhanced hybrid model M3, whereas 10539 samples are correctly predicted by the enhanced hybrid model M4. The enhanced hybrid models give higher detection rate than our base hybrid model. The enhanced hybrid model 4 gives highest detection rate than the other three enhanced hybrid models.

The bar graph that illustrates the classification performance of the proposed models for UNSW-NB are shown in Figure 4.28.

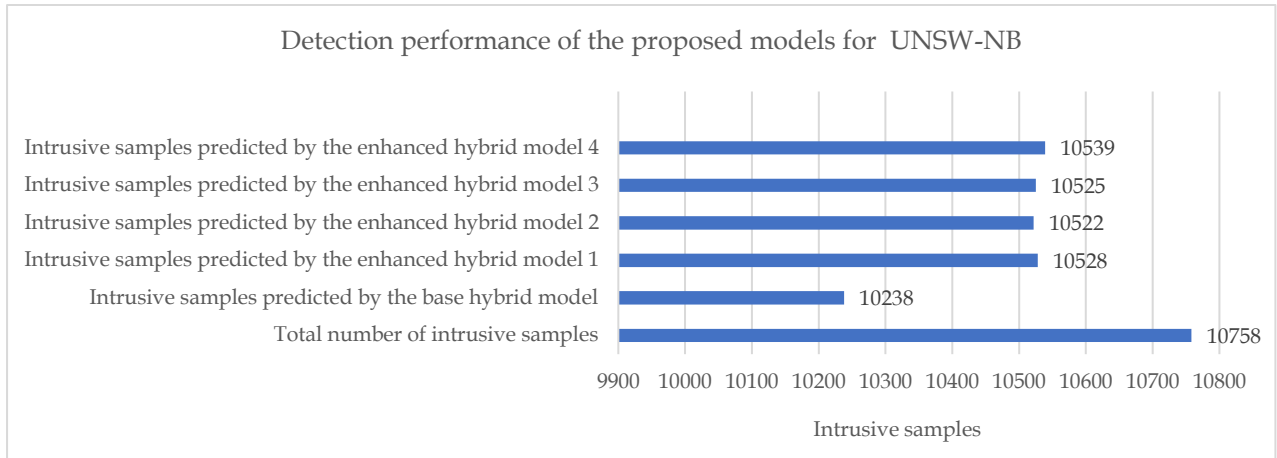


Figure 4.28: The classification performance of the proposed models for UNSW-NB

From the classification performance of the proposed models for the eight standard benchmark datasets as given in Tables (4.23 – 4.41), the detection rate of the enhanced hybrid models is higher than our base hybrid model. The enhanced hybrid model 4 outperforms the other three enhanced hybrid models. The proposed enhanced hybrid models show at least 1% increase in detection rate for the benchmark datasets, when compared with the base hybrid model.

4.4 Performance Comparison

The comparison of classification performance of the proposed models with different schemes are shown in Tables (4.42 – 4.49)

Table: 4.42 shows our proposed model results for the CIDDS-001 with different existing scheme

Table: 4.42 Proposed model results for CIDDS-001 with different existing schemes

No	Features selection	Classifier	Detection rate	Detection loss	Accuracy	Precision	F1score
1.	Statistical methods-entropy	J48	99 % (Mahajan et al., 2020)	-	94%	95%	-
2.	PCA (Grid search)	RF+ KNN	98.14% (Cuautla et al., 2020)	-	98.14%	98.33%	98.23%
3.	Ensemble feature selection	MLP	99.40% (He et al., 2019)	-	-	-	-
4.	Manual selection	CART	96.74% (Verma and Ranga, 2020)	-	-	97.30%	-
5.	Data reshape algorithm	LSTM	89.71% (Oliveira et al., 2021)	-	-	94.03%	91.66%
6.	DAE	LightGBM	99.60% (Ayubkhan et al., 2022) (Base Hybrid Model)	0.2%	99.58%	99.60%	99.59%
7.	Enhanced DAE 1	LightGBM	99.80 % (Enhanced Hybrid Model 1)	0.12%	99.80%	99.78%	99.79%
8.	Enhanced DAE 2	LightGBM	99.79% (Enhanced Hybrid Model 2)	0.13%	99.78%	99.79%	99.79%
9.	Enhanced DAE 3	LightGBM	99.80 % (Enhanced Hybrid Model 3)	0.12%	99.79%	99.77%	99.78%
10.	Enhanced DAE 4	LightGBM	99.85 % (Enhanced Hybrid Model 4)	0.07%	99.85%	99.80%	99.82%

The bar graph that illustrates the detection loss comparison for CIDDS-001 is shown in Figure 4.29.

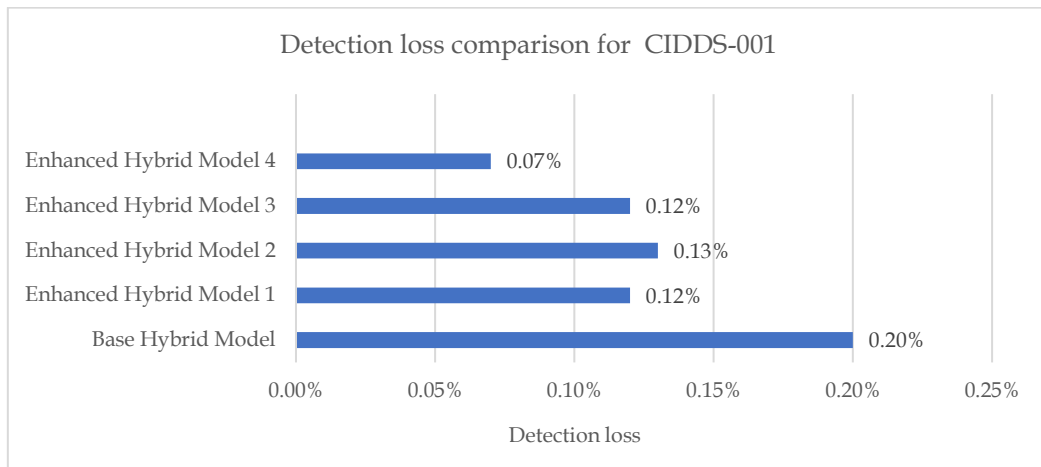


Figure 4.29: The detection loss comparison for CIDDS-001 is shown in graphical bar

The bar graph that illustrates the detection rate comparison for CIDDS-001 is shown in Figure 4.30.

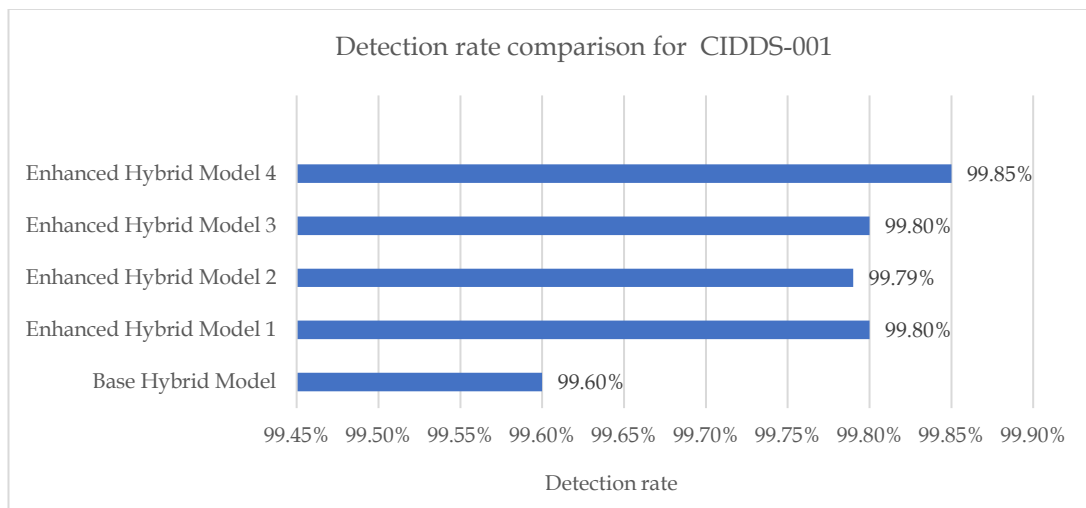


Figure 4.30: The detection rate comparison for CIDDS-001 is shown in graphical bar

On notifying the performance graphs in Figure 4.29 and Figure 4.30. the enhanced hybrid models outperform our base hybrid model by obtaining higher detection rate and minimum

detection loss. The enhanced hybrid model 4 outperform slightly the other three enhanced hybrid models by securing highest detection rate and lowest detection loss.

Table 4.43 shows our proposed model results for the CIDDS-002 with different existing schemes

Table 4. 43 : Proposed model results for CIDDS-002 with different existing schemes

No	Features selection	Classifier	Detection rate	Detection loss	Accuracy	Precision	F1score
1.	-	J48 Tree	99% (Razdan et al., 2021)	-	-	-	-
2.	-	Bagging	99.71% (Ainurrochman et al., 2021)	-	99.71%	-	-
3.	Feature importance embedding	Cart	99% (Thapa et al., 2020)	-	99.64%	-	-
4.	-	XGboost	99.64% (Quang et al., 2021)	-	-	99.60%	-
5.	DAE	LightGBM	99.90% (Ayubkhan et al., 2022) (Base Hybrid Model)	0.35%	99.89%	99.90%	99.89%
6.	Enhanced DAE 1	LightGBM	99.96 % (Enhanced Hybrid Model 1)	0.33%	99.92%	99.93%	99.94%
7.	Enhanced DAE 2	LightGBM	99.97% (Enhanced Hybrid Model 2)	0.32%	99.92%	99.95%	99.96%
8.	Enhanced DAE 3	LightGBM	99.97% (Enhanced Hybrid Model 3)	0.32%	99.93%	99.94%	99.95%
9.	Enhanced DAE 4	LightGBM	99.98% (Enhanced Hybrid Model 4)	0.30%	99.96%	99.97%	99.97%

The bar graph that illustrates the detection loss comparison for CIDDS-002 is shown in Figure 4.31.

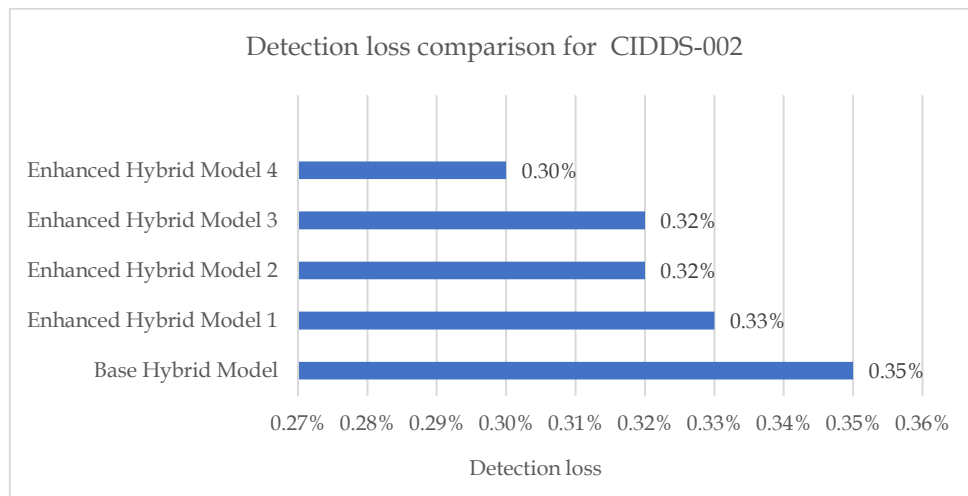


Figure 4.31: The detection loss comparison for CIDDS-002 is shown in graphical bar

The bar graph that illustrates the detection rate comparison for CIDDS-002 is shown in Figure 4.32.

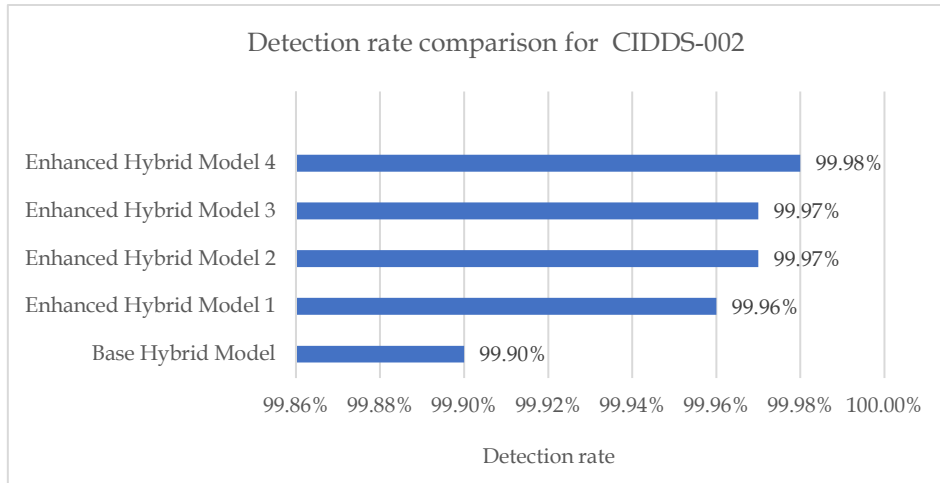


Figure 4.32: The detection rate comparison for CIDDS-002 is shown in graphical bar

On notifying the performance graphs in Figure 4.31 and Figure 4.32, the enhanced hybrid models outperform our base hybrid model by obtaining higher detection rate and minimum detection loss. The enhanced hybrid model 4 outperform slightly the other three enhanced hybrid models by securing highest detection rate and lowest detection loss.

Table 4.44 shows our proposed model results for the IoTID with different existing schemes

Table 4.44: Proposed model results for IoTID with different existing schemes

No	Features selection	Classifier	Detection rate	Detection loss	Accuracy	Precision	F1score
1.	Correlation	DT	88% (Ullah et al., 2020)	-	88%	88%	88%
2.	Extra tree classifier	CNN	97.88 % (Alkahtani et al., 2021)	-	98.33%	97.42%	97.64%
3.	PSO	CNN-RNN	98.20% (Ullah et al., 2022)	-	98%	98.40%	-
4.	-	AE	97% (Song et al., 2021)	-	95.20%	97%	-
5.	DAE	LightGBM	97.43% (Ayubkhan et al., 2022) (Base Hybrid Model)	0.10%	97.43%	97.42%	97.42%
6.	Enhanced DAE 1	LightGBM	98.58% (Enhanced Hybrid Model 1)	0.03%	98.56%	98.52%	98.57%
7.	Enhanced DAE 2	LightGBM	98.57% (Enhanced Hybrid Model 2)	0.05%	98.52%	98.54%	98.55%
8.	Enhanced DAE 3	LightGBM	98.58% (Enhanced Hybrid Model 3)	0.03%	98.50%	98.52%	98.55%
9.	Enhanced DAE 4	LightGBM	98.61% (Enhanced Hybrid Model 4)	0.01%	98.57%	98.58%	98.59%

The bar graph that illustrates the detection loss comparison for IoTID is shown in Figure 4.33.

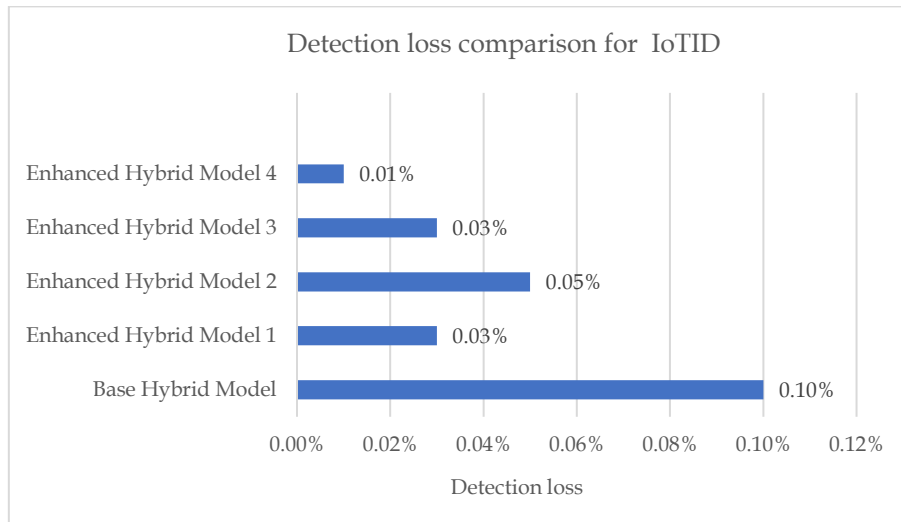


Figure 4.33: The detection loss comparison for IoTID is shown in graphical bar

The bar graph that illustrates the detection rate comparison for IoTID is shown in Figure 4.34.

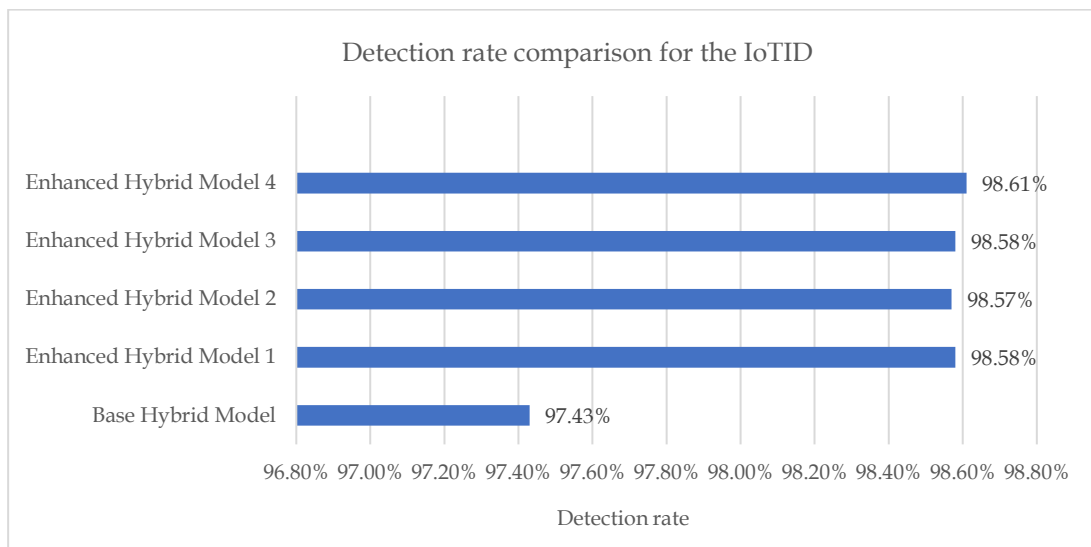


Figure 4.34: The detection rate comparison for IoTID is shown in graphical bar

On notifying the performance graphs in Figure 4.33 and Figure 4.34, the enhanced hybrid models outperform our base hybrid model by obtaining higher detection rate and minimum

detection loss. The enhanced hybrid model 4 outperform slightly the other three enhanced hybrid models by securing highest detection rate and lowest detection loss.

Table 4.45 shows our proposed model results for the ISCX-URL with different existing schemes

Table 4.45: Proposed model results for the ISCX-URL with different existing schemes

No	Features selection	Classifier	Detection rate	Detection loss	Accuracy	Precision	F1score
1.	Infogain + Ranker	RF	97 % (Mamun et al., 2016)	-	97%	97%	-
2.	Grid search +N-gram	RF+ ANN	97.26% (Alsaedi et al., 2022)	-	97.25%	97.36%	97.31%
3.	-	RF	93 % (sahib, 2022)	-	94.9%	97.4%	-
4.	Random	RF	96.1% (Kapil et al., 2019)	-	96.1%	96.1%	96.1%
5.	DAE	LightGBM	97.76 % (Ayubkhan et al., 2022) (Base Hybrid Model)	0.2%	97.72%	97.73%	97.72%
6.	Enhanced DAE 1	LightGBM	97.97% (Enhanced Hybrid Model 1)	0.12%	97.95%	97.93%	97.95%
7.	Enhanced DAE 2	LightGBM	97.97% (Enhanced Hybrid Model 2)	0.12%	97.90%	97.95%	97.96%
8.	Enhanced DAE 3	LightGBM	97.99% (Enhanced Hybrid Model 3)	0.10%	97.90%	97.93%	97.96%
9.	Enhanced DAE 4	LightGBM	98.06% (Enhanced Hybrid Model 4)	0.06%	98.05%	98.06%	98.04%

The bar graph that illustrates the detection loss comparison for ISCX-URL is shown in Figure 4.35.

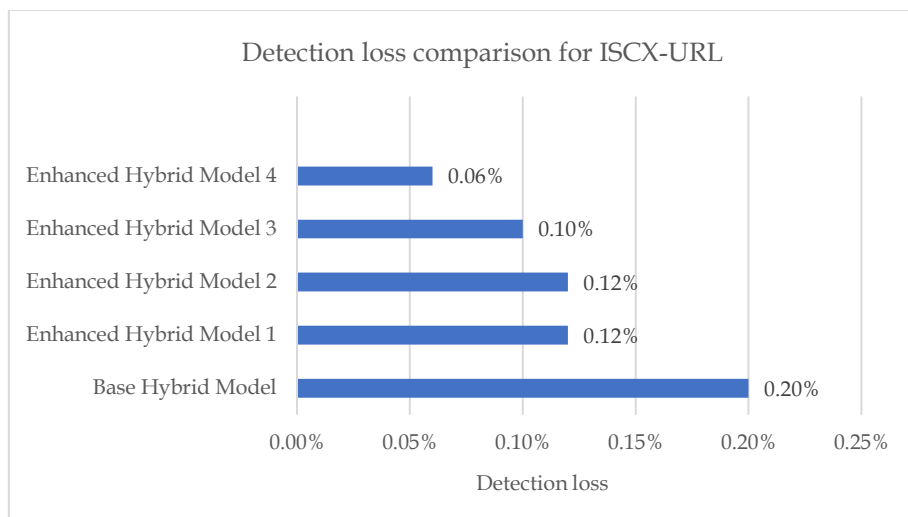


Figure 4.35: The detection loss comparison for ISCX-URL is shown in graphical bar

The bar graph that illustrates the detection rate comparison for ISCX-URL is shown in Figure 4.36.

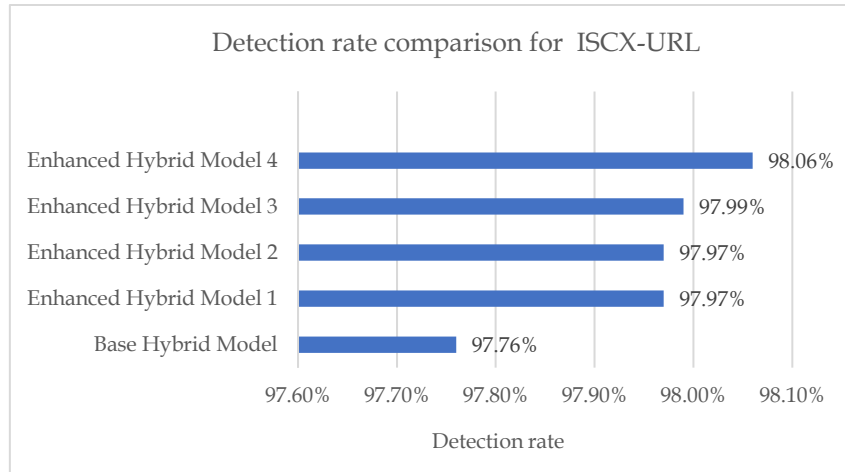


Figure 4.36: The detection rate comparison for ISCX-URL is shown in graphical bar

On notifying the performance graphs in Figure 4.35 and Figure 4.36, the enhanced hybrid models outperform our base hybrid model by obtaining higher detection rate and minimum detection loss. The enhanced hybrid model 4 outperforms slightly the other three enhanced hybrid models by securing highest detection rate and lowest detection loss.

Table 4.46 shows our proposed model results for the CIC-IDS with different existing schemes

Table 4.46: Proposed model results for the CIC-IDS with different existing schemes

No	Features selection	Classifier	Detection rate	Detection loss	Accuracy	Precision	F1score
1.	-	DBN	96.67 % (Manimurugan et al., 2020)	-	95.21%	97.34%	97%
2.	Naïve Bayes	SVM	98.92% (Gu and lu, 2021)	-	-	-	-
3.	-	RF	99% (Lopez et al., 2019)	-	99%	99%	-
4.	LDA	RF	98.% (Attak et al., 2018)	-	-	98.00%	98.00%
5.	Correlation	KNN	98.72% (Santikellur et al., 2019)	-	98.12%	98.17%	98.15%-
6.	DAE	LightGBM	99.86% (Ayubkhan et al., 2022) (Base Hybrid Model)	0.15%	99.84%	99.85%	99.84%
7.	Enhanced DAE 1	LightGBM	99.91% (Enhanced Hybrid Model 1)	0.12%	99.91%	99.89%	99.90%
8.	Enhanced DAE 2	LightGBM	99.92% (Enhanced Hybrid Model 2)	0.11%	99.89%	99.90%	99.91%
9.	Enhanced DAE 3	LightGBM	99.92% (Enhanced Hybrid Model 3)	0.11%	99.90%	99.90%	99.91%
10.	Enhanced DAE 4	LightGBM	99.94% (Enhanced Model Hybrid 4)	0.09%	99.92%	99.91%	99.92%

The bar graph that illustrates the detection loss comparison for CIC-IDS is shown in Figure 4.37.

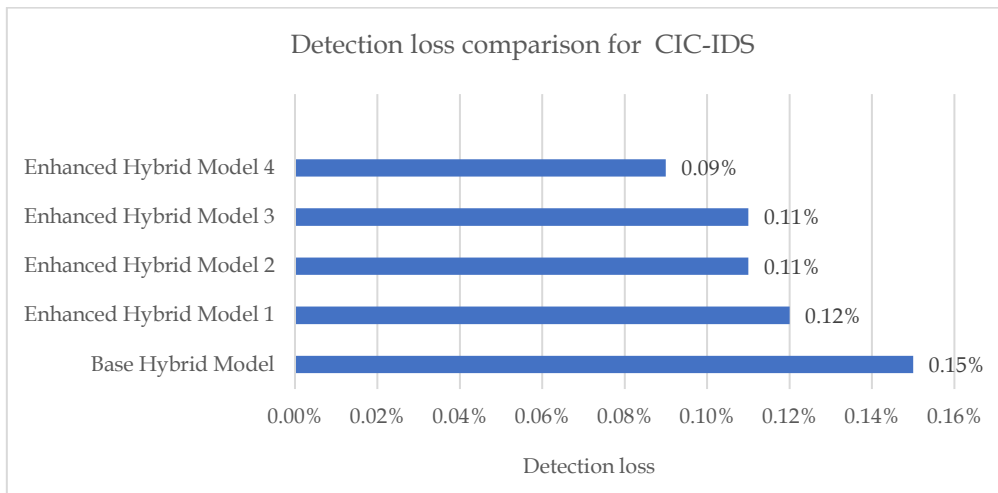


Figure 4.37: The detection loss comparison for CIC-IDS is shown in graphical bar

The bar graph that illustrates the detection rate comparison for CIC-IDS is shown in Figure 4.38.

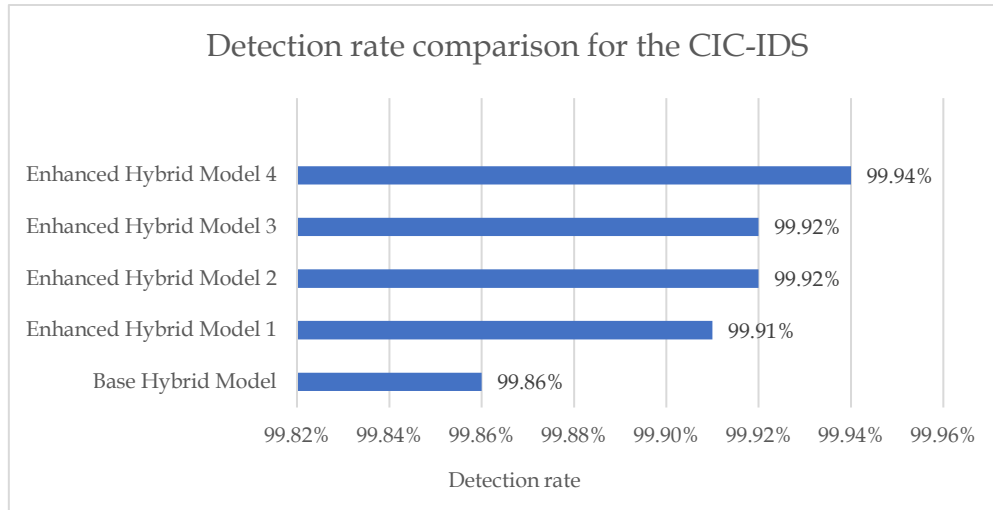


Figure 4.38: The detection rate comparison for CIC-IDS is shown in graphical bar

On notifying the performance graphs in Figure 4.37 and Figure 4.38, the enhanced hybrid models outperform our base hybrid model by obtaining higher detection rate and minimum detection loss. The enhanced hybrid model 4 outperform slightly the other three enhanced hybrid models by securing highest detection rate and lowest detection loss

Table: 4.47 shows our proposed model results for the ISCX-TOR with different existing systems

Table 4.47: Proposed models results for ISCX-TOR with different existing schemes

No	Features selection	Classifier	Detection rate	Detection loss	Accuracy	Precision	F1score
1.	Extra tree classifier	DCNN	86% (Lashkari et al., 2020)	86%	86%	86%	86%
2.	Data augmentation	CNN	86.94% (Shapira et al., 2021)	-	86.94%	-	64.93%
3.	IG	RF	97.1% (Zhou et al., 2020)	-	-	97.1%	97.1%
4.	Genetic algorithm	RNN LSTM	96% (Priya et al., 2021)	-	97%	98%	97%
5.	Apriori algorithm	Multi-Relative Entropy	91% (Yan et al., 2022)	-	91%	91%	-
6.	DAE	LightGBM	97% (Ayubkhan et al., 2022) (Base Hybrid Model)	0.10%	96.96%	97.11%	97.00%
7.	Enhanced DAE 1	LightGBM	98.12% (Enhanced Hybrid Model 1)	0.04%	98.03%	98.05%	98.08%
8.	Enhanced DAE 2	LightGBM	98.12% (Enhanced Hybrid Model 2)	0.04%	98.07%	98%	98.06%
9.	Enhanced DAE 3	LightGBM	98.11% (Enhanced Hybrid Model 3)	0.05%	98.07%	98.10%	98.10%
10.	Enhanced DAE 4	LightGBM	98.14% (Enhanced Hybrid Model 4)	0.02%	98.12%	98.12%	98.13%

The bar graph that illustrates the detection loss comparison for ISCX-TOR is shown in Figure 4.39.

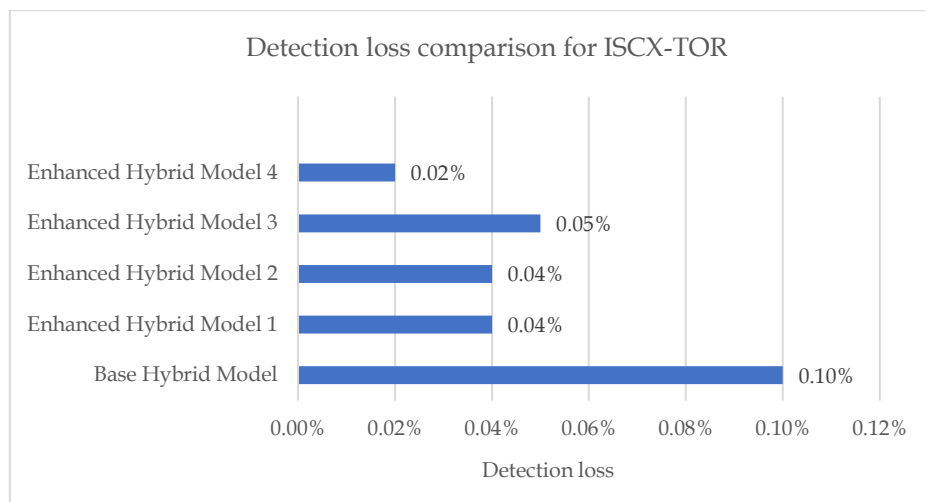


Figure: 4.39: The detection loss comparison for ISCX-TOR is shown in graphical bar

The bar graph that illustrates the detection rate comparison for ISCX-TOR is shown in Figure 4.40.

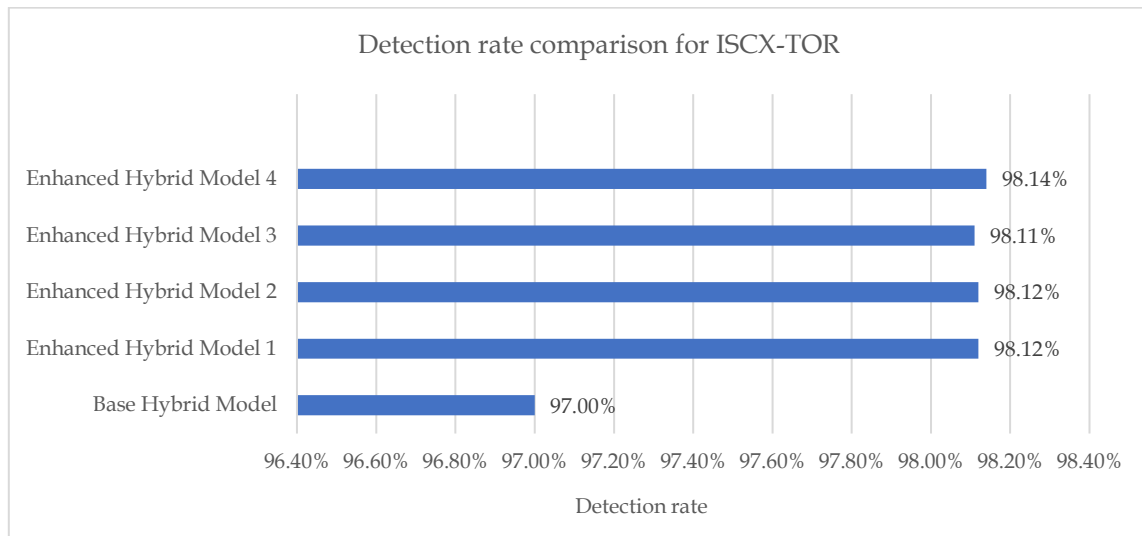


Figure 4.40: The detection rate comparison for ISCX-TOR is shown in graphical bar

On notifying the performance graphs in Figure 4.39 and Figure 4.40, the enhanced hybrid models outperform our base hybrid model by obtaining higher detection rate and minimum detection loss. The enhanced hybrid model 4 outperforms slightly the other three enhanced hybrid models by securing highest detection rate and lowest detection loss.

Table 4.48 shows our proposed model results for UNSW-NB with different existing schemes.

Table 4.48: Proposed models results for UNSW-NB with different existing schemes

No	Features selection	Classifier	Detection rate	Detection loss	Accuracy	Precision	F1score
1.	-	ensemble	91.8% (Hsu et al., 2019)	-	93.2%	91.7%	-
2.	Stacked AE	SoftMax	89.13% (Khan et al., 2019)	-	89.7%	89%	89%
3.	Naïve Bayes	SVM	93.75% (Gu and lu, 2021)	-	-	-	-
4.	-	Ensemble	86.40% (Baig et al., 2017)	-	86.74%	93.38%	89.94%
5.	-	DNN	78.40% (Yan et al., 2018)	-	94.40%	72.50%	82.00%
6.	DAE	LightGBM	96.11% (Ayubkhan et al., 2022) (Base Hybrid Model)	0.96%	96.00%	96%	95.99%
7.	Enhanced DAE 1	LightGBM	97.36% (Enhanced Hybrid Model 1)	0.78%	97.32%	97.31%	97.33%
8.	Enhanced DAE 2	LightGBM	97.34% (Enhanced Hybrid Model 2)	0.74%	97.33%	97.32%	97.33%
9.	Enhanced DAE 3	LightGBM	97.35% (Enhanced Hybrid Model 3)	0.75%	97.31%	97.33%	97.34%
10.	Enhanced DAE 4	LightGBM	97.39% (Enhanced Hybrid Model 4)	0.70%	97.35%	97.36%	97.34%

The bar graph that illustrates the detection loss comparison for UNSW-NB is shown in Figure 4.41.

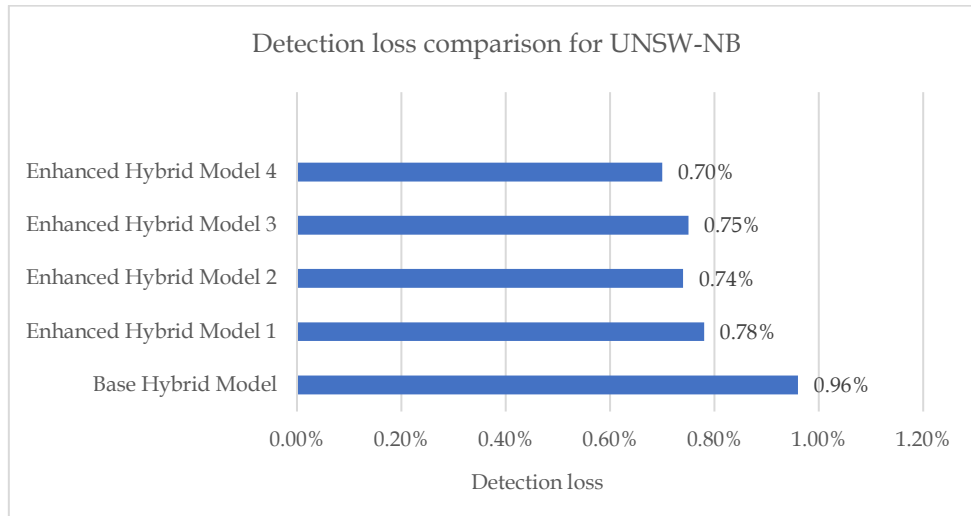


Figure 4.41: The detection loss comparison for UNSW-NB is shown in graphical bar

The bar graph that illustrates the detection rate comparison for UNSW-NB is shown in Figure 4.42.

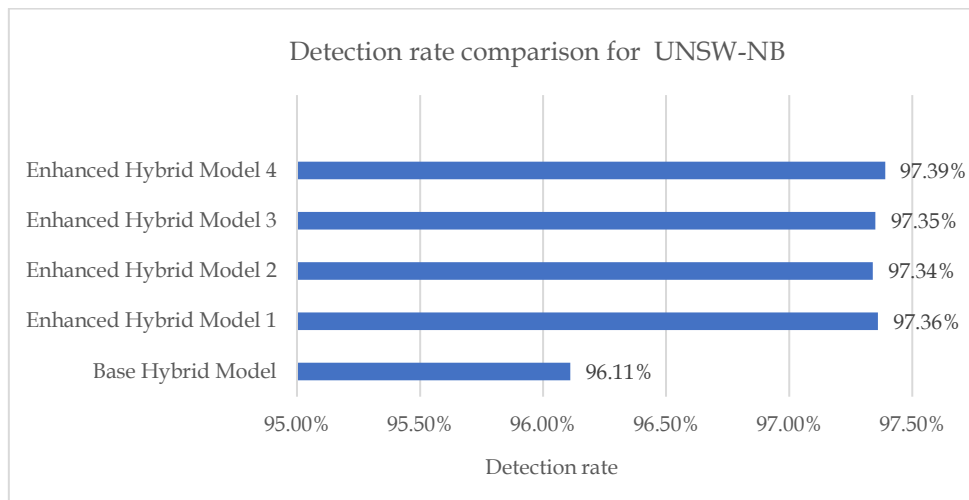


Figure 4.42: The detection rate comparison for UNSW-NB is shown in graphical bar

On notifying the performance graphs in Figure 4.41 and Figure 4.42, the enhanced hybrid models outperform our base hybrid model by obtaining higher detection rate and minimum detection loss. The enhanced hybrid model 4 outperforms slightly the other three enhanced hybrid models by securing highest detection rate and lowest detection loss.

Table 4.49 shows our proposed models results for BoT-IoT with different existing schemes

Table 4.49: Proposed models results for BoT-IoT with different existing schemes

No	Features selection	Classifier	Detection rate	Detection loss	Accuracy	Precision	F1score
1.	Manually selected	Random forest	99.80% (Ullah et al., 2020)	-	99.90%	99.75%	99.80%
2.	Random forest regressor	KNN	99% (Alsamiri et al., 2019)	-	99%	99%	99%
3.	Manual selection	Ensemble	96.99% (Ferrag et al., 2020)	-	-	-	-
4.	Manual selection	CNN	88.30% (Susilo et al., 2020)	-		91.27%	
5.	Block chain framework	Bidirectional LSTM	99.79 % (Alkadi et al., 2021)	-	98.91%	-	-
6.	DAE	LightGBM	99.91% (Ayubkhan et al., 2022) (Base Hybrid Model)	0.32%	99.91%	99.92%	99.91%
7.	Enhanced DAE 1	LightGBM	99.95% (Enhanced Hybrid Model 1)	0.30%	99.93%	99.91%	99.93%
8.	Enhanced DAE 2	LightGBM	99.95% (Enhanced Hybrid Model 2)	0.30%	99.92%	99.91%	99.93%
9.	Enhanced DAE 3	LightGBM	99.96% (Enhanced Hybrid Model 3)	0.28%	99.91%	99.93%	99.94%
10.	Enhanced DAE 4	LightGBM	99.98% (Enhanced Hybrid Model 4)	0.20%	99.96%	99.95%	99.96%

The bar graph that illustrates the detection loss comparison for BoT-IoT is shown in Figure 4.43.

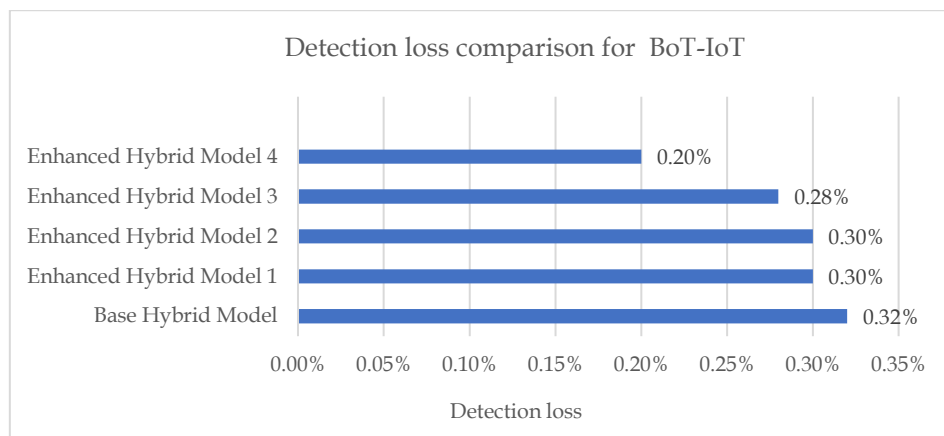


Figure 4.43: The detection loss comparison for BoT-IoT is shown in graphical bar

The bar graph that illustrates the detection rate comparison for BoT-IoT is shown in Figure 4.44.

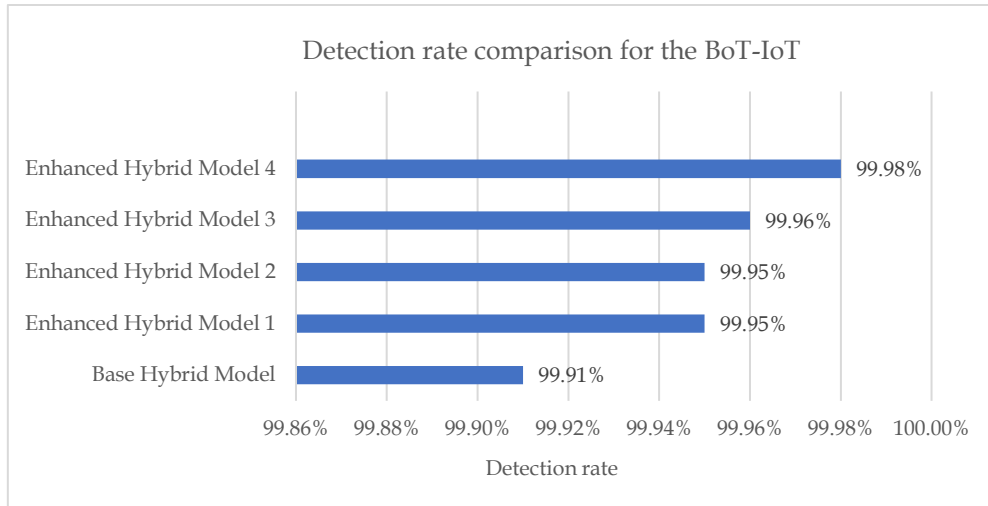


Figure 4.44: The detection rate comparison for BoT-IoT is shown in graphical bar

On notifying the performance graphs in Figure 4.43 and Figure 4.44, the enhanced hybrid models outperform our base hybrid model by obtaining higher detection rate and minimum detection loss. The enhanced hybrid model 4 outperforms slightly the other three enhanced hybrid models by securing highest detection rate and lowest detection loss.

4.5 Overview of the classification results

All the proposed four strategies have its unique standard mathematical algebraic properties in removing the deviations and enhancing the adversarial robustness and representation capacity of our model.

All the four strategies work out very well with DAE i.e. gaussian noise in the DAE, in suppressing the deviations and enhancing the quality of the patterns extracted. All the four strategies outperform our base hybrid model in extracting the useful, continuous complete core

structure of the network traffic. All the four strategies are simple, adaptable, and compatible to use and function with gaussian noise in the DAE.

Among the four proposed strategies, the fourth strategy, enhanced DAE 4 i.e. generative denoising autoencoder, as a generative model it outperforms the other three stochastic and deterministic models since the other three models encode only the discrete real vector values in the latent space structure, the fourth strategy outputs an entire probability distribution of the network traffic, which is full, regular, complete and continuous in nature and gives the more meaningful and quality enriched structure and patterns, that outperforms the other three enhanced DAE models.

The enhanced DAE models are associated with LightGBM classifier and develop four enhanced hybrid models which give higher detection score with very minimum detection loss. For ease of understanding, we present the experimental results on each of the datasets (Tables 4.3, 4.5, 4.7, 4.9, 4.11) before comparing our results with other work (Tables 4.42 - 4.49). Notice that, our proposed schemes achieve a detection rate of at least 96% with a detection loss of at least 0.96% for all eight different datasets. Moreover, refer to Tables (4.42 - 4.49), our proposed schemes outperform other existing schemes against eight different commonly used datasets.

The performance of our proposed schemes against different datasets show that our proposed schemes are lighter and yet achieve good detection rate with minimal detection loss due to the enhanced DAE and LightGBM i.e. due to the proposed novel regularization strategies that can remove deviations and give complete regularization along with gaussian

noise to extract the important low-dimensional hidden patterns lying in the dataset and histogram optimized leaf-wise strategy that can classify the samples using histogram bins of the hidden patterns with higher information gain in short training time.

It is to be noticed from the Tables (4.42 - 4.49), the enhanced hybrid models outperform the base hybrid model by securing higher detection rate with minimum detection loss. Among the four enhanced hybrid models, the enhanced hybrid model 4, outperforms the other three enhanced hybrid models by obtaining highest detection rate with lowest detection loss against all the eight benchmark datasets.

The main functionalities of the DAE and the four enhanced DAE models are shown in Table 4.50.

DAE	Enhanced DAE 1	Enhanced DAE 2	Enhanced DAE 3	Enhanced DAE 4
Contains gaussian noise in the input layer	Contains jacobian gradient norm on the encoder layer along with gaussian noise in the input layer	Contains iterative thresholding function on the encoder layer along with gaussian noise in the input layer	Contains data pairwise similarity weight on the encoder layer along with gaussian noise in the input layer	Contains approximated standard normal distribution with inference strategy on the encoder layer along with gaussian noise in the input layer
Ensures partial regularization on the encoder layer to minimize the deviations during transformation and extract salient patterns	Ensures complete regularization on the encoder layer along with gaussian noise in the input layer	Ensures complete regularization on the encoder layer along with gaussian noise in the input layer	Ensures complete regularization on the encoder layer along with gaussian noise in the input layer	Ensures complete regularization on the encoder layer along with gaussian noise in the input layer
Ensures partial robustness by partially corrupting the portion of input samples using gaussian noise	Ensures complete robustness by minimizing the partial derivatives in removing the distortions along with gaussian noise	Ensures complete robustness by sparsification of the weight matrix in removing the deviations along with gaussian noise	Ensures complete robustness by taking the most weighted similar points in each cluster using the data pairwise similarity weight.	Ensures complete robustness by taking the entire probability distribution of the input network traffic by enforcing the mean and variance to be the closer to the (standard normal) which is to be continuous in nature a
Encodes discrete real latent vector which gives the partial continuous and completeness.	Encodes discrete real latent vector which gives more than partial continuous and completeness	Encodes discrete real latent vector which gives more than partial continuous and completeness	Encodes discrete real latent vector which gives more than partial continuous and completeness	Encodes continuous latent values (close to zero mean and unit variance to ensure full completeness and regularity).

Table 4.50: The main functionalities of the DAE and the four enhanced DAE models

4.5.1 Classification results against minority classes

It is commonly known that the commonly used intrusion detection dataset is unbalanced where most of the samples are considered benign and only a minority of the samples are considered as an intrusion.

Tables (4.4, 4.6, 4.8, 4.10, 4.12) show the performance of our proposed schemes against different unbalanced classes of eight different datasets. Notice that the detection rate of our proposed schemes ranges from 93.33% to 99.98% which demonstrates the effectiveness of our proposed schemes to detect all classes including the minority classes. i.e. 99.94% for DoS, 95.46% for MITM, 97.13% for Scan, 99.98% for Reconnaissance, 93.33% for Theft, 99.52% for DoS Golden Eye, 99.73% for DoS Slow Http Test, 99.64% for DoS Slow Loris, 96.35% for Heartbleed, 99.14% for Brute Force, 99.90% for Port Scan, 98.95% for Ping Scan, 96.65% for Malware, 95.36% for Phishing, 97.70% for Spam, 93.79% for Tor, 97.12% for Attack. It is to be noticed that the enhanced hybrid model 4 achieves the higher detection rate for the minority category samples without any oversampling technique.

The bar graph that illustrates the detection rate of the minority classes for the benchmark datasets as shown in Figure 4.45.

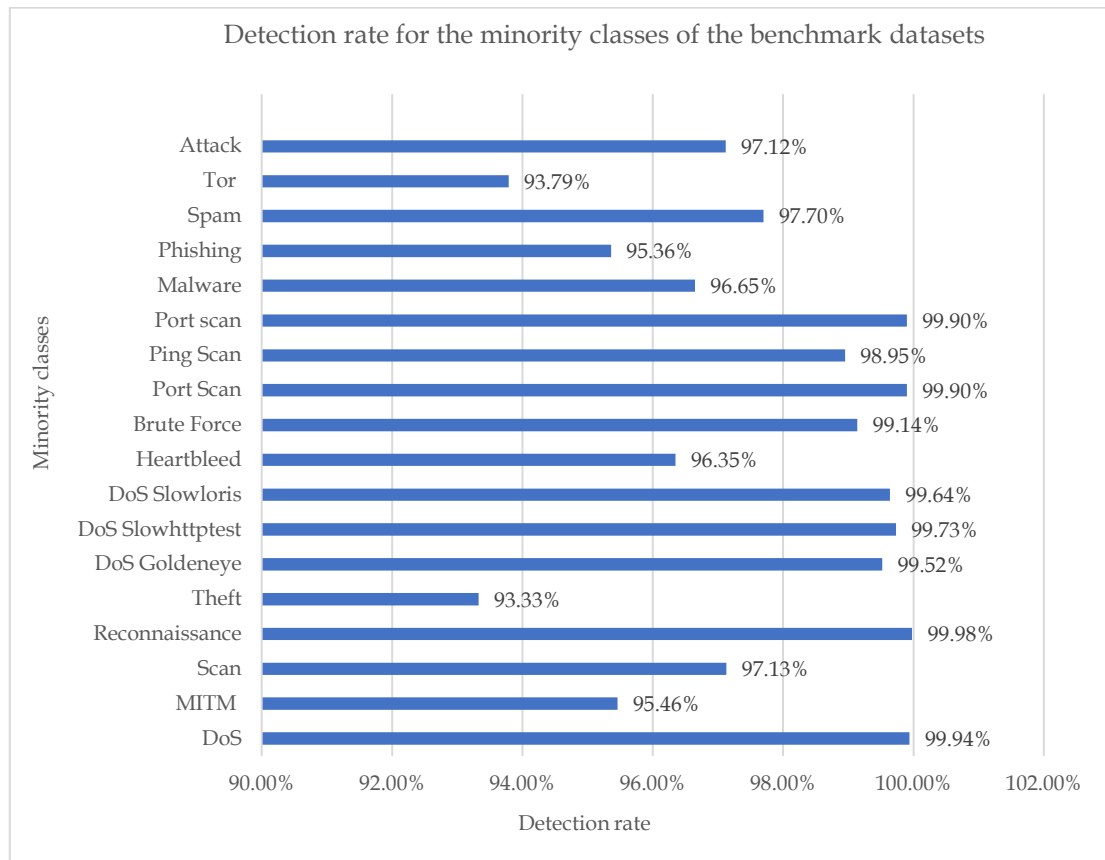


Figure 4.45: The detection rate of the minority classes is shown in graphical bar

The enhanced DAE models enhance the learning capacity of the model and are combined with LightGBM classifier to enhance the predictive capacity of the model. The combined efficiency of the two strategies (Enhanced DAE + LightGBM) are very sufficient to deal with the imbalanced nature of the datasets. The performance of our proposed schemes against unbalanced minority classes demonstrate the effectiveness of our model to detect all classes including any minority class. It is clear that our models are free from overfitting or underfitting issues and are not biased towards any classes and give optimal solution. The proposal has

higher learning and predictive capacity that optimizes the generalization capacity of our IDS system.

The proposed models can be deployed in any industrial sector to ensure secure data transmission i.e. proper data security, whereas the network traffic of those are always mixed with perturbations, where the IDS models are confused by those perturbations, learnt and extract deviated patterns and make wrong predictions. The enhanced DAE which is highly robust to noise and corruptions can be applied to remove those distortions and deviations in the learnt patterns, i.e. structure by making the model clear in its learning and prediction task. Besides, the existing models are very slow in train the features (samples) and learn the network traffic, whereas the LightGBM, which is very much faster, lighter and efficient can be applied to train the security system to learn the network traffic and take corrective decisions. Moreover, the industrial network traffics are very high-dimensional, perturbed and imbalanced in nature, where the combined strategies are very much adoptable and reliable to any network traffic dimensions.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

The conventional IDS models are highly vulnerable to the distortions in the network traffic. To overcome this, we proposed an IDS consisting of DAE and LightGBM. The DAE, which is robust to the distortions are applied to remove the distortions by extracting the hidden features. The LightGBM, which is lighter and efficient are applied to classify the samples. Though the proposed base hybrid model obtained better detection rate, still there are deviations in the latent structure due to the hidden distortions in the network traffic. To remove the distortions and eliminate the deviations, the DAE is enhanced to be the enhanced DAE models. The four enhanced DAE models eliminated the deviations in the latent structure using its own unique mathematical algebraic properties. The patterns extracted from the enhanced DAE models are associated with LightGBM and form four enhanced hybrid models. The effectiveness of the proposed models is evaluated using the standard benchmark datasets. The combined strategy reduced the detection loss and increased the detection rate of our IDS system against all the eight benchmark datasets. More importantly, the proposed models have higher generalization capacity which can deal with the imbalanced nature of the high-dimensional network traffic by securing good DR for the minority class samples without any oversampling technique. The proposed models can be deployed as real time models for any industrial network traffic.

5.2 Future work

The evaluation datasets are in structured type i.e. in CSV format. Our proposed models combining enhanced DAE and LightGBM work very well on structured data types. In future any unstructured cybersecurity datasets can be used, if any, to evaluate the performance and efficiency of the models and see how the models react with those data types. The unstructured data types such as temporal and sequential formats can be explored in future. At present, dense connected network layered architecture is used to design and evaluate our IDS models. In order to support unstructured data types, different architectures such as recurrent and convolution network layers can be used to design and explore our models as our future work. DAE is an enhancement over AE by the insertion of gaussian noise in the input layer of the AE. Enhanced DAE is an enhancement over DAE by the insertion of proposed additional novel regularization strategies on the encoder activation of the DAE. The future work can be done to see if any additional regularization strategies can be used and added in the decoder activation of our enhanced DAE to further reduce the reconstruction cost and make it the most enhanced version and more and more number of updated, modern, real world network traffic category datasets can be used to evaluate our proposed models.

REFERNCES

- Awad, M. and Khanna, R., 2015. Machine Learning. In: *Efficient Learning Machines*. Apress, Berkeley, CA, pp.1-18, https://doi.org/10.1007/978-1-4302-5990-9_1.
- Ahmar, I. et al., 2018. Performance Comparison of Support Vector Machine, Random Forest and Extreme Learning Machine for Intrusion Detection. *IEEE Access*, 6, pp.33789 – 33795.
- Ahmim, A. et al., 2019. A Novel Hierarchical Intrusion Detection System Based on Decision Tree and Rules-Based Models. In: *International Conference on Distributed Computing in Sensor Systems*, IEEE, pp. 228-233.
- Ainurrochman et al., 2021. Ensemble Methods Classifier Comparison for Anomaly Based Intrusion Detection System on CIDDS-002 Dataset. In: *13th International Conference on Information & Communication Technology and System (ICTS)*, IEEE, 21465766, pp. 62 – 67.
- Aksu, D., Üstebay, S., Aydin, M. A., and Atmaca, T., 2018. Intrusion Detection with Comparative Analysis of Supervised Learning Techniques and Fisher Score Feature Selection Algorithm. In: Czachórski, T, et.al., (eds). *Computer and Information Sciences*, Springer, 935, pp. 141-149.
- Ambusaidi, M. A. et al., 2014. A Novel Feature Selection Approach for Intrusion Detection Data Classification. In: *International Conference on Trust, Security and Privacy in Computing and Communications*, <https://doi.org/10.1109/TrustCom.2014.15>.
- Altamira, J. and Alsubhi, k., 2019. Internet of Things Cyber Attacks Detection using Machine Learning. *International Journal of Advanced Computer Science and Applications*, 10, pp. 627-634.
- Amangele, P. et al., 2019. Hierarchical Machine Learning for IoT Anomaly Detection in SDN. In: *Proceedings of the International Conference on Information Technologies*. IEEE, <https://doi.org/10.1109/InfoTech.2019.8860878>.
- Abdulhameed, R. et al., (2019). Features Dimensionality Reduction Approaches for Machine Learning Based Network Intrusion Detection. *Electronics*, 8(3), 322, <https://doi.org/10.3390/electronics8030322>.
- Aygun, R. C. and Yavuz, A.G., 2017. Network Anomaly Detection with Stochastically Improved Autoencoder Based Models. *International Conference on Cyber Security and Cloud Computing*, IEEE, <https://doi.org/10.1109/CSCloud.2017.39>.
- Abusitta, et al., 2019. A Deep learning Approach for Proactive Multi-Cloud Cooperative Intrusion Detection System. *Future Generation Computer System*, 98(3A), pp. 308-318.
- Ahamad, M. et. al., 2009. Emerging Cyber Threats Report for 2009. *Georgia Tech Information Security Center (GTISC)*, 34, pp. 1–9.

Alsaedi, M., Ghaleb, F. A., Saeed, F., Ahmad, J., and Alasli, M., 2022. Cyber Threat Intelligence-Based Malicious URL Detection Model Using Ensemble Learning. *Sensors*, 22(9), 3373, <https://doi.org/10.3390/s22093373>.

Alkahtani, H. and Aldhyani, T. H., 2021. Intrusion detection system to advance internet of things infrastructure-based deep learning algorithms. *Complexity*, 2021, pp.1-18, <https://doi.org/10.1155/2021/5579851>.

Attak, H., Combalia, M., Gardikis, G., and Gaston, B., 2018. Application of distributed computing and machine learning technologies to cybersecurity. *In: The Conference on Artificial Intelligence and Cybersecurity*, pp. 1-13.

Alsamiri, J. and Alsubhi, K., 2019. Internet of Things cyber-attacks detection using machine learning. *Int J Adv Comput Sci Appl*, 10(12), pp. 627-634.

Alkadi, O., Moustafa, N., Turnbull, B., Choo, R. K. K., 2021. A Deep Blockchain Framework-enabled Collaborative Intrusion Detection for Protecting IoT and Cloud Networks. *IEEE Internet of Things Journal*, 8(12), pp. 9463 – 9472.

Bansal, A. and Kaur, S., 2019. Data Dimensionality Reduction (DDR) Scheme for Intrusion Detection System Using Ensemble Standalone Classifiers. *In book: Advances in Computing and Data Sciences*, pp. 436-451. https://doi.org/10.1007/978-981-13-9939-8_39.

Baldi, P. and Hornik, K., 1989. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1), pp. 53-58.

Banks, L.D. and Fienberg, S. E., (2003). Data Mining, Statistics. *Encyclopedia of Physical Science and Technology*, 3, pp. 247-261.

Bergstra, J. S. et al., 2011. Algorithms for hyper-parameter optimization, *In: Advances in neural information processing systems*, pp. 2546–2554.

Bergstra, J. S. and Bengio, Y., 2012. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13, pp. 281-305.

Barreca, D. M., 2001. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems, *ACM SIGKDD Explorations Newsletter*, 3(1), pp. 27–32.

Blumensath, T. and Davies, M.E., 2008. Iterative Thresholding for Sparse Approximations. *Journal of Fourier Analysis and Applications*, 14 (6), pp. 629-654.

Baig, M. M., Awaisa, M. M. and Alfay, E. S. M., 2017. A multi-class cascade of artificial neural network for network intrusion detection. *J Intell Fuzzy Syst*, 32(4), pp. 2875- 2883.

Chowdhury, S., Liang, B. and Tizghadam, A., 2019. Explaining Class-of-Service Oriented Network Traffic Classification with Super Features. *In: Proceeding on Big Data, Machine Learning and Artificial Intelligence for Data Communications, ACM*, pp. 29-34.

- Cuautla, et al., 2020. Synthetic Minority Oversampling Technique for Optimizing Classification Tasks in Botnet and Intrusion-Detection System Datasets. *Applied Sciences*, 10(3), 794. <https://doi.org/10.3390/app10030794>.
- Catak, F. O. and Mustacoglu, A. F., 2019. Distributed denial of service attack detection using autoencoder and deep neural networks. *Journal of Intelligent & Fuzzy Systems*, 37, pp. 3969-3979.
- Choi, et. al., 2019. Unsupervised learning approach for network intrusion detection system using autoencoders, *The journal of Supercomputing*, 75, pp. 5597–5621.
- Chawla, N.V. et al. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16(1), pp. 321-357.
- Cortes, C. and Vapnik, V., 1995. Support- Vector Networks. *Machine learning, Springer*, 20(3), pp. 273-297.
- Chen, T. and Guestrin, C., 2016. XGBoost: A Scalable Tree Boosting System. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM*, pp. 785-794.
- Coates, A., Ng, A. and Lee, H., (2011). An analysis of single-layer networks in unsupervised feature learning. *Journal of Machine Learning Research*, 15, pp. 215-223.
- Cover, T. and Hart, P., (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1), pp. 21–27.
- Dozat, T. 2016. Incorporating Nestervov Momentum into Adam, *ICLR*.
- Denning, D. E. 1987. An intrusion-detection model. *IEEE Transactions on software engineering*, 13(2), pp. 222-232.
- Depren, O. et al., 2005. An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. *Expert systems with Applications*, 29(4), pp. 713–722.
- Dash, M. and Liu, H., 1997. Feature selection for classification. *Intelligent Data Analysis*, 1(4), pp. 131-156. [https://doi.org/10.1016/S1088-467X\(97\)00008-5](https://doi.org/10.1016/S1088-467X(97)00008-5).
- Dawid, A.P. 2011. Posterior Model Probabilities, *Philosophy of Statistics*, 7, pp. 607-630.
- Eskin, E. et.al., 2002. A Geometric Framework for Unsupervised Anomaly Detection. In: Barbara, D., Jajo ia, S., (eds). Applications of Data Mining in Computer Security. *Advances in Information Security, Springer*, 6, https://doi.org/10.1007/978-1-4615-0953-0_4.
- Eidous, O.M. and Shareefa, R.A., 2019. New approximations for standard normal distribution function. *Communication in Statistics: Theory and Methods*, 49(137), pp. 1-18.

- Fahimen, et al., 2018. Anomaly based Intrusion detection using deep neural networks. *International journal of digital content technology and its applications*, 12, pp. 70-82.
- Ford, W., 2015. Vector and Matrix Norms. *Numerical Linear Algebra with Applications*, pp. 119-144. <https://doi.org/10.1016/B978-0-12-394435-1.00007-7>.
- Fornasier, M. and Rauhut, H., 2008. Iterative thresholding algorithms. *Applied and Computational Harmonic Analysis*, 25(2), pp. 187-208.
- Fayyad, U. et al., 1996. From data mining to knowledge discovery in databases. *AI magazine*, 17(3), pp. 1-37.
- Fred, A.L. N., and Jain, A. K., 2006. Learning Pairwise Similarity for Data Clustering. *In: International Conference on Pattern Recognition, IEEE*, <https://doi.org/10.1109/ICPR.2006.754>.
- Ferrag, M. A., Maglaras, L. and Ahmim A., 2020. RDTIDS: Rules and Decision Tree-Based Intrusion Detection System for Internet-of-Things Networks. *Future Internet*. 12(3), 44, <https://doi.org/10.3390/fi12030044>.
- Ferraga, M.A., Maglaras, L., Mochatines, S., and Janicke, H., 2020. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 50, 102419, pp. 1-19.
- Gupta, G.P. and Kulariya, M., (2016). A Framework for Fast and Efficient Cyber Security Network Intrusion Detection using Apache Spark. *In: international conference on Advances in computing & communications, Elsevier*, pp. 824-831.
- Golik, P. et al., 2013. Cross-entropy vs. squared error training: a theoretical and experimental comparison. *In: INTERSPEECH*, pp. 1756– 1760.
- Gregor, K. and LeCun, Y., 2010. Learning fast approximations of sparse coding. *In: Proceedings of the International Conference on Machine Learning*, pp. 399–406.
- Gu, J., and Lu, S., 2021. An effective intrusion detection approach using SVM with Naïve Bayes feature embed-ding. *Comput Secur*, 103, 102158.
- Goodfellow, I. et al., 2009. Learning invariant features through local space contraction. *In: Advances in neural Information Processing System*, pp. 646–654.
- Hong, D. et al., 2017. Asymptotic Performance of PCA for High-Dimensional Heteroscedastic Data. *Journal of Multivariate Analysis*, 167, pp. 435-452.
- Hsu, Y. F., He, Z. Y., Tarutani, Y., and Matsuoka, M., 2019. Toward an Online Network Intrusion Detection System Based on Ensemble Learning. *International Conference on Cloud Computing, IEEE*, <https://doi.org/10.1109/CLOUD.2019.00037> .

- Hanson, S. and Hunt, R., 2005. A taxonomy of network and computer attacks. *Computers & Security*, 24(1), pp. 31–43.
- Hulse, J. V. et al. 2007. Experimental perspectives on learning from imbalanced data. *In: Proceedings of the International Conference on Machine Learning, ACM*, pp. 935–942.
- Hinton, G. E. and Salakhutdinov, R. R., 2006. Reducing the dimensionality of data with neural networks, *Science*, 313 (5786), pp. 504–507.
- Hornik, K. et al., 1989. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), pp. 359–366.
- Hindy, H., Brosset, D., Bayne, E., and Seams, A., 2020. A Taxonomy of Network Threats and the Effect of Current Datasets on Intrusion Detection Systems. *IEEE Access* , 8, pp. 2169-3536.
- He, W., Li, H., and Li, J., 2019. Ensemble features selection for improving Intrusion detection classification accuracy. *In: Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science, ACM*, pp. 28-33.
- Indyk, P. and Motwani, R., 1998. Approximate nearest neighbors: Towards removing the curse of dimensionality, *In: Proceedings of the symposium on Theory of Computing, ACM*, 8, pp. 604-613.
- Im, D.J. et al., 2018. Denoising criterion for variational auto-encoding framework. *In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 2059–2065.
- Jiang, S. et al., 2006. A clustering-based method for unsupervised intrusion detections. *Pattern Recognition Letter*, 27(7), pp. 802-810.
- Kotsiantis, S. B. et al., 2007. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160, pp. 3-24.
- Khan, F. A. et al., 2019. A Novel Two Stage Deep Learning Model for Efficient Network Intrusion Detection. *IEEE Access*, 7, pp. 30373 – 30385.
- KingRobert, A. P. and Eckersley, J., 2019. Bayesian Inferential Statistics I: *Basic Concepts, Statistics for Biomedical Engineers and Scientists*, pp. 71-90. <https://doi.org/10.1016/B978-0-08-102939-8.00013-X>.
- Koroniotis, N. et al. 2018. Towards the development of realistic botnet dataset in the Internet of Things for network forensic Systems. *Future Generation Computer Systems*, 100, pp. 779-796.
- Ke, G. et al. 2017. LightGBM: A highly efficient gradient boosting decision tree. *In: Proceedings of the 31st conference on Advances in Neural Information Processing Systems (NIPS 2017), ACM*, pp. 3149–3157.

- Kapil, D., Bansal, A., Priya, A., Mehra, N., and Joshi, A., 2019. Machine Learning Based Malicious URL Detection. *International Journal of Engineering and Advanced Technology*, 8(4), pp. 2249 – 8958.
- Lee, S. C. et al., 2001. Training a neural-network based intrusion detector to recognize novel attacks. *IEEE Transactions on systems, man, and Cybernetics: Systems and Humans*, 31(4), pp. 294-299.
- LeCun, Y. et al., 1998. Gradient-based learning applied to document recognition. *In: Proceedings of the IEEE*, 86(11), pp. 2278-2324.
- Labatut, V. and Cherifi, H., 2012. Evaluation of performance measures for classifiers comparison. *Ubiquitous Computing and Communication Journal*, 6, pp. 21-34.
- Lopes, O. I. et al. 2022. Effective network intrusion detection via representation learning: A Denoising Autoencoder approach. *Computer Communications*, 194, pp. 55-65.
- Liao, Y. and Vemuri, V. R., 2002. Use of K-Nearest Neighbor classifier for intrusion detection. *Computers & Security*, 21(5), pp. 439–448.
- Lashkari, A. H. et al., 2017. Characterization of Tor Traffic Using Time Based Features. *International Conference on Information System Security and Privacy, ACM*, pp. 253-262.
- Lashkari, H. A., Kaur, G. and Rahali, A., 2020. DIDarknet: A Contemporary Approach to Detect and Characterize the Darknet Traffic using Deep Image Learning. *In: 10th International Conference on Communication and Network Security, ACM, 2021*, pp. 1-13, <https://doi.org/10.1145/3442520.3442521>.
- Lopez, A. D., Mohan, P. A. and Nair, S., 2019. Network Traffic Behavioral Analytics for Detection of DDoS Attacks. *SMU Data Science Review*, 2(14), pp. 2-14. <https://scholar.smu.edu/datasciencereview/vol2/iss1/14>.
- Maas, A. L., Hannun, A. Y. and Ng, A.Y., 2013. Rectifier Nonlinearities Improve Neural Network Acoustic Models, *In: Proceedings of the International Conference on Machine Learning, JMLR*, 28.
- Mishra, P. et al. 2018. A Detailed Investigation and Analysis of using Machine Learning Techniques for Intrusion Detection. *IEEE Communications Surveys & Tutorials*, 21, pp. 686-728.
- Mitchell, T. M., 1997 Machine learning. Burr Ridge, IL: McGraw Hill, 45(37), pp. 870–877.
- Mamun M. S. I., Rathore M. A., Lashkari A. H., Stakhanova N., and Ghorbani A. A., 2016. Detecting Malicious URLs Using Lexical Analysis. In: Chen J., Piuri V., Su C., Yung M. (eds) *Network and System Security. NSS 2016. Lecture Notes in Computer Science*, vol 9955. Springer, Cham, pp. 467-482, https://doi.org/10.1007/978-3-319-46298-1_30.

- Mahajan, H. B. et al., 2020. Detecting HTTP Vulnerabilities in IoT-based Precision Farming Connected with Cloud Environment using Artificial Intelligence. *International Journal of Advanced Science and Technology*, 29, pp. 214 – 226.
- Moustafa, N. et al., 2018. An Ensemble Intrusion Detection Technique Based on Proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things. *IEEE Internet of Things Journal*, 6(3), pp. 4815 – 4830.
- Moustafa, N., Hu, J. and Slay, J., 2019. A holistic review of Network Anomaly Detection Systems: A comprehensive survey. *Journal of Network and Computer Applications*, 128, pp. 33 – 55.
- Moustafa, N. and Slay, J., 2015. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: *Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS)*, IEEE, pp. 1-6.
- Manimurugan, S. et al., 2020. Effective Attack Detection in Internet of Medical Things Smart Environment using a Deep Belief Neural Network. *IEEE Access*, 8, pp. 77396 –77404. <https://doi.org/10.1109/ACCESS.2020.2986013>.
- Nguyen, V. et al., 2018. Using Deep Learning Model for Network Scanning Detection. In: *International Conference on Frontiers of Educational Technologies*. pp. 117-121.
- Nilsson, N. J., 1998. Neural Networks. *Artificial Intelligence: A new Synthesis*, pp. 37-57, <https://doi.org/10.1016/B978-0-08-049945-1.50008-3>.
- Oliveira, N., Praça, I., Maia, E., and Sousa, O., 2021. Intelligent Cyber Attack Detection and Classification for Network-Based Intrusion Detection Systems. *Appl. Sci*, 11(4), 1674, <https://doi.org/10.3390/app11041674>.
- Peddabachigari, s. et al., 2007. Modeling intrusion detection system using hybrid intelligent systems. *Journal of Network and Computer Applications*, 30, pp. 114-132.
- Pedregosa, F. et al., 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, pp. 2825–2830.
- Priya, V. A., Sing, K. H., Prasad, C. S. M., and Sai, S. J. G., 2021. RNN-LSTM Based Deep Learning Model for Tor Traffic Classification. *Cyber-Physical Systems*, 9(1), pp. 25 -42.
- Panda, M. and Patra, M.R., 2007. Network intrusion detection using naive bayes. *International Journal of Computer Science and Network Security*, 7, pp. 258–263.
- Quinlan, J. R., 1986. Induction of decision trees. *Machine learning*. Springer, 1(1), pp. 81–106.

- Razdan, S. et al., 2021. Performance analysis of network intrusion detection systems using DT and naïve bayes algorithms. *In: International Conference for Convergence in Technology, IEEE*, 9(4), pp. 1-7.
- Quang, D., 2021. Evaluating machine learning algorithms for intrusion detection systems using the dataset CIDDS-002. *In: Proceedings of the 4th International Conference on Computer Science and Engineering, ACM*, pp. 112 -118. <https://doi.org/10.1145/3494885.3494906>.
- Ring, M., Wunderlich, S., Scheuring, D., Landes, D., and Hathi, A., 2019. A survey of network-based intrusion detection data sets. *Computers & Security*. 86, pp. 147-167.
- Ring, M., Wunderlich, S., Gruedl, D., Landes, D., and Hotho, A., (2017). Creation of Flow-Based Data Sets for Intrusion Detection. *Journal of Information Warfare*. 16(4), pp. 40-53, <https://www.jstor.org/stable/26504117>.
- Ring, M., Wunderlich, S., Gruedl, D., Landes, D., and Hotho, A., (2017). Flow-based benchmark data sets for intrusion detection. *In: Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS), ACPI*, pp. 361-369.
- Rumelhart, D. E., Hinton, G. E. and Williams, R. J., (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), pp. 533–536.
- Reddi, S. J., Kale, S. and Kumar, S., 2018. On the convergence of Adam and Beyond. *ICLR*.
- Rego, K. T. and Lupu, E. C., 2021. Jacobian Regularization for Mitigating Universal Adversarial Perturbations. In: Farkas, I., et.al., (eds). *Artificial Neural Networks and Machine Learning. Lecture Notes in Computer Science, Springer*, 12894, pp. 202-213.
- Sharda, R. et al., 2018. Business Intelligence, Analytics, and Data Science: A managerial perspective. *4th ed. Pearson*.
- Saqib, N. M., 2022. URL Filtering by Using Machine Learning. *International Journal of Computer Science and Network Security*. 22(8), pp. 275 – 279.
- Shapira, T. and Shavitt, Y., 2021. FlowPic: A Generic Representation for Encrypted Traffic Classification and Applications Identification. *IEEE Transactions on Network and Service Management*, 18(2), pp. 1218 – 1232.
- Snehal, A. et al., 2010. Intrusion detection system using support vector machine and decision tree. *International Journal of Computer Applications*, 3(3), pp. 40–43.
- Shukla, P. and Rai, R., 2017. Ara-mac: Attacker identification using logistic regression. In: *International Conference on Recent Innovations in Signal processing and Embedded Systems, IEEE*, pp. 124-128.
- Susilo, B. and Sari, R. F., 2020. Intrusion Detection in IoT Networks Using Deep Learning Algorithm. *Information*, 11(5), 279, <https://doi.org/10.3390/info11050279>.

- Santikellur, P., Haque, T., Zewairi, M. A., and Chakraborty, R., 2019. Optimized multi-layer hierarchical network intrusion detection system with genetic algorithms. *In: International Conference on new Trends in Computing Sciences, IEEE*, pp. 1-7.
- Sugiyama, M., Eidous, M. O. and Shareefa, R. A., 2019. New approximations for standard normal distribution function. *Communication in Statistics- Theory and Methods*, 49(137), pp. 1-18.
- Sharafaldin, I. et al., 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. *International Conference on Information Systems Security and Privacy*, pp. 108-116.
- Sornsuwit, P. and Jaiyen, S., 2019. A New Hybrid Machine Learning for Cybersecurity Threat Detection Based on Adaptive Boosting. *Applied Artificial Intelligence*, 33(1), pp. 462-482.
- Scott, S.L., 2004. A Bayesian paradigm for designing intrusion detection systems. *Computational Statistics & Data Analysis*, 45, pp. 69–83.
- Song, Y., Hyun, S. and Cheong, Y. G., 2021. Analysis of Autoencoders for Network Intrusion Detection. *Sensors*, 21(13), 4294, <https://doi.org/10.3390/s21134294>.
- Shone, N. et al., 2018. A Deep Learning Approach to Network Intrusion Detection, *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1), pp. 41-50.
- Shimada, T. et al., 2021. Classification from Pairwise Similarities and Unlabeled Data via Empirical Risk Minimization, *Neural Computation*, 33(5), pp. 1-35.
- Tsai, C. F. et al., 2009. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10), pp. 11994–12000.
- Thapa, N., Liu, Z., KC, D. B., Gokaraju, B., and Roy, K., 2020. Comparison of Machine Learning and Deep Learning Models for Network Intrusion Detection Systems. *Future Internet*, 2(10), 167, <https://doi.org/10.3390/fi12100167>.
- Tama, B. A. and Rhee, K. H., 2019. An in-depth experimental study of anomaly detection using gradient boosted machine. *Neural Comput & Applic*, 31(4), pp. 955–965.
- Ullah I. and Mahmoud Q. H., 2020. A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks. In: Goutte C., Zhu X. (eds) *Advances in Artificial Intelligence. Canadian AI 2020. Lecture Notes in Computer Science*, vol 12109. Springer, Cham, pp. 508-520, https://doi.org/10.1007/978-3-030-47358-7_52.
- Ullah, I. and Mahmoud, Q.H., 2020. A Two-Level Flow-Based Anomalous Activity System for IoT networks. *Electronics*, 9(3), 530.
- Ullah, S. et al., 2022. A New Intrusion Detection System for the Internet of Things via Deep Convolutional Neural Network and Feature Engineering. *Sensors*, 22(10), 3607, <https://doi.org/10.3390/s22103607>

- Vincent, P. et al., 2010. Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *Journal of Machine Learning Research*, 11, pp. 3371-3408.
- Vincent, P. et al. 2010. Extracting and composing robust features with denoising autoencoders. *In: Proceedings of the international conference on Machine learning, ACM*, pp.1096-1103. <https://doi.org/10.1145/1390156.1390294>.
- Verma, A. and Ranga, V., 2020. Machine learning based intrusion detection systems for IoT applications. *WirelPers Commun.* 111, pp. 2287-2310.
- Verwoerd, T. and Hunt, R., 2002. Intrusion detection techniques and approaches. *Computer Communications*, 25(15), pp. 1356–1365.
- Vinayakumar, M. et al. 2019. Deep Learning Approach for Intelligent Intrusion Detection System, *IEEE Access*, 7, pp. 41525 – 41550.
- Wu, et al., 2018. A Novel Intrusion Detection Model for a Massive Network Using Convolutional Neural Network, *IEEE Access*, 6, pp. 50850 – 50859.
- Wang, E. and Sun, J., 2015. Survey on distance metric learning and dimensionality reduction in data mining. *Data Mining and Knowledge Discovery*, 29(2), pp. 534-564.
- Wei, X., Ren, S., Li, Y., Wang, X. X., and Jin, M., 2021. Intrusion Detection Algorithm Based on SDA-ELM. In: Fujita, H., Selamat, A., Lin, J.CW., Ali, M. (eds) *Advances and Trends in Artificial Intelligence. From Theory to Practice. IEA/AIE 2021. Lecture Notes in Computer Science*, vol 12799, Springer, Cham. pp. 495–505, https://doi.org/10.1007/978-3-030-79463-7_42.
- Yin, C. et al., 2017. A deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access*, 5, pp. 21954 – 21961.
- Yulianto, A. et al., 2019. Improving AdaBoost-based Intrusion Detection System (IDS) Performance on CIC IDS 2017 Dataset. *In: International Conference on Data and Information Science. IOP Publishing.* 1192, <https://doi.org/10.1088/1742-6596/1192/1/012018>.
- Yan, J., Jin, D., Lee, C. W., and Liu, P. A., 2018. Comparative study of off-line deep learning-based network intrusion detection. *In: Tenth International Conference on Ubiquitous and Future Networks, IEEE*, pp. 299-304.
- Yan, H. et al., 2022. Bidirectional Statistical Feature Extraction Based on Time Window for Tor Flow Classification. *Symmetry*, 14(10), 2002, <https://doi.org/10.3390/sym14102002>.
- Zhang, J. et al., 2008. Random-Forests-Based Network Intrusion Detection Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 38(5), pp. 649-659.

Zhou, Y. et al., 2019. Building an Efficient Network Intrusion Detection System Based on Feature Selection and Ensemble Classifier. *Computer Networks*, 174, pp. 1-21, <https://doi.org/10.1016/j.comnet.2020.107247>.

Zarpelao, B. B. et al., 2017. A survey of intrusion detection in Internet of Things. *Journal of Network and Computer Applications*, 84, pp. 25-37.

Zhou, K., Wang, W., Wu, C., and Hu, T., 2020. Practical evaluation of encrypted traffic classification based on a combined method of entropy estimation and neural networks. *ETRI journal*, 42(3), pp. 311 -323.

LIST OF PUBLICATIONS

- Ayubkhan, S. A. H., Yap, W. S., and Morris, E., Performance Comparison of Gradient Boosting Frameworks for Intrusion Detection. *Wireless Personal Communications – Under Review*.
- Ayubkhan, S. A. H., Yap, W. S., Morris, E., and Rawthar, K. M. B., A practical intrusion detection system based on denoising autoencoder and LightGBM classifier with improved detection performance. *J Ambient Intell Human Comput* (2022). <https://doi.org/10.1007/s12652-022-04449-w>.
- Ayubkhan, S. A. H., Yap, W. S., Morris, E., and Rawthar, K. M. B., Enhanced Denoising Autoencoder and LightGBM for Improved Intrusion Detection. *Expert System with Applications – Under Review*.

APPENDIX

FEATURES OF THE DATASETS

CIC-IDS-2017

The features of the CICIDS dataset are listed in the below table

Features	Features	Features
Destination Port	Flow Duration	Total Fwd Packets
Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets
Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean
Fwd Packet Length Std	Bwd Packet Length Max	Bwd Packet Length Min
BwdPacketLength Mean	Bwd Packet Length Std	Flow Bytes/s
Flow Packets/s	Flow IAT Mean	Flow IAT Std
Flow IAT Max	Flow IAT Min	Fwd IAT Total
Fwd IAT Mean	Fwd IAT Std	Fwd IAT Max
Fwd IAT Min	Bwd IAT Total	Bwd IAT Mean
Bwd IAT Std	Bwd IAT Max	Bwd IAT Min
Fwd PSH Flags	Bwd PSH Flags	Fwd URG Flags
Bwd URG Flags	Fwd Header Length	Bwd Header Length
Fwd Packets/s	Bwd Packets/s	Min Packet Length
Max Packet Length	Packet Length Mean	Packet Length Std
Packet Length Variance	FIN Flag Count	SYN Flag Count
RST Flag Count	PSH Flag Count	ACK Flag Count
URG Flag Count	CWE Flag Count	ECE Flag Count
Down/Up Ratio	Average Packet Size	Avg Fwd Segment Size
Avg Bwd Segment Size	Fwd Header Length2	Fwd Avg Bytes/Bulk
Fwd Avg Packets/Bulk	Fwd Avg Bulk Rate	Bwd Avg Bytes/Bulk
Bwd Avg Packets/Bulk	Bwd Avg Bulk Rate	Subflow Fwd Packets
Subflow Fwd Bytes	Subflow Bwd Packets	Subflow Bwd Bytes
Init_Win_bytes_forward	Init_Win_bytes_backward	act_data_pkt_fwd
min_seg_size_forward	Active Mean	Active Std
Active Max	Active Min	Idle Mean
Idle Std	Idle Max	Idle Min
Label		

Features of CIC-IDS2017

ISCX-TOR 2016

The following table lists the features of the dataset, which are all numerical features, except the features, Source IP and Destination IP, that both are nominal

Features	Features	Features	Features
Source IP	Source Port	Destination IP	Destination Port
Protocol	Flow Duration	Flow Bytes/s	Flow Packets/s
Flow IAT Mean	Flow IAT Std	Flow IAT Max	Flow IAT Min
Fwd IAT Mean	Fwd IAT Std	Fwd IAT Max	Fwd IAT Min
Bwd IAT Mean	Bwd IAT Std	Bwd IAT Max	Bwd IAT Min
Active Mean	Active Std	Active Max	Active Min
Idle Mean	Idle Std	Idle Max	Idle Min
Label			

Features of ISCX-Tor 2016

UNSW-NB15

The features of the UNSW-NB 15 dataset are given in following table

Features	Datatype	Features	Datatype	Features	Datatype
srcip	nominal	sport	integer	dstip	nominal
dsport	integer	proto	nominal	state	nominal
dur	float	sbytes	Integer	dbytes	integer
sttl	integer	dttl	Integer	sloss	integer
dloss	integer	service	nominal	sload	float
dload	float	Spkts	integer	dpkts	integer
swin	integer	dwin	integer	stcpb	integer
dtcpb	integer	smeansz	integer	dmeansz	integer
trans_depth	integer	res_bdy_len	integer	sjit	float
djit	float	Sstime	Timestamp	ltime	timestamp
Sintpkt	float	dintpkt	Float	tcprrt	float
synack	float	ackdat	Float	is_sm_ips	binary
ct_state_ttl	integer	ct_flw_http_mthd	Integer	is_ftp_login	binary
ct_ftp_cmd	integer	ct_srv_src	integer	ct_srv_dst	integer
ct_dst_ltm	integer	ct_src_ltm	integer	ct_src_dport_ltm	integer
ct_dst_sport_ltm	integer	ct_dst_src_ltm	integer	label	

Features of UNSW-NB 15

CIDDS

The following table gives an overview of the attributes in the CIDDS- datasets (i.e. CIDDS-001 and 002)

Features	Data Type	Features	Data Type
Src IP	Nominal	Src Port	Numeric
Dest IP	Nominal	Dest Port	Numeric
Proto	Nominal	Date first seen	Nominal
Duration	Numeric	Bytes	Nominal
Packets	Numeric	Flags	Nominal
Class	Nominal	AttackType- label	Nominal
AttackID	Nominal	AttackDescription	Nominal

Features of CIDDS datasets

ISCX-URL2016

The following table shows the 80 numerical features of the ISCX-URL-2016

Features	Features	Features
Querylength	domain_token_count	path_token_count
avgdomaintokenlen	longdomaintokenlen	avgpathtokenlen
tld	charcompvowels	charcompacc
ldl_url	ldl_domain	ldl_path
ldl_filename	ldl_getArg	dld_url
dld_domain	dld_path	dld_filename
dld_getArg	urlLen	domainlength
pathLength	subDirLen	fileNameLen
this.fileExtLen	ArgLen	pathurlRatio
ArgUrlRatio	argDomanRatio	domainUrlRatio
pathDomainRatio	argPathRatio	executable
isPortEighty	NumberOfDotsinURL	ISIPAddressInDomainName
CharacterContinuityRate	LongestVariableValue	URL_DigitCount
host_DigitCount	Directory_DigitCount	File_name_DigitCount
Extension_DigitCount	Query_DigitCount	URL_Letter_Count
host_letter_count	Directory_LetterCount	Filename_LetterCount
Extension_LetterCount	Query_LetterCount	LongestPathTokenLength
Domain_LongestWordLength	Path_LongestWordLength	sub-Directory_Longest WordLength
Arguments_LongestWord Length	URL_sensitiveWord	URLQueries_variable
spcharUrl	delimiter_Domain	delimiter_path
delimiter_Count	NumberRate_URL	NumberRate_Domain
NumberRate_DirectoryName	NumberRate_FileName	NumberRate_Extension
NumberRate_AfterPath	SymbolCount_URL	SymbolCount_Domain
SymbolCount_Directoryname	SymbolCount_FileName	SymbolCount_Extension
SymbolCount_Afterpath	Entropy_URL	Entropy_Domain
Entropy_DirectoryName	Entropy_Filename	Entropy_Extension
Entropy_Afterpath	URL_Type_obf_Type	

Features of ISCX-URL2016

BoT-IoT

The features of the BoT-IoT dataset are given in table below. There are 36 numerical and 7 categorical features

Features	Datatype	Features	Datatype
pkSeqID	numeric	Stime	numeric
Flgs	nominal	flgs_number	numeric
Proto	numeric	proto_number	nominal
Saddr	nominal	Sport	nominal
daddr	nominal	Dport	nominal
Pkts	numeric	Bytes	numeric
State	nominal	state_number	numeric
Ltime	numeric	Seq	numeric
Dur	numeric	Mean	numeric
Stddev	numeric	Sum	numeric
Min	numeric	Max	numeric
Spkts	numeric	Dpkts	numeric
Sbytes	numeric	Dbytes	numeric
Rate	numeric	Srate	numeric
Drate	numeric	TnBPSrcIP	numeric
TnBPDstIP	numeric	TnP_PSrcIP	numeric
TnP_PDstIP	numeric	TnP_PerProto	numeric
TnP_Per_Dport	numeric	AR_P_Proto_P_SrcIP	numeric
AR_P_Proto_P_DstIP	numeric	N_IN_Conn_P_DstIP	numeric
N_IN_Conn_P_SrcIP	numeric	AR_P_Proto_P_Sport	numeric
AR_P_Proto_P_Dport	numeric	Pkts_P_State_P_Protocol_ P_DestIP	numeric
Pkts_P_State_P_Protocol_ _P_SrcIP	numeric	Category	Class Label

Features of BoT-IoT dataset

IoTID-2020

The following table shows the 80 numerical features of the IoTID2020 dataset

Features	Features	Features
Src_Port	Dst_Port	Protocol
Flow_Duration	Tot_Fwd_Pkts	Tot_Bwd_Pkts
TotLen_Fwd_Pkts	TotLen_Bwd_Pkts	Fwd_Pkt_Len_Max
Fwd_Pkt_Len_Min	Fwd_Pkt_Len_Mean	Fwd_Pkt_Len_Std
Bwd_Pkt_Len_Max	Bwd_Pkt_Len_Min	Bwd_Pkt_Len_Mean
Bwd_Pkt_Len_St	Flow_Byts/s	Flow_Pkts/s
Flow_IAT_Mean	Flow_IAT_Std	Flow_IAT_Max
Flow_IAT_Min	Fwd_IAT_Tot	Fwd_IAT_Mean
Bwd_IAT_Mean	Fwd_IAT_Max	Fwd_IAT_Min
Bwd_IAT_Tot	Bwd_IAT_Mean	Bwd_IAT_Std
Bwd_IAT_Max	Bwd_IAT_Min	Fwd_PSH_Flag
Bwd_PSH_Flags	Fwd_URG_Flags	Bwd_URG_Flags
Fwd_Header_Len	Bwd_Header_Len	Fwd_Pkts/s
Bwd_Pkts/s	Pkt_Len_Min	Pkt_Len_Max
Pkt_Len_Mean	Pkt_Len_Std	Pkt_Len_Var
FIN_Flag_Cnt	SYN_Flag_Cnt	RST_Flag_Cnt
PSH_Flag_Cnt	ACK_Flag_Cnt	URG_Flag_Cnt
CWE_Flag_Count	ECE_Flag_Cnt	Down/Up_Ratio
Pkt_Size_Avg	Fwd_Seg_Size_Avg	Bwd_Seg_Size_Avg
Fwd_Byts/b_Avg	Fwd_Pkts/b_Avg	Fwd_Blz_Rate_Avg
Bwd_Byts/b_Avg	Bwd_Pkts/b_Avg	Bwd_Blz_Rate_Avg
Subflow_Fwd_Pkts	Subflow_Fwd_Byts	Subflow_Bwd_Pkts
Subflow_Bwd_Byts	Init_Fwd_Win_Byts	Init_Bwd_Win_Byts
Fwd_Act_Data_Pkts	Fwd_Seg_Size_Min	Active_Mean
Active_Std	Active_Max	Active_Min
Idle_Mean	Idle_Std	Idle_Max
Idle_Min	Cat -Label	

Features of IoTID dataset