Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Software Department

Developed Intrusion Detection System based on Hybrid Feature Reduction and Machine Learning Techniques for Software Defined Networking

A Dissertation

Submitted to the Council of the College of Information Technology, University of Babylon in Partial Fulfillment of the Requirements for the Doctor of Philosophy Degree in Information Technology/ Software

By

Hasanain Ali Obyes Abboud

Supervised by

Prof. Dr. Wesam S. Bhaya

2023 A.C. **1445 A.H.**

بِسْمِ اللَّـهِ الرَّحْمَـٰنِ الرَّحِيمِ

اللَّهُ لَا إِلَٰهَ إِلَّا هُوَ الْحَيُّ الْقَيُّومُ ۚ لَا تَأْخُذُهُ سِنَةٌ وَلَا نَوْمٌ ۚ لَّهُ مَا فِي السَّمَاوَاتِ وَمَا فِي الْأَرْضِ ۗ مَن ذَا الَّذِي يَشْفَعُ عِندَهُ إِلَّا بِإِذْنِهِ ۚ يَعْلَمُ مَا بَيْنَ أَيْدِيهِمْ وَمَا خَلْفَهُمْ ۖ وَلَا يُحِيطُونَ بِشَيْءٍ مِّنْ عِلْمِهِ إِلَّا بِمَا شَاءَ ۚ وَسِعَ كُرْسِيُّهُ السَّمَاوَاتِ وَالْأَرْضَ ۖ وَلَا يَؤُودُهُ حِفْظُهُمَا ۚ وَهُوَ الْعَلِيُّ الْعَظِيمُ ۞ لَا إِكْرَاهَ فِي الدِّينِ ۖ قَد تَّبَيَّنَ الرُّشْدُ مِنَ الْغَيِّ ۚ فَمَن يَكْفُرْ بِالطَّاغُوتِ وَيُؤْمِن بِاللَّهِ فَقَدِ اسْتَمْسَكَ بِالْعُرْوَةِ الْوُثْقَىٰ لَا انفِصَامَ لَهَا ۗ وَاللَّهُ سَمِيعٌ عَلِيمٌ ۞ اللَّهُ وَلِيُّ الَّذِينَ آمَنُوا يُخْرِجُهُم مِّنَ الظُّلُمَاتِ إِلَى النُّورِ ۖ وَالَّذِينَ كَفَرُوا أَوْلِيَاؤُهُمُ الطَّاغُوتُ يُخْرِجُونَهُم مِّنَ النُّورِ إِلَى الظُّلُمَاتِ ۗ أُولَٰئِكَ أَصْحَابُ النَّارِ ۖ هُمْ فِيهَا خَالِدُونَ ۞

صدق اللَّهُ العظيم

سورة البقرة الآيات 255-257

Supervisor Certification

I certify that this dissertation entitled "**Developed Intrusion Detection System based on Hybrid Feature Reduction and Machine Learning Techniques for Software Defined Networking**" was prepared under my supervision at the Department of Software / College of Information Technology / University of Babylon, by **Hasanain Ali Obyes Abboud** as a partial fulfillment of the requirements of the degree of **Ph.D. in Information Technology / Software.**

Signature:

Name: **Dr. Wesam S. Bhaya**

Title: **Professor.**

Date: / / 2023

The Head of the Department Certification

In the view of available recommendations, I forward the dissertation entitled **"Developed Intrusion Detection System based on Hybrid Feature Reduction and Machine Learning Techniques for Software Defined Networking"** for debate by the examination committee.

Signature:

Name: **Dr. Sura Zaki Alrashid**

Title: **Assist Professor.**

Date: / /2023

Declaration

I hereby declare that this Dissertation, submitted to University of Babylon in partial fulfillment of requirement for the degree of Ph.D. in Information Technology / Software, has not been submitted as an exercise for a similar degree at any other University. I also certify that this work described here is entirely my own except for experts and summaries whose source are appropriately cited in the references.

Signature:

Name: Hasanain Ali Obyes Abboud

Date: / /2023

Certification of the Examination Committee

We chairman and members of the examination committee, certify that we have studied the dissertation entitled (**Developed Intrusion Detection System based on Hybrid Feature Reduction and Machine Learning Techniques for Software Defined Networking**) presented by the student **Hasanain Ali Obyes Abboud** and examined him in its content and what's related to it, and that in our opinion is adequate with standing as a Dissertation for the **degree of doctor of philosophy in Information Technology/Software**.

Signature:

Name: **Dr. Rafah M. Almuttairi**
Title: Prof.
Date**:** **/ /2023**
(Chairman)

Signature:

Name: **Dr. Belal I. Al-Khateeb**
Title: Prof.
Date**:** **/ /2023**
(Member)

Signature:

Name: **Dr. Ahmed Saleem Abbas**
Title: Prof.
Date**:** **/ /2023**
(Member)

Signature:

Name: **Dr. Firas Sabah Salih**
Title: Assist Prof.
Date**:** **/ /2023**
(Member)

Signature:

Name: **Dr. Rula A. Hamid**
Title: Assist Prof.
Date**:** **/ /2023**
(Member)

Signature:

Name: **Dr. Wesam S. Bhaya**
Title: Prof.
Date**:** **/ /2023**
(Supervisor)

Approved by the Dean of the College information technology, University of Babylon.

Signature:

Name: **Dr. Wesam S. Bhaya**
Title: **Professor.**
Date**:** **/ /2023**

Dedication

With love to my

Father and Mother,

Wife,

Brothers,

Sisters,

to my sons Mustafa, Ruqaya, and Ameer Al-Haq.

Hasanain

Acknowledgements

First of all, I have to express all thanks and gratitude to my God who gives me the ability to achieve such a perfect work and all that has been done without His blessing and support nothing can be done. I would like to thank my advisor Prof. Dr. Wesam S. Bhaya for his continuous support during my Ph.D. He was always there to listen and to give advice.

Special thanks go to all the Staff Members of Information Technology College for their faithful efforts to give me the utmost scientific topics and endless support in all directions aiming to bring into perfection their scientific followers.

Finally, I apologize to those whose names might be missed. But, I am grateful to all of them for the help that I have received in various research and day-to-day life matters.

Abstract

Software-defined networking (SDN) addresses the constraints of conventional networks through the separation of the control plane from data plane. This separation allows to transfer the network logic to a specific component known as the controller, which effectively manages the devices within the data plane. However, the SDN platform has become an attractive target for various attacks that can be exploited by intruders. For example, Denial of service (DoS) or distributed DoS (DDoS) attacks are one of the most destructive attacks that can interrupt SDN network functionality and make most network services unavailable to authorized clients.

Hence, the aim of this dissertation is to construct intrusion detection models based on optimized machine learning approaches to secure an SDN environment against DoS/DDoS as well as other types of attacks. The efficiency and effectiveness of detection solutions are enhanced through proposed four optimized models: Hybrid Feature Reduction (HFR) approach, Optimized CatBoost model (OCBM), Fine-Tuned Deep Neural Network (FTDNN) model, and Developed Soft-Based Heterogeneous Ensemble Detection (DSBHED) model.

HFR employs filter-based, wrapper-based, and embedded-based feature reduction approaches to select the best subset of features, improving the speed and performance of the models. Initially, features are evaluated using Mutual Information, Whale Optimization Algorithm, and Extra Tree Classifier. Subsequently, a majority-voting technique (MVT) filters out insignificant and inconsistent features, creating a reduced dataset. The mean technique is then applied to unify feature weights obtained from MVT, resulting in a subset of significant features.

The ML-based model, OCBM, and the DNN-based model, FTDNN, have been proposed to improve the binary prediction of DoS/DDoS attacks while maintaining a low false positive rate (FPR) and reducing processing time for training and testing. Optimizing the hyperparameters of the OCBM classifier including the number of boosted trees, maximum depth of each tree and learning rate, is crucial for accurately distinguishing normal traffic from DoS/DDoS attacks. This optimization process is carried out using the GridSearchCV (GSCV) technique to identify the best values for these hyperparameters.

On the other hand, in the case of the DNN-based model, RandomSearchCV (RSCV) is a valuable tool for determining the best structure and parameters of DNN by efficiently exploring the hyperparameter space. Moreover, unlike many other studies, several empirical cases were conducted to evaluate the performance of both the OCBM and FTDNN models against zero-day (unknown) attacks.

In addition, to accurately detect multi-class attacks such as port scan, brute force attack, SYN, UDP, and others, the DSBHED model is proposed. This model relies on a combination of multiple ML-based classifiers for prediction of final results. The components of the DSBHED model include the Optimized Random Forest (ORF), K-Nearest Neighbors (KNN), and Optimized LightGBM (OLightGBM) classifiers. The hyperparameters of ORF and OLightGBM are optimized using the GSCV technique to achieve the best performance.

The efficiency and effectiveness of the proposed models were assessed in terms of detection rate, accuracy, precision, F1-score, and FPR, in addition to the training and testing times, which were also computed for all proposed models. Using the InSDN, CICDDoS-2019, SNMP, and

CICIDS-2017 datasets, the empirical outcomes indicate that the proposed models attain better performance in attack detection due to all evaluation metrics, as compared to previous studies and existing ML classifiers carried out on the same datasets. Hence, our proposed models provide great confidence in securing SDN.

Declaration Associated with this Dissertation

Some of the works presented in this dissertation that has been published or accepted are listed below:

1- **Network Attacks Detection Depend on Majority Voting Weighted-Average for Feature Selection and Various Machine Learning Approaches.**
Webology journal, 2022

2- **Ensemble learning classifiers hybrid feature selection for enhancing performance of intrusion detection system.**
Bulletin of Electrical Engineering and Informatics, 2023

3- **Detection of DDoS attacks in software-defined networks based on majority voting-average for feature selection and machine learning approaches.**
The Second International Conference on Advanced Computer Applications, (ACA2023)

Table of Contents

CHAPTER THREE THE PROPOSED INTRUSION DETECTION MODELS

CHAPTER FOUR RESULTS AND DISCUSSION

List of Tables

List of Figures

List of Abbreviation

Abbreviation	Meaning
Adam	Adaptive Moment Estimation
CICDDoS-2019	Canadian Institute for Cybersecurity Distributed Denial of Service Attack 2019
CICIDS-2017	Canadian Institute for Cybersecurity Intrusion Detection System 2017
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DNN	Deep Neural Network
DoS	Denial of Service
DSBHED	Developed Soft Based Heterogeneous Ensemble Detection
FPR	False Positive Rate
FTDNN	Fine-Tuned Deep Neural Network
GBM	Gradient Boosting Machine
GSCV	Grid Search Cross Validation
HFR	Hybrid Feature Reduction
LR	Logistic Regression
ML	Machine Learning
MVT	Majority Voting Technique
NB	Naïve Base
OCBM	Optimized CatBoost Model
OLightGBM	Optimized Light Gradient Boosting Machine
ORF	Optimized Random Forest
RF	Random Forest
ReLU	Rectified Linear Unit

RSCV	Random Search Cross Validation
SNMP	Simple Network Management Protocol
SVM	Support Vector Machine
WOA	Whale Optimization Algorithm
XGBoost	Extreme Gradient Boosting

Chapter One

General Introduction

General Introduction

1.1 Introduction

Computer networks have become an essential part of modern life, being a core element, for instance, in security, health systems, industry, business, entertainment and education. Advents like Internet of Things (IoT) are connecting every type of electronic devices and control systems, increasing network requirements even further. This increasing number of devices that are communicating via networks, like the Internet, require new network paradigms that are more flexible in management, implementations, configuration and more hardware independent. Over the last decades, the network protocols and infrastructure evolution was restricted by some factors, such as vendor-specific technologies, compatibility with legacy technologies, and so on. That network ossification is a major barrier to innovation, and disruptive approaches are demanded to offer alternatives to this stagnation [1]. Therefore, conventional networks technologies are not adequate to supply current communication needs.

Therefore, Software Defined Networking (SDN) has emerged to tackle the problems of traditional networks. SDN decouples the control layer (which makes decisions) from the data layer (which forwards data packets). This separation allows for centralized control which enables more efficient traffic management and programmability. It is highly programmable where the network administrators can define network policies and behaviors using software, making it easier to adapt to changing requirements [2]. Due to its centralization, programmability, and dynamic adaptability, offers a more versatile approach to network management and is well-suited for the complex and ever-changing requirements of today's

networks. Thus, these features encourage many industrial and commercial companies to employ SDNs solutions in their network environments [1].

Although the SDN model offers many advantages, it is not immune to emerging threats that can be exploited by intruders to carry out various malicious activities. If an intruder succeeds in gaining access to the SDN controller, the whole network may be susceptible to serious threats. For instance, a Denial of Service (DoS) or Distributed DoS (DDoS) attacks, one of the reasons that make the SDN controller be unreachable [3].

DoS/DDoS attacks are designed to exhaust a target's resources, obstructing authorized clients from accessing them. These attacks can result in reputational and financial losses. They are distributed in nature and can be executed on a global scale using distributed botnets. Due to their widespread nature, the volatile temporal patterns, and the sheer scale of these attacks, detecting DoS/DDoS attacks is challenging. This challenge is due to the use of spoofed IP addresses and the complexity of identifying traffic patterns [2].

Unfortunately, all elements of SDN network (i.e., application, data, and control planes) are exposed to DoS/DDoS assaults, for instance, the communication line between the data and control planes. The intruder can send a large number of data traffic to an OpenFlow switch through data plane. If the source IP addresses of the these packets are spoofed (that is, their flow rules are missing in the flow table), therefore, the switch will send these packets to the controller via Packet-in-message in order to get a new rule. Due to this case, when the controller deals with large numbers of fake IPs, wide processing can be degraded its sources like CPU, bandwidth, and memory and that may causes also lost the whole network [3]. Thus, building efficient solutions (e.g., intrusion detection systems) to

secure the SDN platform from these attacks becomes an important aspect for the researchers in the last years.

An Intrusion Detection System (IDS) is a standard security measure designed to monitor and identify malicious or harmful activities within an SDN network. When incoming or outgoing network traffic displays patterns indicative of suspicious behavior, it triggers an alert for identifying potential attacks [4]. Hence, numerous approaches have been researched and developed with the aim of enhancing the detection rate and performance of IDSs. Some of these approaches include machine learning (ML) and deep neural network (DNN) techniques, which offer intelligent, efficient, and dynamic solutions to enhance SDN security [5].

On the other side, feature reduction (FR) approaches are one of the crucial stage to accurately building intrusion detection models. These approaches can eliminate redundant and irrelevant features while retaining only the most representative attributes or features from the original space (datasets). Utilizing the best subset of features (important features) not only enhances the detection rate and accuracy of the model but also diminishes the execution time (testing time) [5]. Therefore, utilizing fewer and more important features can aid in avoiding dimensionality issues and accurately detecting various network attacks with less processing time.

Encouraged by these facts, this dissertation proposes an efficient HFR approach to attain the best subset of features which can obtain from SDN network. In addition, it optimizes the ML-based and DNN-based models to build accurate and reliable models to detect DoS/DDoS attacks as well as different types of attacks in SDN platform.

1.2 Related Work

In recent years, various defense methods have been proposed for detecting DoS/DDoS attacks in SDN architectures. According to the nature of DoS/DDoS attacks, where the attacker can rapidly exhaust the resources of SDN networks or degrade their performance by utilizing large amounts of data traffic toward the SDN controller with the aid of only a few clients under his control [5]. Thus, securing the SDN platform from such attacks is crucial for maintaining the operation of the SDN and avoiding potential outages of equipment or network services.

This dissertation explores the latest solutions for detecting attacks in the SDN architecture. These solutions are primarily categorized as (I) statistical-based approaches and (II) ML and DNN based techniques.

I- Statistical-based Detections

The authors in [6], proposed a scheme to detect TCP-SYN flooding assaults in SDN architecture by utilizing the entropy computation method. They used a few features that based on destination IP and TCP-flags to compute the entropy value. Conversely, the proposed solution was used to detect only the TCP-SYN assault, with no consideration of other types of DDoS assaults.

Shanshan Yu et al. [7] provided a combined approach that based on entropy and an RF classifier for detecting DDoS in the SDN. A fast model relying on the entropy method was used at the data plane (switches) to compute the entropy value of the receiver's IP. If abnormalities are suspected, another RF-based detection technology will be applied on the SDN controller for additional detection. However, the researches utilized the OpenFlow switch to detect process, which is contrary to the aspect of

SDN functionality, to avoid any decision-making in the forwarding devices.

In [8], an entropy-based method was proposed to identify and prevent DDoS assaults in SDNs. Three threshold values were used to decrease the false positive rate (FPR). The proposed solution achieved a detection accuracy of 0.982 and an FPR of 0.004.

The authors in [9], utilized a generalized entropy measure to identify DDoS assault on an SDN controller. The results indicate that the proposed metric performed better than the information entropy measure and other distance measures.

Although all approaches that depend on entropy values are fast in their computation and consume a smaller amount of computational resources, but these methods are heavily based on the experience of security researchers to determine in advance the threshold worth. Moreover, the network data traffic is dynamic and unstable at all times. For instance, the traffic size is high throughout at the working time, whereas it is low after working time or on holidays. Therefore, choosing the ideal threshold requires lots of calculations and needs the historical observation of traffic behavior. In addition, the nature of data traffic is frequently changed over time; for example, new network protocols are often developed, while other network protocols are not utilized. Consequently, up-to-date network data traffic is complex because the attacker can simply construct novel attack with high degree of similarity to the normal packet, which is difficult to be recognized using a threshold based method. Thus, reliable and accurate detection solutions are remain required to effectively secure SDNs from assaults.

II. ML-based and DNN-based Detections

Recently, ML and DNN techniques based have been effectively utilized in order to detect DoS/DDoS and other types of attacks on SDN platform. The ML/DNN techniques are capable enough in discovering the differences between normal and attack traffic with a higher performance than other techniques, such as statistical-based methods.

The authors in [10] evaluated three ML algorithms, Naïve base, SVM, and DNN to detect Distributed DoS attack in SDN architecture. The SVM classifier attained a highest detection accuracy of (80%). However, additional improvements are required in terms of the accuracy metric.

A DNN technique [11] that depends on GRU was proposed to address the issue of DDoS assaults in the SDN network. The proposed method uses the NSL-KDD dataset for the training and testing processes without using any feature reduction techniques. The detection model achieved an accuracy of (0.89).

The authors in [12] introduced an ensemble learning-based approach for classifying DDoS assaults in SDNs. The proposed ensemble model employs the SVM, K-NN, and ELM classifiers. All of these classifiers make their decisions about the data, and then a max-voting method is applied to take the last decision. Wireshark network inspection tool utilized in this study to generate the network traffic. Based on the experimental results, this model achieved acceptable values in terms of sensitivity, specificity, and accuracy.

In [13], an IDS was proposed based on the XGBoost classifier for detecting different types of DDoS attacks, which was validated using the CIC-IDS2017 dataset. The proposed classifier utilized 80 features of the dataset. This model attained a detection accuracy of 0.9136, TPR of 0.974,

and FPR of 0.12. The proposed solution achieves a high accuracy; however, the FPR must be reduced.

As shown in [14], the authors suggested an ensemble-based approach that combined diverse ML classifiers to classify DDoS assaults in the SDN network. The proposed algorithms were SVM, NB, and KNN, which were combined separately with the SOM algorithm. The experimental outcomes indicated that the blend of SOM with SVM achieved a high detection rate and accuracy with a low FPR using the CAIDA-2016 dataset.

The authors of [15] introduced an intrusion detection model that relied on a deep auto encoder approach to detect DoS attacks. The proposed model attained an accuracy (95.73) using the CICIDS-2017 dataset, displaying its ability to identify DoS assaults. However, the researchers used a full feature dataset, and a manual-based approach was used to build the structure of deep network.

In [16], the authors design a new detection framework to identify DDoS assaults in SDN architecture. The proposed solution combined the KNN with K-Means algorithms for a detection approach. However, they was employed the Scapy method to mimic the normal and DDoS assaults for building the training data. The mimicked traffics lacked the attack diversity and were free from application-based DDoS assaults. Moreover, the detection model's performance was assessed using the outdated NSL-KDD dataset, which was created using traffic records from twenty years ago.

A study in [17] proposed an enhanced KNN classifier for detecting DDoS attacks on an SDN platform. An SDN-based topology was employed, consisting of ten virtual devices and one server, to generate a simulated dataset. However, the researchers used their dataset but have not made it public for validation. Furthermore, the simulated dataset lacked a

wide range of DDoS attack types that could target various layers of the OSI model, including those that specifically targeted the application layer.

The authors in [18] implemented various ML classifiers, namely, SVM, REP tree, MLP, J48, Random Tree (RT), and Random Forest (RF), to identify DDoS assaults in SDNs. They used the CICIDS-2017 dataset to verify the detection performance of these classifiers. When the assaults are identified by one of the utilized classifiers, a mitigation method starts to block the malignant traffic before the network is shut down. The experimental results revealed that the proposed approach achieved a detection ratio of 0.95.

In [19], the authors proposed a novel IDS framework to identify DDoS assaults in the context of SDN environments. The proposed framework consists of three different ensemble-based voting classifiers, namely, voting-RKM, voting-CKM, and voting-CMN, for attack detection. The proposed models were validated using CICIDS-2017, NSL-KDD, and UNSWNB15 datasets. Voting-RKM attained the best outcomes among all the prediction models utilized in this study. However, they did not adopt feature reduction techniques to their work.

In [20], the researchers proposed a solution based on LSTM and a single-class SVM to identify malicious traffic in an SDN network. The experimental results using the InSDN dataset show that the proposed model achieves a high detection rate with a low execution time. However, the authors were utilized manually-based approach to obtain the best values of hyperparameters for the LSM model.

In [21], the authors suggested DDoS-Net, an intrusion detection system that relies on a DNN-based technique that specifically combines the RNN with an autoencoder to detect DDoS assaults in an SDN architecture. The

proposed DDoS-Net model was verified using the CICDDoS-2019 dataset, which contains a comprehensive variety of DDoS attacks. The assessment of the DDoS-Net method demonstrates enhancements in the detection of attacks compared to other methods.

In [22], a security system was proposed to efficiently detect DDoS assaults on the SDN controller. The proposed system used a Generative Adversarial Network (GAN) technique to detect attacks. An experimental evaluation was performed using the CICDDoS-2019 dataset. The researchers compared the outcomes obtained from the GAN network with the LSTM, CNN, and MLP methods.

The authors in [23] presented a stack-ensemble learning model, which is a combination of neural networks, decision forest, logistic regression, and decision jungle algorithms, to classify different types of attacks. Moreover, multi and binary based classifications were performed. The experimental results conducted on the CICIDS 2017, UNSW NB-15, and CICDDOS-2019 datasets showed that the proposed model achieved good results.

The authors in [24] evaluated the effectiveness of three ensemble learning approaches (bagging, boosting, and stacking) and three ML algorithms (KNN, RF, and NB) for DDoS attack detection. The researchers used the CICDDoS-2019 benchmark to validate the performance of the proposed approaches. Two attribute selection methods were adopted to select significant attributes for intrusion detection. The experimental outcomes present that the stacking approach outperforms the other classifiers owing to four assessment measures: TPR, FPR, TNR, and accuracy.

The study in [25] proposed an intrusion detection model for a secure SDN environment. The proposed model contains a combination of a convolutional neural network (CNN), an attention mechanism, and a deep autoncoder (DAE), to detect and classify network traffic. The experiments in this study were performed using an InSDN dataset collected from an SDN environment. However, they conducted experiments using the full features of the InSDN dataset. The outcomes from the proposed solution illustrate high accuracy for both binary and multi classification tasks. Table (1.1) provides a summary of previous studies with their details.

Table (1.1) Summery of Related Work

References	Year	Dataset	Technique	Best Outcomes
Ref. [10]	2017	Simulated	ML	Acc.= 1 Pre.= 0.97 Recall= 0.96
Ref. [11]	2018	NSL-KDD	DNN	Acc.=0.89
Ref. [12]	2018	Simulated TCP traffic	ML	Acc=0.80 Pre.=0.80 Recall=0.80
Ref. [13]	2018	CICIDS-2017	ML	Acc.=0.9136 TPR = 0.974 FPR=0.12
Ref. [14]	2019	CAIDA 2016	ML	Acc.= 0.981243 Recall =0.971477 FPR=0.27140
Ref. [15]	2019	CICIDS-2017	DNN	Acc.= 0.9573

References	Year	Dataset	Technique	Best Outcomes	
Ref. [16]	2020	NSL-KDD	ML	Pre.= 0.994 Recall= 0.9835 FPR= 0.127	
Ref. [17]	2020	Simulated	ML	Acc.=0.994 Pre.=0.993 Recall=0.994 F1=0.9935 FPR=0.009	
Ref. [18]	2020	CICIDS-2017	DNN	Acc.=0.9501 Pre.=0.9546 Recall=0.9451 F1=0.9498 FPR=0.0052	
Ref. [19]	2020	UNSWNB15 CICIDS-2017 NSL-KDD	ML	CICIDS-2017 Acc.= 0.9777 Pre.= 0.9983 Recall=0.9313 F1=0.9636	
Ref. [20]	2020	InSDN	ML+DNN	DNN Acc.=0.905 Pre.=0.93 Recall=0.93 F1=0.93	ML Acc.=0.875 Pre.=0.89 Recall=0.93 F1=0.91
Ref. [21]	2020	CICDDoS-2019	DL	Acc. =0.990 Pre.=0.995 Recall=0.990 F1=0.990	
Ref. [22]	2020	CICDDoS-2019	DL	Acc.=0.9438 Pre.=0.9408 Recall=0.9789 F1=0.9594	

References	Year	Dataset	Technique	Best Outcomes	
Ref. [23]	2021	CICDDoS-2019	ML	Binary Acc.=0.97 Pre.=0.99 Recall=0.97 F1=0.978	Multi Acc.=0.8493 Pre.=0.8465 Recall=0.8409
Ref. [24]	2021	CICDDoS-2019	ML	Acc.= 0.973 TPR= 0.96 TNR=1 FPR= 0.7	
Ref. [25]	2022	InSDN	DNN	Binary-Acc.=0.977 Multi-Acc.=0.955	

Therefore, this dissertation focuses on addressing the critical limitations of solution detection methods.

- Numerous previous studies have been constructed either out-of-date datasets (e.g., NSL-KDD), non-SDN datasets, or research datasets that are not made public, which reduces their ability to adapt to new and unseen attacks, such as those described in [10-12] and [16-19].
- Lots of irrelevant and redundant data in high dimensional datasets interfere with the detection process, as demonstrated in studies such as [10-11], [13], [15], [19], and [25].
- Employing a single classifier may prove inadequate for accurately detecting the diverse types of network assaults, including port scans, DoS or DDoS attacks, and brute force attacks, as demonstrated in [23].

- Developing effective DNN-based models often requires a laborious and time-consuming trial-and-error approach due to the lack of a systematic approach for their construction, as showed in studies like [15] and [20]
- The prevalence of high false positive rates (FPRs) poses a significant challenge for most IDSs, as demonstrated in [11] and [13].

1.3 Problem Statement

SDN's programmable and dynamic nature introduces numerous security vulnerabilities, requiring the development of innovative security solutions. A critical concern in SDNs is the susceptibility of SDN controllers to DoS/DDoS attacks, which can render the entire SDN architecture unreachable [26][27]. In addition, according to Kaspersky lab report [28] "a doubling of DDoS attacks in the first quarter of 2020 compared with the fourth quarter of 2019, also an 80% jump compared with the same quarter last year". DDoS attacks exploited the COVID-19 virus by targeting the educational platforms, healthcare organizations, and government agencies. Moreover, the intelligent report for the same lab also indicates that the DDoS assaults are the most prevalent threats recorded in 79 countries in the majority of organizations. Furthermore, the complexity and number of these assaults are grow which required efficient detection methods for such assaults [29].

Thus, the main aim of this dissertation is to construct accurate and efficient detection models that are capable of recognizing DoS/DDoS from normal traffic, as well as identify multi-types of attacks. To achieve this, an efficient feature reduction approach is proposed, and various ML-based and DNN-based approaches are optimized, thereby increasing the security level of the SDN architecture.

1.3 Dissertation Objectives

The objectives of this dissertation can be summarized as follows:

1. To propose a new hybrid feature reduction model to identify significant features.
2. To construct optimized machine learning-based and deep neural network-based models to improve their performance for binary detection of DoS/DDoS attacks and outperform competitive approaches.
3. To evaluate the performance of ML-based and DNN-based models in detecting zero-day or unknown attacks (in our situation DoS/DDoS).
4. To propose a soft-based heterogeneous ensemble learning to improve the prediction accuracy and detection rate with low false alarm rate via their ability to detect multi-class attacks instantiated in the InSDN-2020, SNMP, and CICDDoS-2019 datasets.

1.4 The Dissertation Contributions

This dissertation has achieved the following contributions in comparison to previous work:

1. Develop a hybrid feature reduction (HFR) approach to obtain only significant features that result in a speedup detection stage, reduce computation time, and accurately detect DoS/DDoS attacks. Therefore, improve the overall performance of proposed models.
2. Propose an optimized CatBoost model based on grid search optimization technique by focusing on most effective hyperparameters that aids to construct a lightweight IDS model

and making it appropriate for online or real-time detection of assaults. To the best of our knowledge, this is the first attempt to utilize this model in construction the intrusion detection model for SDNs.

3. Propose a deep neural network model based on a random search optimization technique to minimize human intervention in the design network phase and select a suitable structure with hyperparameters for binary attack prediction.
4. Propose a new soft-based heterogeneous ensemble learning model of three ML classifiers from different classifier families. The selected classifiers are, LightGBM, Random forest (RF), and K-nearest neighbor (KNN). In our model, the crucial hyperparameters of LightGBM and RF classifiers are optimized based on grid search technique. The outputs of the three predictors are combined by soft voting method to obtain the final output of the ensemble model.

1.5 Dissertation Outline

This dissertation consists of various chapters. It is summarized as follows:

Chapter Two: presents the SDN environment and its features, the challenges faced by SDNs, and multiple types of attacks. In addition, the protocol (OpenFlow) and its parameters are explained. The ML and DNN based techniques have also been described. Finally, the evaluation of the ML and DNN based techniques was explained.

Chapter Three: depicts the proposed models in depth, which involve the proposed frameworks and algorithms to implement them.

Chapter Four: discuss the outcomes of proposed models. The validation results are viewed and compares with competing ML-based and DNN-based models as well as other existing works.

Chapter Five: the conclusions of the dissertation was explained, and some ideas were suggested for future works.

Chapter Two

Theoretical Background

2.1 Software Defined Networking

Software defined networking (SDN) can be defined as an emerging architecture that is manageable, dynamic, cost-effective, and adaptable, making it ideal for the high bandwidth and dynamic nature of today's applications. This paradigm separates the forwarding functions and network control enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services [29]. Figure (2.1) illustrates the framework of SDN network.

The main goal of SDN platform is to split the control plane from the data plane. One consequence of this methodology is that the control logic is implemented in a logically centralized controller and that network switches simply forward packets. The SDN controller is now programmable, making it possible to establish dynamic routes for network packets. In conventional networks, control plane and data plane devices are coupled with each other. In this case, if any new rule is needed to be updated in the existing mechanism, it has to be applied in all the devices. Even a simple modify can take days or weeks to be accomplished [2]. This limitation of conventional networks makes this process costly and time consuming. However, in the time of digital world, companies and their networking facilities require to be fast and flexible. SDN addresses this requirement by reducing the limitations of conventional networks. In SDN, if any update has to be implemented as per the user requirement, then these changes will be made only in the control plane. This process is accomplished in less time in SDN compared to the conventional networks [3].

Consequently, SDN enhances the performance of the network and obtains cost effective network services. In SDN, the controller represents

the brain of the whole network and offers global visibility to all the network functionalities. This unique feature of the SDN controller provides many applications and mechanisms for security purposes [8]. At the same time, architecture design problems such as separation of data and control planes, and the place of controller lead to several security challenges [9]. One such challenge is to secure SDN platform from different types of attack, i.e., DoS or DDoS attacks [30]. Nowadays, DoS/DDoS has become one of the most destructive attacks. These attacks try to consume the system resources by the flood of network traffic at victim device. This flooding of network traffic degrades the performance of system, rendering services unavailable to authorized clients. DoS/DDoS attacks target the communication channel between switches and controller and the controller's resources. Attack like DoS/DDoS on an SDN controller can paralyze the entire network [30]. Thus, it is essential to build an accurate and efficient solution to detect DoS/DDoS attacks in SDN. Numerous IDSs have been introduced for detecting such attacks by researchers [12]. An IDS is a security application or device that is applied to surveillance and inspect the network traffic for any malevolent activities and to alert the administrator's system when attacks are detected. In recent years, ML and DNN based techniques are being utilized extensively. There are several IDSs have been used ML and DNN techniques for numerous attack detection by enhancing the accuracy of the prediction system [31]. In this dissertation, various ML and DNN based approaches were proposed for DoS/DDoS detection as well as different types of attack such as port scan, brute force attack, etc. The proposed IDS models present promising approaches for achieving more accurate predictions.

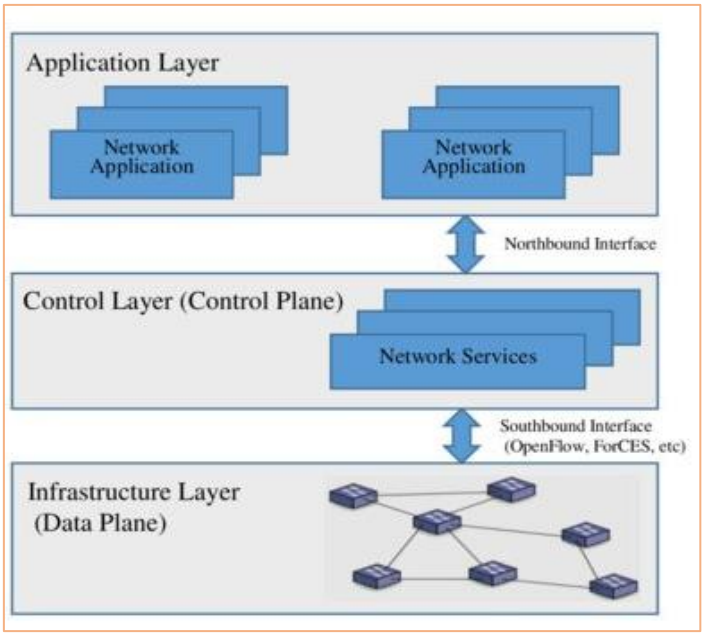


Figure (2.1): The SDN infrastructure [31].

As seen from Figure (2.1), the SDN infrastructure contains three layers: The application, control, and infrastructure layers.

2.1.1 Infrastructure layer

The infrastructure or data plane consists of devices that contain tables of flow entries. These devices use a secure transport layer protocol to securely communicate with a controller about new entries that are not currently in the flow table. Each flow entry includes matching rules and actions that guide the data plane on the action to take for the matched flow. When a packet arrives at a switch the header fields will be matched against the matching rules available in the flow table and if a match is found the action specified by the flow entry will be taken by the switch otherwise the packet header will be forwarded to the controller for further process. The controller then processes the header and constructs a flow rule to be installed in the flow tables of the switches along the chosen path [31].

2.1.2 Control layer

This layer has a controller, which is responsible for the proper functioning of the network. The controller keeps a global eye on all the devices and their working in the network. In the case of an unseen or new incoming packet, a message called packet-in is transferred to the SDN controller, which takes the proper decision for that packet. SDN controller and switches communicate through OpenFlow protocol [33].

2.1.3 Application layer

This layer maintains a pool of network based applications, for instance firewalls, routing, IDSs, and so on. These applications are implemented on the controller in order to define the control logic according to the requirement of the functionalities [32].

2.2 The Challenge Faced in SDN

The programmable and dynamic nature of SDN introduces a range of security challenges which require innovative security solutions. For instance, the centralized controller provided by SDN offers numerous advantages for network management and flexibility. However, it also may lead to various risks, particularly the risk of attacks aimed at disrupting or shut down a service, network, or system by overwhelming it with excessive traffic or requests. This can lead to service unavailability, rendering the target system inaccessible to legitimate users. This represents a significant security challenge for SDN environments [34].

2.3 Advantages of SDN

SDN platform provides the following benefits [35]:

- Global control: The controller has a global view of the network topology, network status، and application requirements.

- Flexibility and programmability: The data plane can be dynamically programmed to improve network resource allocation.
- Openness: Communications between the controller and forwarding devices do not depend on the device suppliers.

2.4 OpenFlow Protocol

The OpenFlow protocol [36] has represented the backbone of SDN. The SDN switches have managed by OpenFlow, the controller can process all flows of network packets by OpenFlow. The tables which exist in the switches explain the ingress and egress paths of incoming packets. OpenFlow has made these tables reachable by controller software. Then, the OpenFlow switch has received flow table entries or flow entries from the controller by the secure channel.

OpenFlow specification: The OpenFlow specification has encouraged vendors to execute OpenFlow on their devices. The organization which encourages acceptance of SDN and works with many vendors is called Open Networking Foundation, it has various groups to develop OpenFlow specifications. Figure (2.2) explains the architecture of an OpenFlow [37]. Searching, matching, and forwarding according to action determines by the controller software, all rely on the specification of OpenFlow. The first version of OpenFlow (version 1.1) was released in February-2011.

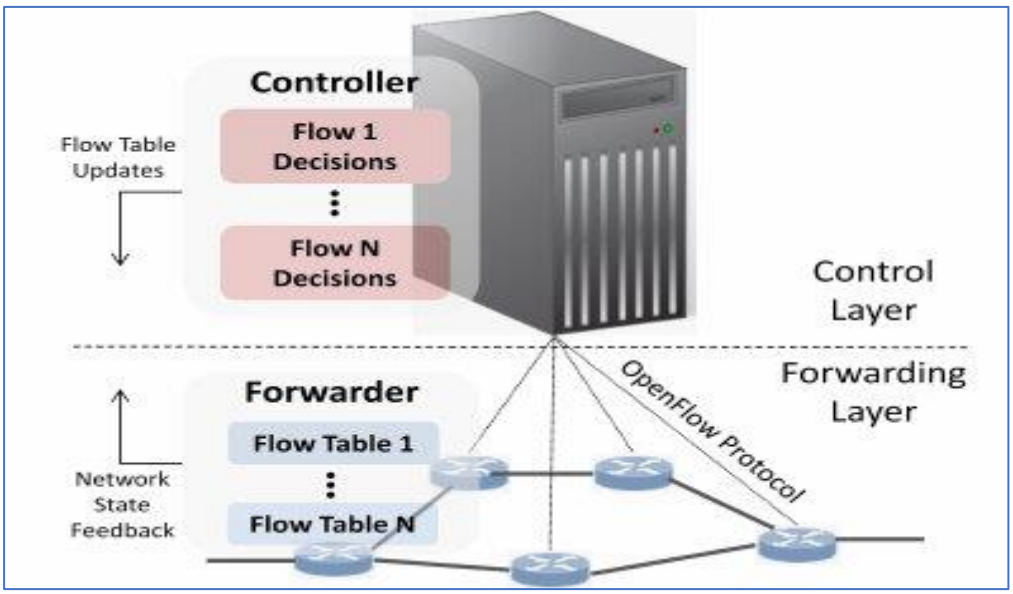


Figure (2.2): The architecture of OpenFlow [37].

The main process of an OpenFlow is as follows [37]:

- The controller software puts flow entries in the switch.
- Incoming packets have been checked by the switch and detected matching flow, then the SDN switch executes the related action.
- If matching rule is not found in the switch's flow table, the SDN switch passes the packet to the controller to learn the switch how to treat the packet.
- Afterward, the switch with new flow entries will be updated by the controller, then the switch can process the packet locally.

Secure channel: The data plane and control plane have been connected by a secure channel. When the switch has activated in the network, the switch looks for the controller to connect to it. Then, the switch builds a secure channel through which it exchanges information with the controller. The secure channel may be connected to the controller physically or virtually (by another switch remotely), this connection is a TCP connection [38]. To protect mutual data between the controller and switch from penetration, data must be encrypted. Generally, transport layer security (TLS) has been used in encryption. If the connection has failed, the SDN

switch will search for another controller to connect with it. If there only one controller has been found, it enters in "fail secure mode" then, every unknown packet has been dropped. When the OpenFlow switch processed the unknown incoming packet without sending it to the controller, this type is named Hybrid Switch. In case of a controller failure and the switch depends on its intelligence, it means that the switch has not followed an OpenFlow protocol, which means the network has lost SDN architecture. Figure (2.3) shows the secure channel in OpenFlow [39].

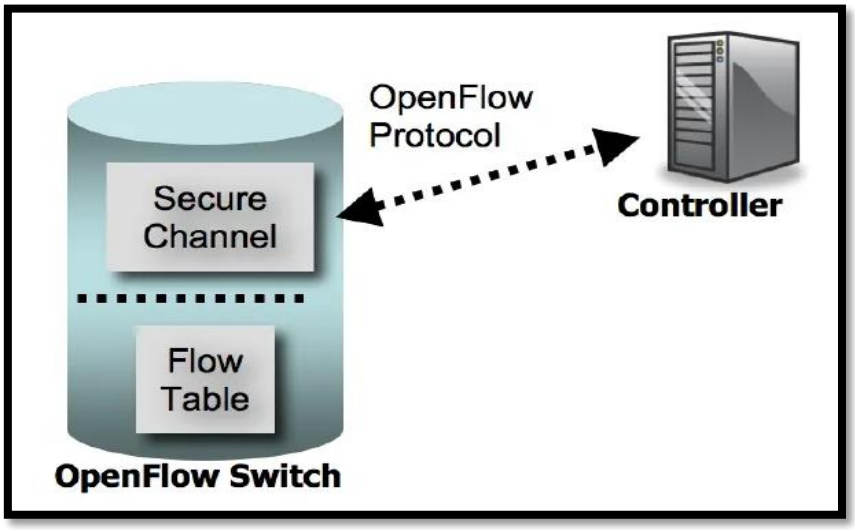


Figure (2.3) The secure channel in OpenFlow [40] .

Flow table: The flow table an existing in the switch, it includes flow entries. The fields in OpenFlow version 1.0 as shown in figure (2.4).

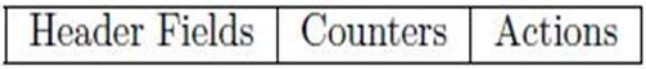


Figure (2.4): The fields of flow table in OpenFlow version 1.0 [40].

The number of fields in a flow table of OpenFlow version 1.3 will increase as presented in figure (2.5). These fields will be described.

Figure (2.5) The fields of flow table in OpenFlow version 1.3 [40].

- Match Fields: Match fields of the flow table in OpenFlow version 1.3 are the same header fields of the flow table in OpenFlow 1.0 version, these fields were obtained from packet. These fields are necessary for the lookup process inside the flow table, like Src-mac address, Des-IP address, and etc. Figure (2.6) illustrates match fields of the flow table.

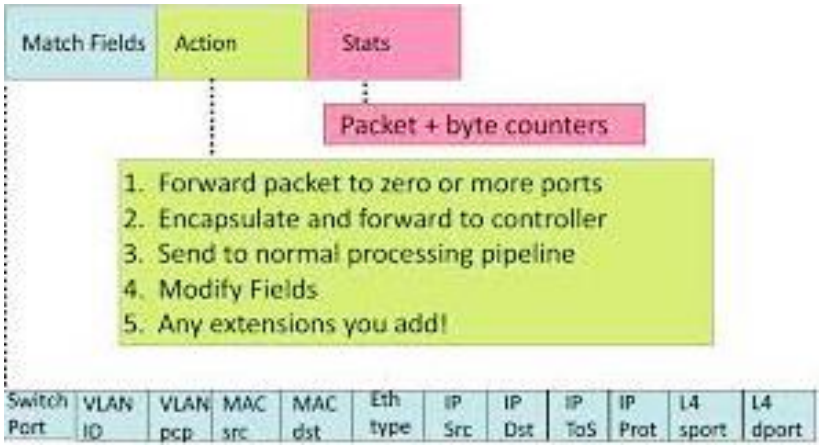


Figure (2.6): Match Fields of the Flow Table [41].

- Instruction Sets: These should be executed when a packet is matched. They determine the way in which the packet is handled (dropped or forwarded).
- Counters: It collects statistic information about a specified flow, including, the amount of bytes, how many packets come in, and flow duration.
- Priority: The flow rule in SDN means flow entry in the switch's flow table. When the SDN switch searches in all flow rules to match an incoming packet of traffic flow, if there are relevant flow rules, the rule with the highest priority has been selected.
- Timeout: Flow entries have an idle and hard timeout. Hard timeout means timeout after which the entry is removed from the device. Idle timeout means timeout in which if no packets are hitting to specific entry for the duration, the entry will be removed from the device.

- Cookie: Data value has placed by the controller, it has used to filter flow statistics, and flow modification [41].

2.5 Attacks on SDN

The centralized design of SDN platform introduces new vulnerabilities that can make the SDN architecture vulnerable to different types of security threats [42]. In fact, all SDN planes are unavoidably subject to diverse types of attacks. Some of these attacks are specific for the SDN i.e. as a result of splitting the control and data plane functionality. These attacks can occur in the SDN controller or on the communication channels between the data plane devices and control plane. Moreover, there are numerous attacks that are popular between SDN standard and the traditional networks for example, the attacks on the application plane or data layer elements. For instance, in the case that the intruder successfully gets illegal access to a vulnerable application or machine in a traditional network, a small portion of this network or a single machine is affected by this assault. The intruder needs to increase his privilege or utilizes the target machine to start new assaults against different machines [43]. Attack classes against SDNs are illustrated as follows [32]:

2.5.1 Denial of Service Attack

Denial of Service (DoS) attacks are considered the most attractive types of attack to the intruders. It is also considered one of the most dangerous threats to the security of SDN networks [12]. Another version of the DoS attack is DDoS attack. The DDoS attack is an explicit attempt to prevent authorize users from accessing the network services. The emergence of the DDoS attacks typically does not occur suddenly, but the onset of an attack on a target system produces from a series of preparatory steps by the adversary that we can identify and measure. The operation of

DDoS attacks follows many consecutive phases. The attacker initially starts to compromise multiple agent machines that are widely distributed geographically by scanning the vulnerabilities in these devices. Once an intruder successfully identifies some system vulnerabilities, he can compromise these machines using a malicious program such as Trojan Horse. By replicating the malicious file in multiple agents، the intruder has the capability to control many devices that can reach several tens or thousand (commonly referred to as bots) to initiate DDoS attacks without the awareness of the device's owner [44]. The discovery of vulnerabilities and exploitation process of the agents are usually performed automatically, for example, by sending e-mail messages contain the attack code attachment. The groups of bots, known as a botnet can get orders remotely from an attacker, i.e., bot-master. The bot-master can perform large-scale DDoS attacks to flood a legitimate service or network by sending a control command to the botnet agents to generate useless traffic without getting noticed. Consequently, the victim resources become overwhelmed with a crushing volume of traffic in a short duration, which significantly slows down the system service or the network ability to respond to the legitimate users. Denial of service attacks can be divided as follows [45]:

- **Volumetric attacks**: these are the most used attacks and are based on flooding a target with heavy traffic to exhaust its network bandwidth. The magnitude of this kind of attacks is measured in bytes per second. Common volumetric attacks include UDP flooding, spoofed packet flooding, and ICMP flooding.
- **State exhaustion/protocol exploitation**: these methods are used to exhaust the resources of devices by exploiting network protocols, in order to turn the network or operating system unavailable. Its magnitude is measured in packets per second. Fragmented packets,

Ping of Death, and SYN Flooding are examples of the protocol exploitation attacks.

- **Application layer attack**: this kind of method exploits the application layer protocols in order to crash the application or underlying a server. The magnitude of such attacks is measured in requests per second. Slowloris, slow-rate HTTP POST, slowhttptest are example of slow rate DoS application attacks.

2.5.2 Password Guessing Attack

The PGA attains access to the target machine by breaking the password and username credentials. There are two different scenarios of BGAs are followed to find the valid credentials. First, dictionary assault is involved by generating a dictionary for all possible passwords and users afterward try each of them. Second, an auxiliary scanner tool was used to identify the right credentials [7].

2.5.3 Web Application Attack

In this attack, the SQL injection and Cross-site scripting (XSS) are example of this attack. In an XSS attack, the intruder can bypass the access controls of the client machine by injecting malicious code into a trusted website. Once the client access the web application site, the malicious script will be executed. Consequently, the attacker can obtain sensitive information from the client machine, such as session tokens, cookies, and so on. In SQL injection, the intruder can utilize malevolent quarries to manipulate the database behind the web application, thereby allowing the intruder to get the content of the entire SQL database. The intruder can attain unauthorized access to any sensitive data or web application on the website [15].

2.5.4 Port Scanning Attack

Port scanning or probe is the most vital phase for an intruder before starting his attack. Typically, it is used to find some information about a target machine or system to perform another attack, that is, a port scan usually precedes another attack, which may be used to achieve a malicious goal [46]. Port scan consists of sending several types of packets in order to know more about a target host or network. Through the answers obtained from analyzing these packet, the attacker is able to gather information that may help in future attacks. Some types of information that can be discovered include the activity of the servers, information regarding software used in the system, information about the firewall or network topology [47]. Port scan attacks are typically performed by using specific tools, such as nmap [48].

2.5.5 Botnet Attack

Although several things or devices access the Internet, the provided security does not assurance the optimum functioning to prevent infiltration attacks. The attacker can control many infected devices, referred to botnet to run different malicious activities such as fraud attack, stealing information, web applications server, or launching DDoS against victim server [46].

2.5.6 Exploitation Attack

In this attack, the intruder is able to gain access the system with a normal user account and then exploits system vulnerabilities to gain root access to control the system [31].

2.6 Intrusion Detection System for SDN

Intrusion Detection System (IDS) represent a standard security measure designed to monitor and identify malicious or hateful actions in SDN environment. When outgoing or incoming network traffic exhibits patterns indicative of suspicious behavior, an alert is triggered, referring to an identify attack [49].

For many years, authors have adopted traditional ML-based approaches, such as Naïve Base (NB), Support Vector Machine (SVM), Deep Neural Network (DNN), and others for attack detection. These suggested solutions have attained different grades of success while also revealing some inherent restrictions. These solutions had a high false alarm rates (FARs) with high computational cost as mentioned in [50]. Thus, this dissertation proposes various optimized ML and DNN approaches in order to construct efficient IDS models in terms of all evaluation metrics that maximize the security level for SDNs.

2.7 Machine Learning Approach

ML is an important area of artificial intelligence concerned with constructing and understanding models that learn from data. The ML concept includes many techniques that construct the model on a sample of data named a training set. Then the quality of the model or algorithm is measured by predicting the second part of the data, named the testing set. From these datasets and by applying ML techniques, important information and hidden patterns can be extracted [51]. Therefore, ML can be used for intrusion detection system to automatically build a predictive model depends on the training dataset. A general workflow of the ML based detection mechanism is shown in Figure (2.7).

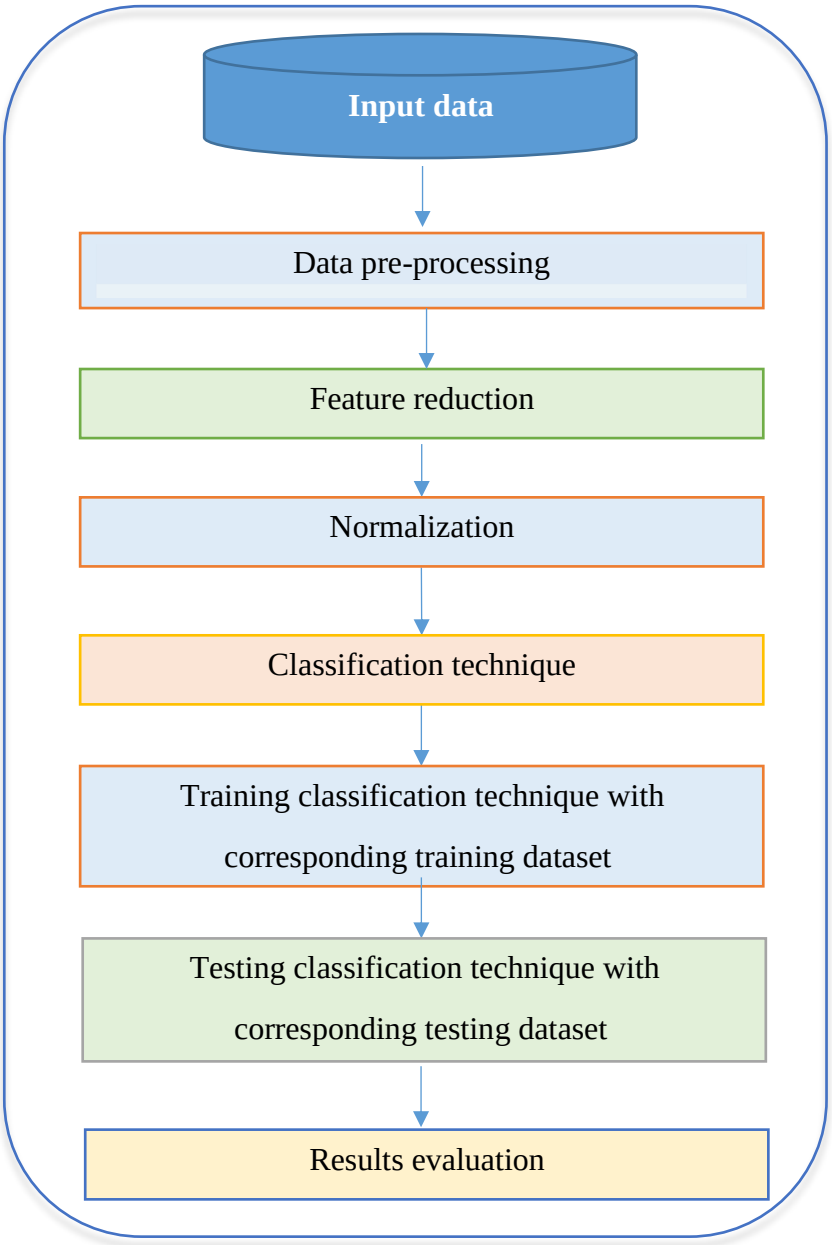


Figure (2.7): Workflow of intrusion detection utilizing ML approach [51].

2.7.1 Data Pre-processing

Data pre-processing stage is essential in ML to prepare the dataset that tends to have low-quality data [52]. This stage involves data cleaning operations such as handling missing values, duplicate data, inconsistent data, and label conversion. Efficiently preprocessing data can improve its accuracy, which can increase the quality of models and enhance its reliability [53].

2.7.2 Feature Reduction

Feature reduction (FR) or selection is a crucial stage in IDS ML-based techniques. Determining the appropriate FR approach resulting in an improve the performance of IDS. It also reduces the operational load by reducing the number of features in the dataset and creating a new one [54]. FR approaches can be classified as filter-based, wrapper-based, and embedded methods [55]. The key role of this strategy is to assess the importance of the elected subset of features, which is explored in the search space provided to get the optimum solution.

2.7.2.1 Filter Approach

In filter-based approach, evaluate the significance and choice of the features (or attributes) depend on the statistical metric such as mutual information method. That is, filter approaches are usually used before the prediction model [56]. Filter approach framework is illustrated in Figure (2.8):

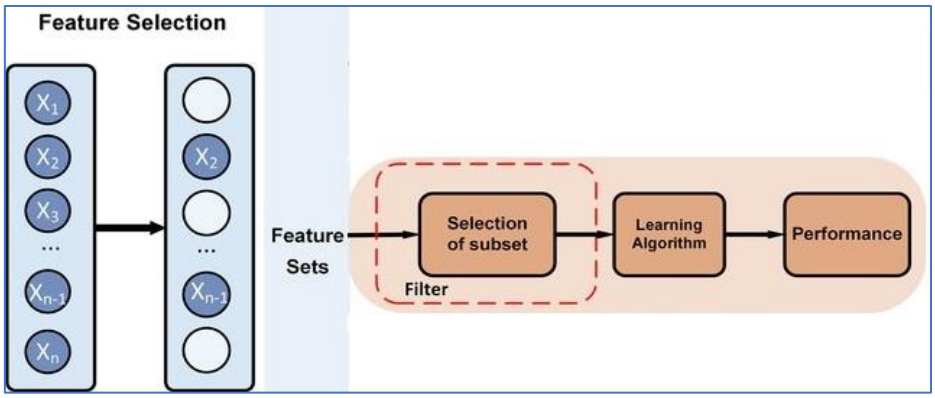


Figure (2.8) Filter-based feature reduction technique [56].

Mutual Information (MI)

MI is a metric of the amount of information that one random variable has about another variable. In the context of FR, this measure is valuable because it provides a way to determine the importance of a subset of

features with respect to the target vector. Formally, the MI is known as follows [57]:

$$MI(F,T) = H(F) - H(F|T) \qquad (2.1)$$

Wherein MI (F , T) is the MI for F and T. H(F) is the entropy for F and H(F | T) is the conditional-entropy for F given T.

2.7.2.2 Embedded Approach

Embedded approach performs feature selection during the implementation of the prediction algorithm. One of the key advantages of this approach is its ability to consider the interactions between features and the model's performance [58]. Many types of decision trees have performed embedded methods (e.g., extra tree classifier). The framework of embedded method is shown in Figure (2.9)

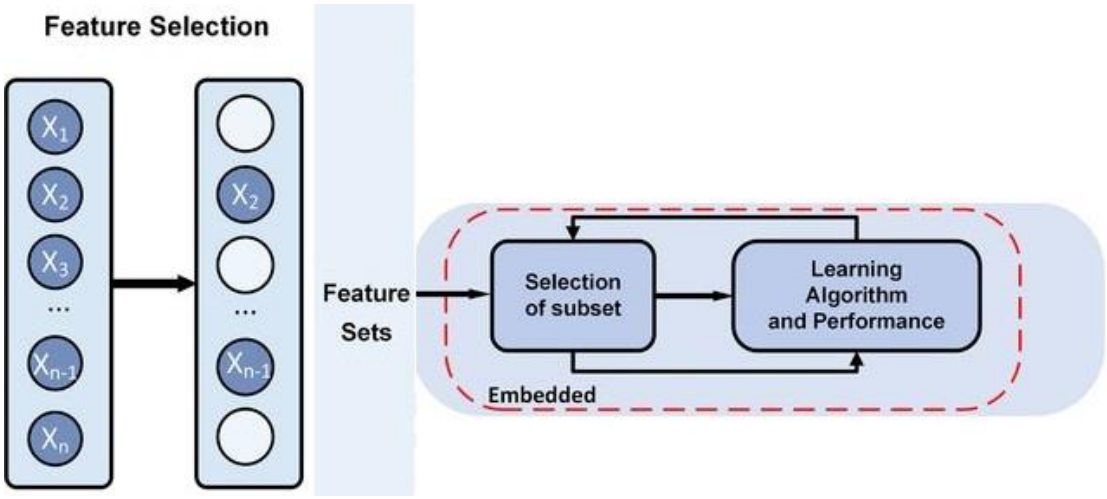


Figure (2.9) Embedded-based feature reduction technique [58].

Extra Tree Classifier (ETC)

ETC create several decision trees from the entire training dataset. For the root node, the algorithm selects a split rule depend on a random subset of features (M). It chooses a random split to divide the parent node

into two random child nodes. This process is repeated in each child node until reaching a leaf node. A leaf node is a node that does not have a child node. The predictions of all the trees are combined to form the final prediction using a majority vote strategy. To perform FR, during the construction of the forest, for each feature, the Gini importance is calculated. Each feature is ordered in descending rank according to its Gini importance of each feature [59].

2.7.2.3 Wrapper approach

Wrapper approach represents a sophisticated and powerful approach to identifying the most informative features for ML models using evolutionary algorithms such as whale optimization algorithm (WOA). Unlike filter methods, which rely on statistical measures, and embedded methods, which integrate feature selection into the model training process, wrapper methods handle feature reduction as a search problem. They evaluate subsets of features by training and testing the ML model iteratively, simulating the real predictive task [60]. Figure (2.10) shows the general framework of wrapper approach.

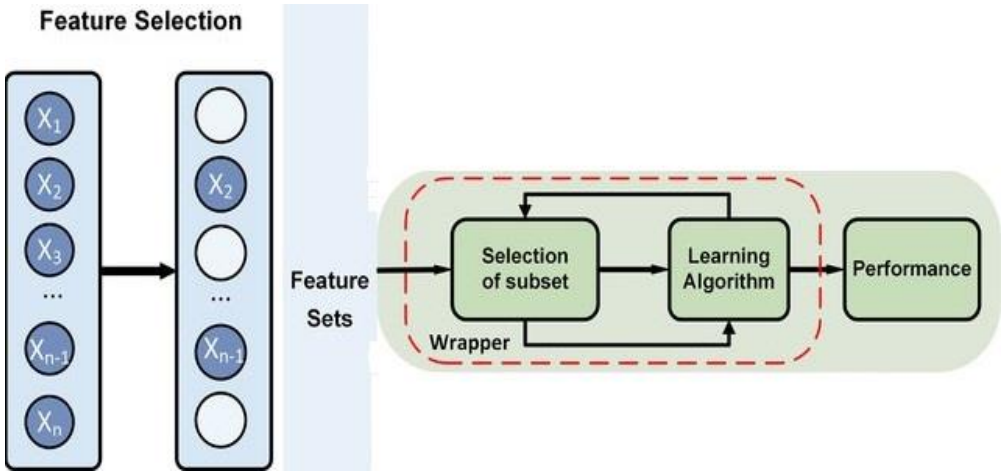


Figure (2.10) Wrapper-based feature reduction technique [60].

Whale Optimization Algorithm (WOA)

WOA is an optimization algorithm proposed by Lewis and Mirjalili in 2016. It is inspired to simulate the natural behavior of the humpback whales. These whales are typically depending on the hunting behavior as a way of survival. However hunting strategy have been previously introduced to address optimization problems, the uniqueness of the whale algorithm is the ability to employ a random or best agent in the search space to chase the prey. It also has the capability to mimic the bubble-net attaching mechanisms of the humpback whale by using spirals The modelling of this algorithm involves three operators simulate the search for prey (exploration phase), the encircling prey, and the bubble-net foraging (exploitation phase) behavior of humpback whales [61].

2.7.3 Data Normalization

Normalization is necessary to avoid a difference with large values that dominate the results, so the term normalization is used to make a normal variable (feature). The aim of normalization is to ensure that a whole set of values have a certain property. So that all the features should be expressed in the same unit of measurement [62].

There are many important types of data normalization such as min-max normalization and Z-score. In this dissertation, a Min-Max normalization technique is adopted to perform a transformation of the data to fall within a specified range [0,1]. The equation below generates a new variable $\bar{x}$ [62].

$$\bar{X} = \frac{x - x_{min}}{x_{max} - x_{min}} \qquad (2.1)$$

Wherein: Xmax is the maximum value of original value for any feature. Xmin is the minimum value of original value for any feature. X represents the feature value.

2.7.4 Classification Techniques

The classification technique is one of the techniques utilized in data mining, and it has been widely utilized in IDSs in SDN network security. The simplest definition of data mining is knowledge discovery. Another definition is that it denotes to the ability to take data as an input and extract patterns that cannot be observed by traditional methods [63] [64].

Classification can be defined as the key element of data mining, which is a technique of supervised learning. It is the process of identifying each instance of data into a kind of predefined classes.

Typically, classification algorithms involve two main stages, as outlined below [63] [65]:

A- Training stage: at this stage, training data are analyzed for the purpose of building a learning model through which the data instances are classified by class-based labels.

A- Testing stage: at this stage, the training model is used for the purpose of assigning a label classification to unlabeled instances or prediction classes for predefined classes.

The IDS can be considered as the classification issue. The system must gather enough audit data. A classification algorithm is then utilized this data to build a classifier. Finally, the classifier receives the new unlabeled incoming data then predicts each instance's class based on the training model that was previously built [65].

2.7.4.1 CatBoost Algorithm

CatBoost is a new variant of the gradient boosting decision tree approach developed by Prokhorenkova et al. in 2018 [66] . It uses a symmetric tree as the base learner, as showen in Figure (2.11). Unlike the standerd decision tree, a symmetric tree confirms that the nodes at similar depth depend on the same features for spiltting. The benefit of building a symmetric tree, improves the processing speed and feature handling when compared with a typical decision tree[67].Moreover, it can handle catogrical features without need to convert them to the numeircal during the preprocessing step.

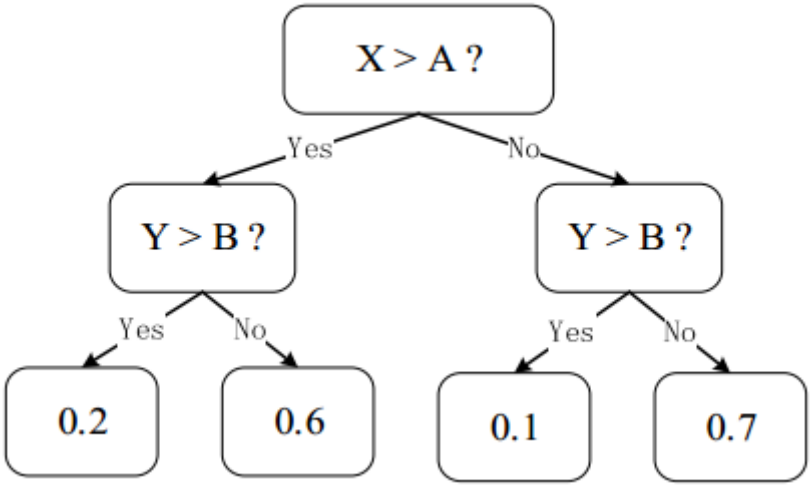


Figure (2.11) The structure of symmetric tree [67].

During the construction of DTs, the CatBoost algorithm utilizes a method named gradient optimization, that is, the algorithm usually looks at the direction that reduces the loss function. This process helps the algorithm prioritize the regions of data where the error is higher, thus leads to more accurate predictions. In the context of intrusion detection, adopted the CatBoost classifier provides many advantages, such as handling large datasets and mitigating sample data imbalance by using a weighted loss function, which improves the detection rate of minor classes[68].

2.7.4.2 Random Forest Algorithm

Random forest (RF) is a supervised ML technique depends on the decision tree model and generates k different training data subsets from an original dataset and then k decision trees are constructed by training these subsets. Finally, RF is constructed from these decision trees. Every new sample of the testing dataset is predicted by all decision trees. For classification, the predictions are combined using a max vote scheme [69]. The feature can be assessed in every tree using several methods such as entropy and information gain methods [70]. The core idea of the RF algorithm is illusterated in Figure (2.12).

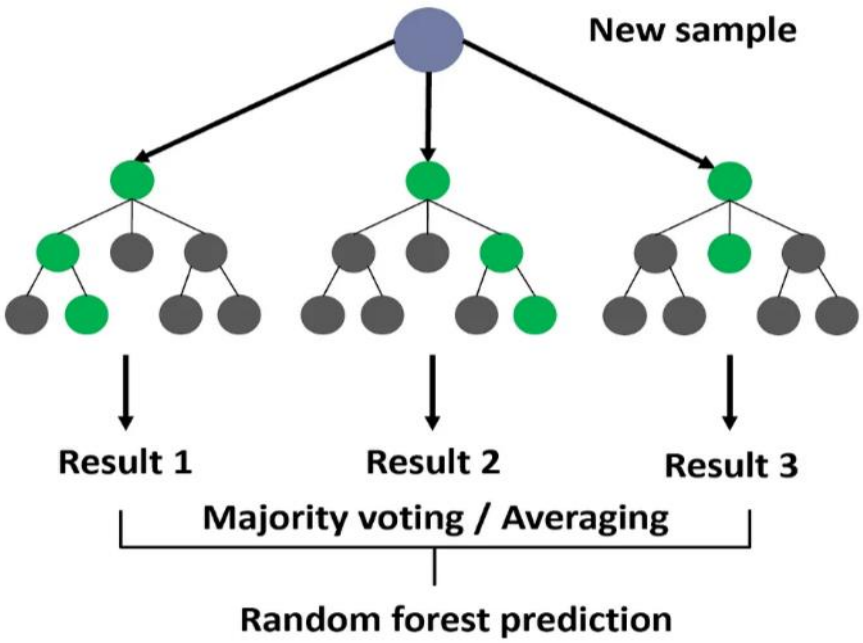


Figure (2.12) General Framework of Random Forest [70].

2.7.4.3 K-Nearest Neighbor Algorithm

K-NN [71] is a supervised learning algorithm that is used in the IDSs. The key idea of this algorithm is that it classifies the dataset based on their similarity measure with its neighbors. Its working involves storing all the cases and classifying the new case by using a similarity measure,

that is, distance measure. Usually, Euclidean distance is used to carry out the similarity between two different instances. To classify a new instance, a majority voting is performed by its closest k number of neighbors using the distance measure and it is assigned to the most common class among k number of nearest neighbors [72]. Figure (2.13) depicts KNN classifier.

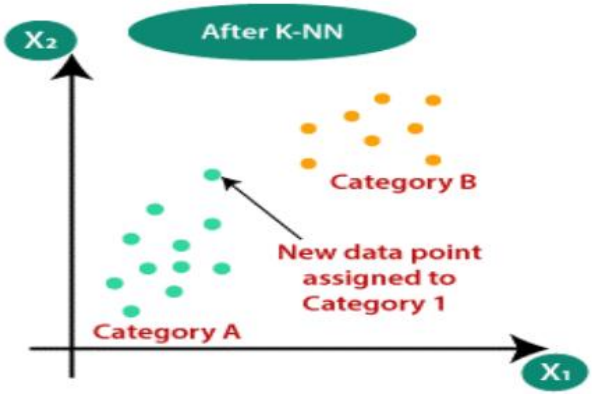


Figure (2.13) KNN Classifier [72].

2.7.4.4 LightGBM Algorithm

The LightGBM short for light gradient boosting machine. It utilizes a gradient boosting decision tree (GBDT) learning algorithm. LightGBM can be utilized for both classification and regression [73]. It generates decision trees that expand leaf by leaf, as shown in Figure (2.14). However, this approach may result in overfitting, particularly when dealing with small datasets. To mitigate this issue, it is possible to restrict the depth of trees.

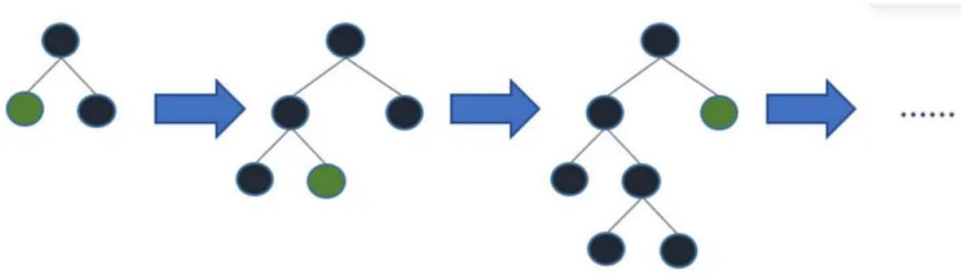


Figure (2.14) Leaf Wise Tree Grows [73].

This algorithm employs a histogram-based method that significantly accelerates the training process while maintaining high accuracy. That is, rather than working with separate data points, LightGBM initially divides the data into "bins" depend on the distribution of feature values. These bins group similar data points together, creating a histogram-like structure for each feature. This step helps reduce the complexity of the dataset while retaining essential information. During the training process, LightGBM iteratively calculates metrics, such as the information gain or Gini impurity for each bin, which represents how well a particular split separates the data into different classes. These characteristics make the LightGBM algorithm a suitable choice for several ML tasks, such as intrusion detection, especially when dealing with complex and high-dimensional datasets where speed and performance are critical [74].

2.7.4.5 Support Vector Machine

SVM is an AI technique [75] utilized to find an optimal hyperplane that minimizes the generalization error and maximizes the margin between the optimal hyper plane and the neighboring training samples in the high dimensional space. This technique does not require any information about correlation among variables during the prediction process. For data that is not linear in nature, the initial model is upgraded by incorporating a function that transforms the data into a space with higher dimensions and therefore obtaining a hyperplane line (or optimal line) that can separate the classes in space [75].

2.7.4.6 Naïve Base Algorithm

NB is a classification algorithm based on Bayes' theorem (equation 2.2) and offers a high level of prediction and scalability in various research domains with large input feature spaces. It allows the use of probabilistic

knowledge to clearly and unambiguously express its statistical components and expected results [76].

$$P(c|x) = \frac{P(c|x)P(c)}{P(x)} \quad (2.2)$$

Wherein c: class label or dependent event, x: feature or prior event, P(c) denotes the prior probability of c, P(c | x) denotes the posterior probability of x, and P(x) denotes the probability of x.

When employing this method, the classifier determines the class label of unseen data by matching the posterior probabilities for every class. The class with the maximum posterior probability value was considered the predicted class for the unseen data [76]. For instance, in the context of intrusion detection, suppose the training data D={X1,X2,X3,…,Xm} wherein every X contains m of features {X11,X12,X13, …, X1m} with class label t that denotes either normal or DoS/DDoS attack. The goal of NB model is to define the mapping function f :(X11, …, X1m) that can classify of unseen instance X= (X11,…,X1m) normal on inspection of the training data.

2.7.4.7 Logistic Regression Algorithm

LR depend on association among the features and target class [77]. Contrary to linear regression, which is utilized with continuous values, LR can be utilized with discrete values as it treats data with a different distribution and shape. Figure (2.15) presents a simple example of a data distribution that can be processed by LR algorithm utilizing the sigmoid function, while the target class (dependent variable) is illustrated by Equation (2.3) [78].

$$Y = \frac{1}{1 + e^{-x}} \quad (2.3)$$

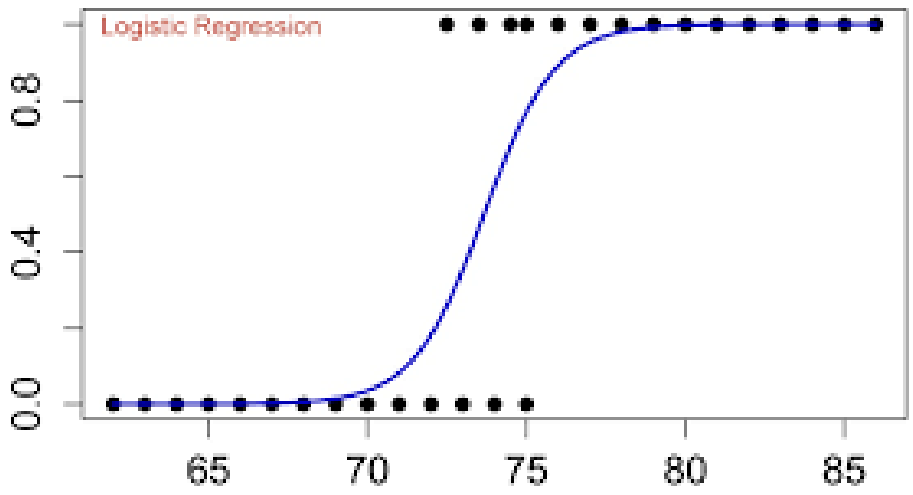


Figure (2.15) LR utilizing sigmoid function [78].

2.7.4.8 AdaBoost Algorithm

AdaBoost is short for Adaptive Boosting, that is, this algorithm initially gives equal weights (1/M) to every instance M in the training set. It then gives higher weights to every instance that is incorrectly classified. Now all data points with significant weights have been selected to build the next model [69]. The training process repeated until a lower error is attained. The update of the weight of instances and error rate with the performance of classifier are computed according to the following equations (2.4) and (2.5).

$$Err_i = \frac{1}{M}\left[\sum_{j=1}^{M} \omega_j \, I\left(p_j\right)\right] \qquad (2.4)$$

$$\alpha_i = \frac{1}{2} ln\left(\frac{1- Err_i}{Err_i}\right) \qquad (2.5)$$

Wherein: P is the predicted value and I (p) = 1 if the predicate p is true, and 0 otherwise, α_i: the performance of each model and Err_i : the error rate, and i denotes the base classifier. Note that (a_i) has a large negative value if the error rate is close to (1) and a large positive value if the error

rate is close to (0). Then, the weight has been updated to any instance through the following formula:

$$\omega_i^{j+1} = \frac{\omega_i^{(j)}}{Z_j} \times \begin{cases} exp^{-\alpha_j} & if\ C_j(X_i) = y_i \\ exp^{\alpha_j} & if\ C_j(X_i) \neq y_i \end{cases}$$

where (Zj) is the normalization features used to ensure that $\sum \mathbf{w_i^{(j+1)}} = \mathbf{1}$.

2.7.4.9 GBM Algorithm

Gradient Boosting (GB) is a boosting algorithm that combines various weak classifiers into strong classifiers, in which every new model is trained to reduce the loss function, such as cross entropy of the previous model utilizing gradient descent. In each iteration, the GB algorithm calculates the gradient of the loss function with respect to the predictions of the current ensemble and then trains a new weak classifier to minimize this gradient. The predictions of the new model were then added to the ensemble, and the process was repeated until the stopping criterion was met like number of estimators [79]. The main goal of GB is to find the optimal ensemble F(X) that minimize the loss function:

$$\min_F = \sum_{i=1}^{n} L(y_i, F(x_i)) \qquad (2.6)$$

Wherein:

- Y be the target class (the class we want to predict).
- F(X) be the current ensemble of weak classifiers' predictions.
- L(Y,F(X)) be the loss function, which computes the error between the actual target values and the predicted values.

2.7.4.10 XGBoost Algorithm

Extreme Gradient Boosting (XGBoost) is a powerful and widely used ML algorithm known for its efficiency and effectiveness in several fields, such as intrusion detection. It belongs to the family of GB algorithms, in which an ensemble of weak classifiers is constructed sequentially. Each new learner corrects the errors made by the previous ones [79]. Because most GB algorithms depend on the DT algorithm, XGBoost and CatBoost algorithms split the tree according to level-wise strategy. While, the LightGBM algorithm splits the tree based on leaf-wise approach. Figure (2.16) shows the level wise tree grows strategy.

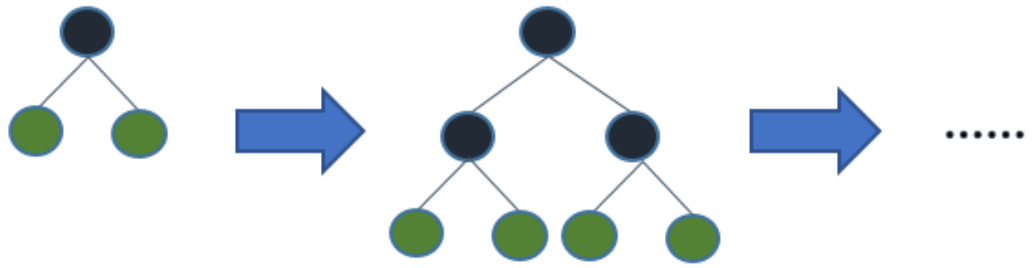


Figure (2.16) Level Wise Tree Grows [80].

2.7.4.11 Heterogeneous Ensemble Learning

The main idea behind heterogeneous is an ensemble decision relays on the aggregation of diverse base learner decisions. More precisely, the term heterogeneous denotes to implicate of diverse classification techniques to make a decision process. The attractive point of such structure is that, each base learner has been created based on different learning algorithm which is mean that each one has its own and different capability for classification task. The final output of each learner must be aggregating and combined to obtain the final result [81].

The authors in [81] referred to two essential questions that must be taken into account when designing an ensemble mechanism. First, how the base learners will be created? and second how they differ from each other? The answer to these questions eventually will determine the degree of ensemble diversity which is very important criteria for improving the ensemble mechanism. In order to obtain the final decision, there are two types of methods to combine the output of the components of heterogeneous ensemble learning:

A- Hard-Voting method: In this method, the output class is predicted according to most of the votes from all prediction classifiers. It also named as majority-voting. In equation (2.7) L is the final (or predicted) class obtained by majority votes of M classifiers. Figure (2.17) depicts the steps of aggregating the M classifiers utilizing hard-voting method [82].

$$L = Max - vote\{C1, C2, \ldots., CM\} \quad (2.7)$$

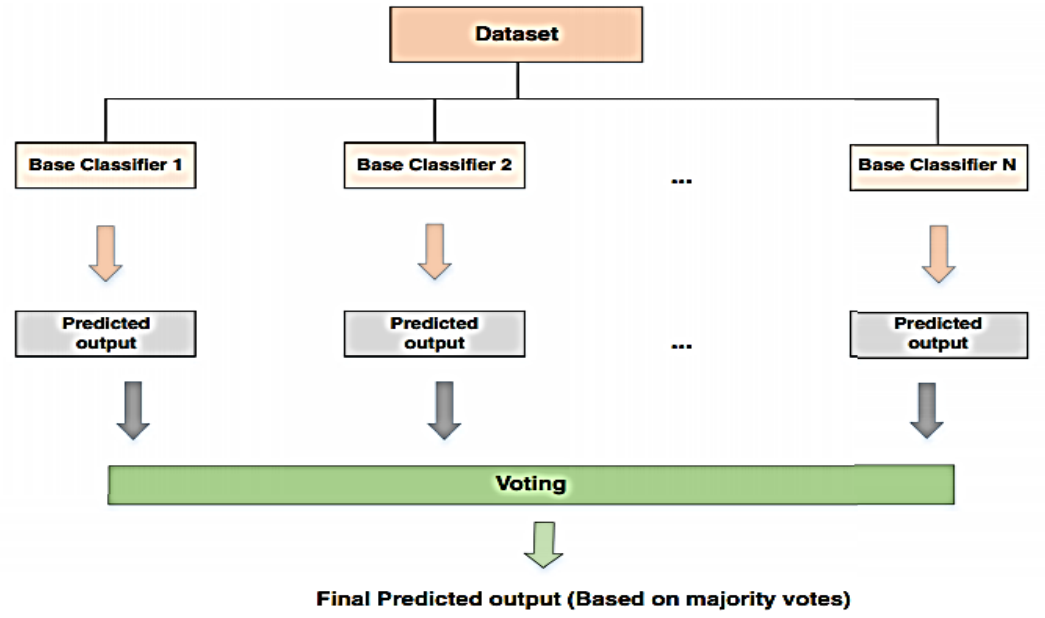


Figure (2.17) Heterogeneous Ensemble Learning Classifier [82].

B- Soft-Voting method: In this method, the output class is predicted according to the predicted probabilities for every classifier. In equation

(2.8) Y is the final (or predicted) class is calculated utilizing weight (w_j) and probability (p_{ij}) given to the classifier (Cj) for class labels i ∈ {0, 1} [82].

$$Y = argmax_i(\sum_{j=1}^{M} wj\ pij) \quad (2.8)$$

2.7.4.12 Deep Neural Network

DNN is a multi-layer feed forward neural network [83]. It consists of an input layer, number of hidden layers, and output layer. During this network, the data moves in one direction, start from the input layer, via the hidden layers, and to the output layer. That means no loops or cycles in this network. Every nodes in one layer has been direct relations to the nodes in the next layers. In order to learn the weights of DNN, the Back Propagation algorithm is utilized [84]. DNNs are popular owing to two aspects:

- They possess the capability to provide highly accurate approximations for complex, multivariate, nonlinear function directly based on input values.
- They demonstrate robust modeling capabilities across a broad spectrum of both artificial and natural phenomena.

In addition, DNNs have shown significant results in historically challenging domains of machine learning, such as enhanced self-driving cars and answering natural language queries [85]. These successes in several domains, encouraged to construct an efficient DNN-based IDS model for SDN with the goal to decrease the impact of intrusions (known DoS/DDoS and unknown attacks) while ensuring a high accuracy, high detection rate, and low FPR with an acceptable computational cost.

2.7.4.13 Long Short Term Memory

LSTM neural network was firstly suggested by [86]. It is a type of RNN developed precisely to fix the vanishing gradient problem. This problem is addressed by the LSTM being capable to respond to short and long -term dependencies. As shown in Figure (2.18), the LSTM cell is consists of four layers, named gates. The four gates are the input-gate (ik), the forget-gate (fk)، the state candidate cell (cek), and the output-gate (ok). Every gate has distinct parameters [87].

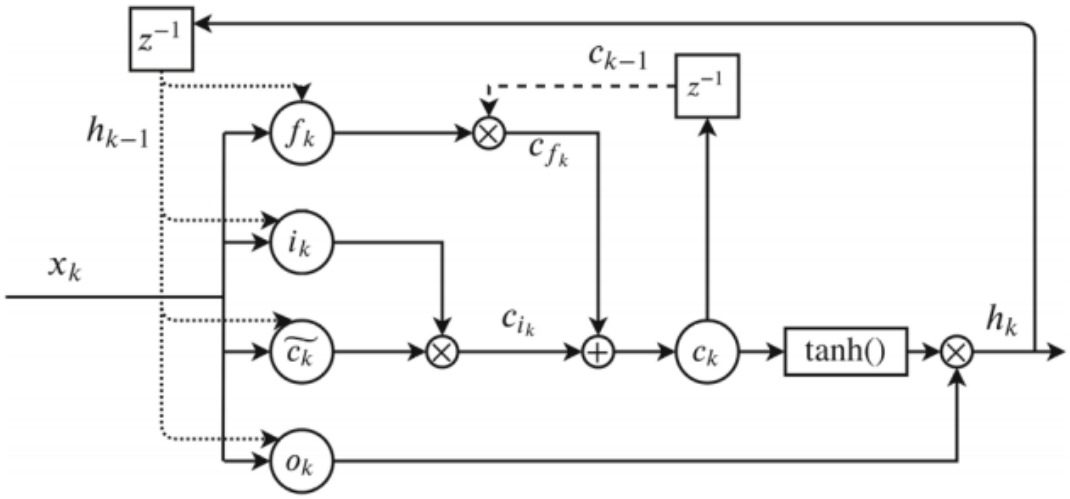


Figure (2.18) Architecture of LSTM cell [87].

The LSTM cell's behavior is as follows [87]:

The forget-gate decides which information to be remain in it or left and is computed due to Equation (2.9).

$$f_{(k)} = \sigma\left(W_{(f)}x_k + \ U_{(f)}h_{k-1} + b_{(f)}\right) \qquad (2.9)$$

Wherein $W_{(f)}$: input weight array, x_k: input vector, $U_{(f)}$: previous output weight array, h_{k-1}: previous output, $b_{(f)}$: bias.

$$c_{(fk)} = f_{(k)} \ \odot \ c_{k-1} \qquad (2.10)$$

$$\hat{c}_{(t)} = tanh\left(W_{(i)}x_k + \ U_{(i)}h_{k-1} + b_{(i)}\right) \qquad (2.11)$$

$$i_{(k)} = \sigma\left(W_{(i)}h_{t-1} + \ U_{(i)}x_t + b_{(i)}\right) \qquad (2.12)$$

$$c_{(ik)} = i_{(k)} \ \odot \ \hat{c}_{(k)} \qquad (2.13)$$

The input-gate decides which information to be remain in it or left and is computed due to Equation (2.11), (2.12), and (2.13)

Then, the output-gate computes its output based on a sigmoid function: (2.14) [17].

$$o_{(k)} = \sigma\left(W_{(o)}x_k + \ U_{(o)}h_{k-1} + b_{(0)}\right) \qquad (2.14)$$

Finally, the cell state (c_k) is updated with tanh activation function and the output of h_k is computed according to Equation (2.15)

$$h_{(k)} = o_{(k)} \odot \tanh(c_{(k)}) \qquad (2.15)$$

2.7.5 Performance Metrics

The performance of classification techniques can be measured by using the commonly used confusion matrix. This matrix summarizes the number of instances that have been tested and have been incorrectly classified using the classification model. It is possible to obtain several measures, including the Accuracy, Detection rate, and Precision, etc [88]. From it, many important evaluation measures are derived such as FPR. Figure (2.19) shows the confusion matrix for two classes. The matrix is based on the calculation of actual and predicted values, where the rows represent the actual values of the instances being tested, while the columns represent the predicted values. Five performance measures will be used to test the efficiency of the classification accuracy of the proposed

classification models (OCBM, FTDNN, and DSBHED) in this dissertation. These measures are:

Detection rate: is computed as the percentage of true positives over the sum false negatives and true positives.

$$Detection\ rate\ (recall) = \frac{TP}{TP+FN} \qquad 2.16$$

Accuracy: is computed as the percentage of correctly foreseen DoS/DDoS attacks and normal traffic over total traffic.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \qquad 2.17$$

Precision: is computed as the percentage of true positives over the sum false positives and true positives.

$$Precision\ = \frac{TP}{TP + FP} \qquad 2.18$$

F-Measure: is defined as a balance ratio between detection rate and precision.

$$F - Measure\ = \frac{2 * (detection\ rate * precision)}{(detection\ rate + precision)} \qquad 2.19$$

FPR: is computed as the percentage of false positives over the sum false positives and true negatives.

$$FPR\ = \frac{FP}{FP + TN} \qquad 2.20$$

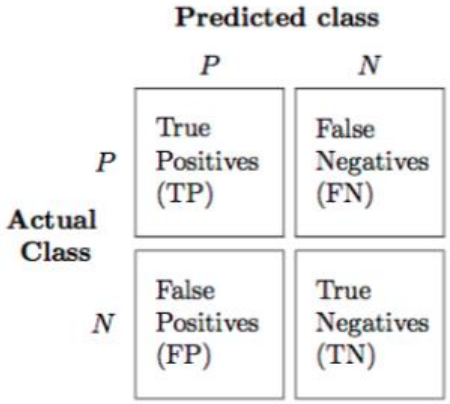


Figure (2.19) The Confusion Matrix [87].

Wherein TP: the DoS/DDoS traffic correctly predicted, TN: the normal traffic correctly predicted, FP: the normal traffic incorrectly predicted, and FN: the DoS/DDoS traffic incorrectly predicted.

Computation time: The last performance measure used in this dissertation is the computational time. This is not related directly to classification, but rather, it illustrates the training and testing times taken by a classifier. This measure gives an indication of the efficiency of the classifier [87]. The recorded training time and testing time depends on a Windows system with an i5 processor and 8 GB RAM. In order to prove the efficiency and effectiveness of the detection model, it was necessary to have a high detection rate, high accuracy, and low FPR. In addition, the test time should be as short as possible.

Chapter Three

The Proposed Intrusion Detection Models

3.1 Introduction

This dissertation introduces the OCBM, FTDNN, and DSBHED for improving the detection of attacks. In addition, a hybrid feature reduction (HFR) approach and hyper parameter optimization techniques were utilized to attain higher performance for the proposed models. The architecture and details of the proposed models are reviewed and discussed in the following sections.

3.2 Proposed Intrusion Detection Models

Figure (3.1) depicts the proposed IDS models framework, which involves machine learning and deep neural network approaches for binary detection of DoS/DDoS in SDN platform. In addition, a multi classification-based model has been proposed. The proposed framework consists of three improved models: OCBM, FTDNN, and DSBHED. The OCBM and DSBHED are developed machine learning models, whereas the FTDNN is an improved architecture of deep neural network.

The proposed models include four main stages to achieve their objectives: The first stage is data preprocessing, which contains data cleaning, and label encoding techniques were performed on the four datasets utilized in this dissertation. The second stage is to construct an HFR approach, which is a hybrid feature reduction model. The HFR is applied in two manners: binary and multi-classification. HFR is used to obtain the best subset of features from the original space via the proposed majority-voting and mean techniques. Then, data normalization is performed on the best subset of features. The third stage contains the construction of prediction models: OCBM, DSBHED, and FTDNN. The OCBM is an optimized CatBoost model trained on the best subset of features and utilizes the GSCV optimization technique to obtain an accurate and efficient model that requires less computational and time

complexity in terms of resource-constrained platforms. The DSBHED is a soft-based heterogeneous ensemble model for detecting different attacks. This model is based on a blend of numerous ML-based classifiers for final detection. The FTDNN classifier, which is based on a deep neural network model, was constructed and its hyperparameters were fine-tuned using the RSCV optimization technique. It is also trained on the output of the HFR approach.

The fourth stage is the assessment of the performance of the developed models based on assessment metrics, for instance detection rate, accuracy, precision, F1-score, and FPR, as well as the training and test times were also computed.

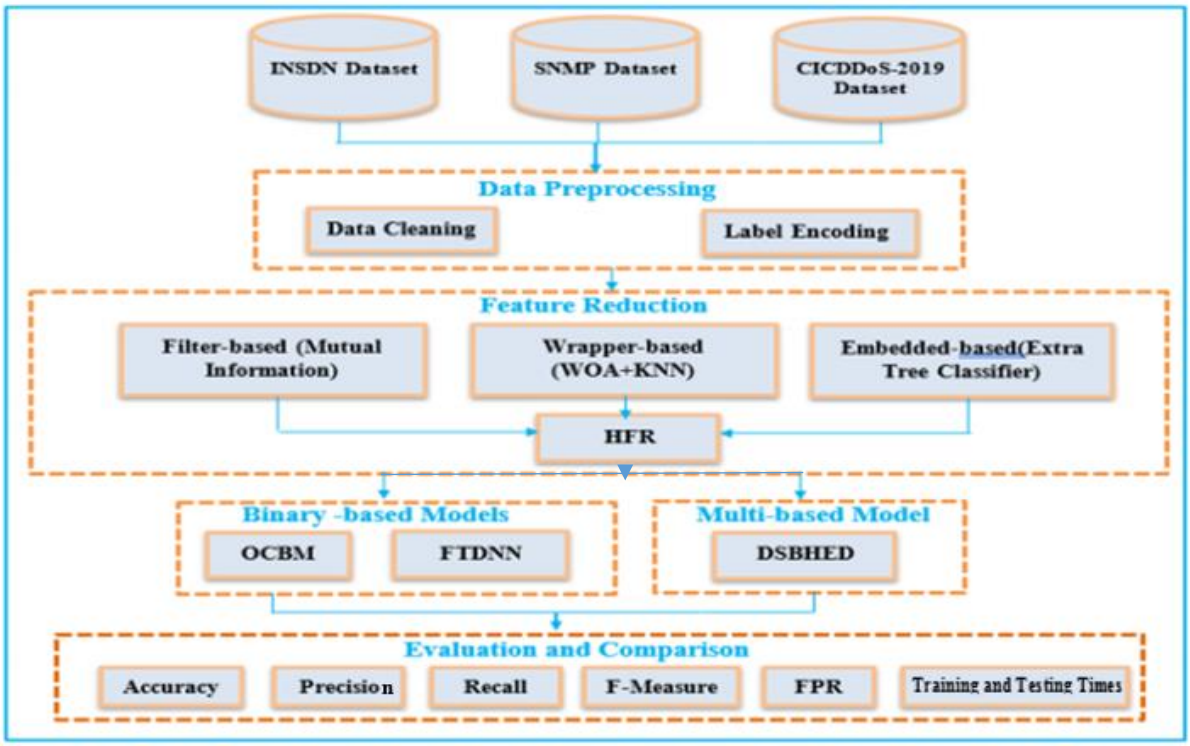


Figure (3.1) The Proposed Models Framework.

3.3 Traffic Capture

The objective of this phase is to capture the data traffic for security purposes. The traffic capture phase (TCP) utilizes network sniffing tools such as Tcpdump and Wireshark to gather raw packets for security

purposes. In principle, the TCP in this dissertation is similar to traffic sniffing or capturing utilized by Elsaied et al. [20] to prepare the InSDN dataset, and by the researchers of the CICDDoS-2019 and CICIDS-2017 datasets. Thus, we decided to utilize the most up-to-date InSDN, SNMP, CICDDoS-2019, and CICIDS-2017 datasets, rather than our network data capture. Because this allowed us to compare with the outcomes of related studies stated in the literature. The features of the datasets were then fed to the data preprocessing stage to be transformed into a suitable format for ML techniques.

3.4 Data Preprocessing Stage

Data preprocessing is primarily intended to prepare data in a suitable form for ML techniques. In this dissertation, the preprocessing was performed using the following steps: data cleaning and label encoding. In the data-cleaning step, duplicate data and features that were not necessary in the prediction process were removed. For instance, the CICIDS-2017 dataset contains two identical features named "Fwd Header Length." Consequently, one of these features was removed to avoid redundancy. In addition, handling the infinity, and Not a Number (NaN) values, was replaced by 0's for all utilized datasets. Then, the utilized datasets include continuous, binary, and symbolic values. For example, the attribute 'protocol' in the InSDN dataset contains symbolic worth for instance: “udp,” “icmp,” and “tcp.” Because various models receive only numerical inputs, the label encoding step is regarded as crucial and has an important effect on IDS detection accuracy. In this step, the one-hot encoding method was used to deal with non-numerical features. Algorithm (3-1) demonstrates data preprocessing steps.

Algorithm (3-1) Data Pre-processing

Input: *Two-dimensional array* $DS_{(i,j)}$ *where i is the number of examples and j is the number of features.* ***Unnecessary features list*** *contains {ID, Flow ID, Source IP, Destination IP, Source Port, Destination Port}.*

Output: *Two-dimensional array D- Prepared* $_{(i,j)}$ *after pre-processing.*

// Data Cleaning

Begin

1 for n = 1 to i

2 for m = 1 to j

3 if feature n is redundant or unnecessary then

4 drop feature n ***// To avoid produce a biased model***

5 end if

6 if values val in feature n is missing or NaN or infinity then

7 set val = 0

8 end if

9 if values val in feature n is categorical then

//Convert non-numeric features to the numbers utilizing:

10 Apply Label-Hot Encoding () method

/ this process is utilized to convert non-numeric features to the numbers for instance transform protocol types such as TCP, and UDP to numerical data*/*

 # The Label –Hot Encoding Steps

Create an empty DataFrame to store the encoded data.

Initialize a LabelEncoder object.

For each categorical feature (i.e., Protocol) in the original DataFrame:

Extract the categorical feature values (i.e., TCP and UDP).

Encode the categorical feature values using the LabelEncoder object.

Create a new column in the encoded DataFrame and store the encoded values.

Return the encoded DataFrame.

11 end if

12 end for

13 end for

14 return D-Prepared
End

3.5 Hybrid Feature Reduction Stage

Feature reduction plays a significant role in the development of ML and DNN classifiers, particularly for IDS. It aids in decreasing the number of features in the datasets, building and testing times, and computation costs as well as enhancing the overall performance of the models. Network traffic datasets, for instance, typically contain tens of features. This large number of features has made it difficult to apply ML techniques efficiently. In addition, a significant number of dimensions within the original dataset may lack relevance or contain noisy data for the particular analytical task being undertaken, resulting in an increase in classifier complexity.

Therefore, the best subset of features were selected utilizing the HFR approach. The proposed approach, HFR, covers all feature reduction approaches (filter-based, wrapper-based, and embedded-based) in addition to the majority-voting and mean techniques were proposed to obtain the best subset of features.

3.5.1 Majority-Voting Technique

At this stage, rather than utilizing an only feature reduction algorithm, this dissertation utilizes three different FR algorithms wherein Mutual Information (MI) from filter-based approach, Whale Optimization Algorithm (WOA) from wrapper-based approach, and Extra Tree Classifier (ETC) from embedded-based approach to assess features. Due to these approaches, the features are ranked in three diverse lists independently. After that, the majority-voting technique (e.g., each feature had two or three votes has been selected) has been utilized to

neglect the insignificant and inconsistent features and provide a reduced dataset.

In other words, three different approaches have been utilized to prune the less informative features individually, as follow:

- Due to MI method, which calculates the mutual information between features and the target variable, all less informative features have been eliminated. This is accomplished via an experimental threshold.
- Due to WOA, which based on a search approach to choice candidate subset of features which are assessed by using an objective function. A search approach is therefore required to guide the feature subset choice process as it discovers the space of all possible feature combinations. The object function then evaluates candidate subset of features and provides a measure of their quality.

  Therefore, the WOA was combined with KNN algorithm, to exploit the power of optimization capabilities to search for subset of features while utilizing KNN for accurate evaluation of the selected features.
- According to ETC, that measures the importance of each feature based on how much it contributes to reducing the impurity at each split. Due to this method, the importance values ranging from (0) to (1), so less informative features have been removed depend on an experimental threshold.

Consequently, the aim of this stage is to consider different perspectives in the context of feature reduction. By exploring different algorithms, we can get a more comprehensive understanding of how to effectively reduce the number of features in datasets. To mitigate the

algorithms limitation, a MVT has been utilized to facilitate the identification of features that consistently appear as important across different algorithms. In general, the features selected by two or three feature reduction techniques tend to be more effective than the features that are eliminated by all or two of techniques.

3.5.2 Mean Technique

Due to the previous step (MVT), each feature has three weights. Thus, the mean technique has been utilized in order to assign one weight for every feature and measure the importance of feature in relation to the rest of the features wherein every feature having a weight value greater than an experimental threshold has been chosen. The output of this stage is the best subset of features (significant features).

In summary, the proposed HFR approach allows us to identify the best subset of features that enhances the model's performance, decreases high-dimensional features, and improves the computational cost. For more details, see Algorithm (3-2) and figure (3.2).

3.6 Data Normalization

The output of the HFR is an array consisting of the best subset of features; therefore, in the normalization step, the feature value (the best subset of features) is mapped into a normalized form to range (0 – 1) through calculating the maximum and minimum values for every feature, thereby implementing the mathematical formula. Thus, the main aim of this step was to avoid dominant features with high values in the classification performance. Algorithm (3-3) demonstrates data normalization step.

Algorithm (3-2) Hybrid Feature Reduction Approach
Input: *Two dimensional array* $Pr_{epared\ (i,\ j)}$ *wherein i: number of examples and j: number of features, an experimental threshold* Θ*, flag denotes as an indicator assigned to each feature to determine its status.*
// Output of Algorithm (3-1)
Output: *The best subset of features (OSOF) from the proposed Hybrid Feature Reduction approach.*
Begin
// Applying Mutual Information method to evaluate features
for f = 1 to j
Calculating weight of feature f utilizing Mutual Information method
if weight < Θ then
eliminate this feature
flag_[f]=false
else
W_MI[f]=weight
flag_[f]=true
end if
end for
// Applying WOA with KNN classifier to evaluate features
for f = 1 to j
W_WOA[f]= evaluating weight of feature f utilizing WOA with fitness function (KNN)
if (feature f ∈ W_WOA[f]) then
flag_[f]=true
else
flag_[f]=false
end if
// Applying Extra Tree classifier to evaluate features
evaluating weight of feature f utilizing Extra tree classifier
if weight < Θ then
eliminate this feature

20	*flag_[f]=false*
21	*else*
22	*W_ETC[f]=weight of classifier*
23	*flag_[f]=true*
24	*end if*
25	*end for*
	// Applying majority -voting approach
26	*for f = 1 to j*
27	*if (f ∈ at least two out of three lists [W_ETC[f], W_WOA[f] W_MI[f]])then*
28	*flag-MV[f]=true*
29	*else*
30	*flag-MV[f]=false*
31	*end if*
	// Applying mean approach
32	*for f = 1 to j*
33	*if (flag-MV[f] = true) then*
34	*W_mean[f]= (W_ETC[f]+ W_WOA[f]+W_MI[f])/3*
35	*else*
36	*eliminate this feature*
37	*end if*
38	*end for*
39	*for f = 1 to j*
40	*if (W_mean[f] ≥ Θ) then*
41	*OSOF= W_mean[f]*
42	*else if*
43	*eliminate this feature*
44	*end if*
45	*end for*
46	*end for*
47	*return OSOF*
	End

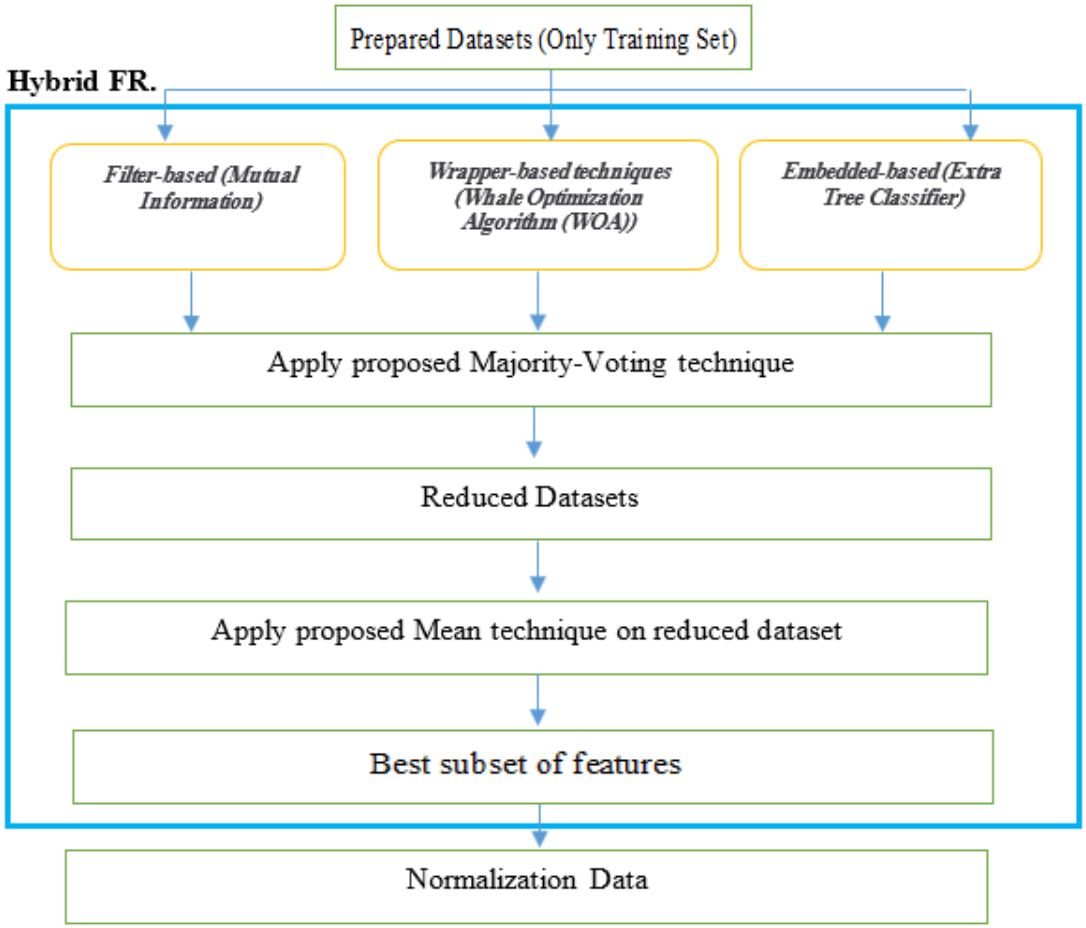


Figure (3.2): Workflow of HFR approach.

Algorithm (3-3) Data Normalization
Input: *Two dimensional array* $D_{(i,j)}$ *wherein i: number of examples and j: number of features* ***// Output of Algorithm (3-2)***
Output: *Two dimensional array* (D_{ij}) *of normalized dataset.*
1 Let Max [j] and Min [j]are two arrays holding (maximum and minimum values)
2 for k = 1 to j
3 set max and min to the initial value of feature k
4 for t = 1 to i
5 if v_{kt} *< min then* ***// vkt : represents the value of a particular feature k for a specific data point t***
6 min = v_{kt}
7 else if v_{kt} *> max then*

8	*max = v_{kt}*
9	*end if*
10	*end if*
11	*update each value v in feature k due to the equation:* $\widehat{V} = \frac{V - min_A}{max_A - min_A}$ **[2.1]**
12	*end for*
13	*Store results in D_{ij};*
End.	

3.7 Hyperparameter Optimization Stage

Without proper hyper parameters tuning, most ML/DNN algorithms cannot attain best results. Selecting a powerful ML/DNN algorithm is a crucial initial step towards creating a highly accurate model that can be complemented by appropriate hyper parameters. Manual hyper parameters optimization can be time-consuming, especially when dealing with numerous parameters that require adjustment. For the proposed ML-based algorithms in this dissertation (OCBM, ORF, OLGBM), the major difficulties lie in determining the best selection of the number of trees, maximum depth of each tree, learning rate, and so no. While for the proposed DNN-based model (FTDNN), a significant challenge arises in identifying the appropriate architecture and its configurations, including the choice of activation function, number of layers, number of neurons in each layer, learning rate, and optimizer algorithm. In this dissertation, the GridSearch technique with cross-validation is utilized for the proposed ML-based classifiers to fine-tune their hyper parameters in order to produce a lightweight model. While for the proposed DNN-based classifier, a random search with cross-validation was used to attain the best configuration of the DNN-based model (see Figure (3.3)).

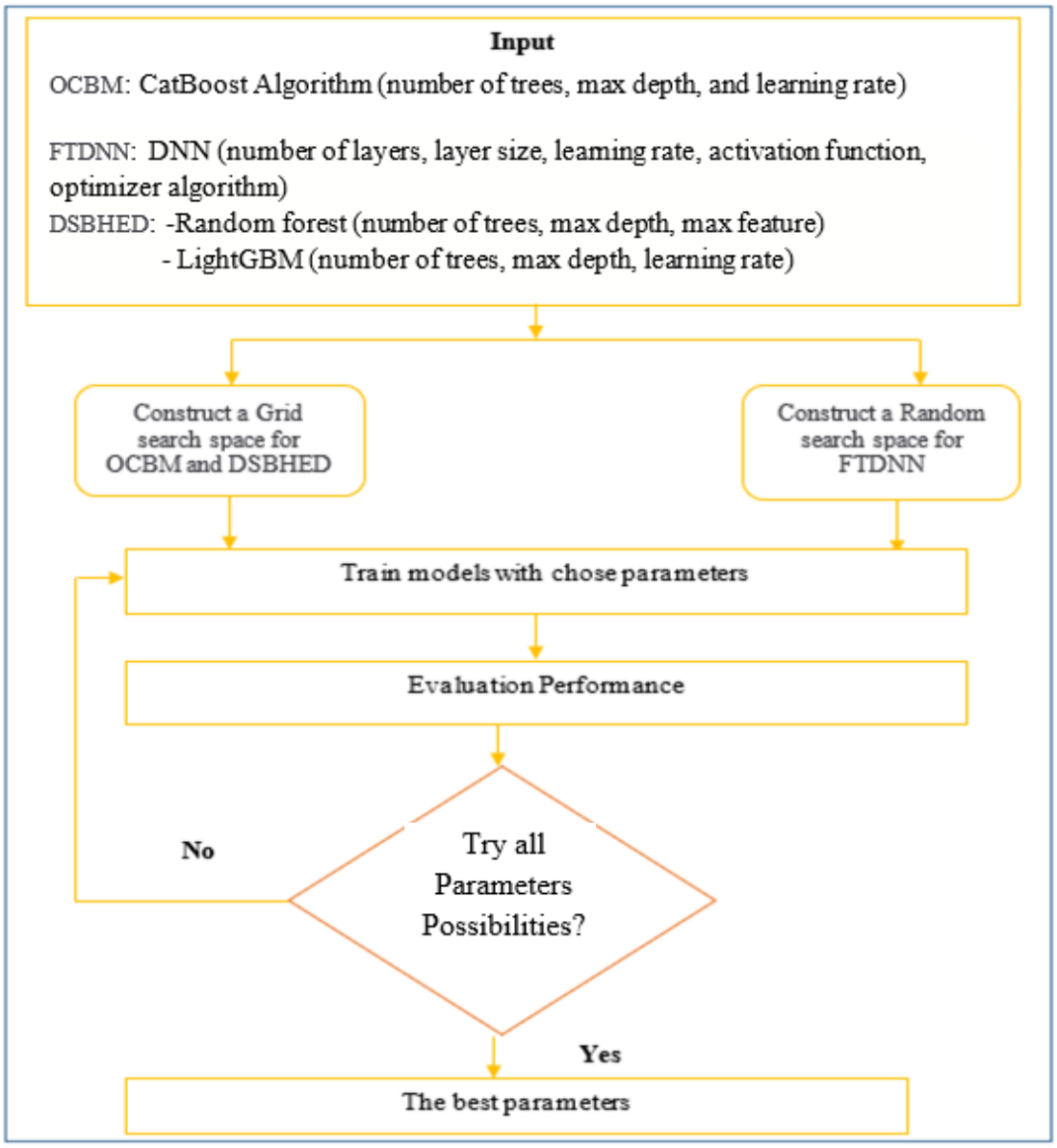


Figure (3.3): Hyperparameters Tuning.

A- Optimized ML-Based using GridSearchCV

In order to construct an effective and accurate IDS, it is important to find the right balance between predictive power and model complexity. By utilizing fewer trees, such as in the case of the CatBoost classifier, it is aid to build a lightweight model that can efficiently identify assaults in real-time networks. This approach helps reduce the training and testing

times and computational resources required for detection, while still maintaining a high level of performance.

To achieve this, the GridSearch (GS) technique with cross validation (CV) is proposed to find the best hyper-parameter values of classifiers. The GSCV technique involves systematically assessing the effect of different hyperparameter combinations on the ML model's performance by iterating through all the possible hyperparameter choices.

In order to precisely select the hyperparameter values (number of trees, maximum depth, maximum feature, and learning rate) of the proposed ML-based models utilizing M-fold CV. Initially, the prepared data is split into M subsets, where M is set to 5. During this process, one of the subsets is used as the testing data, whereas the remaining M-1 subsets are used for training. This process is repeated M times, and the average performance across all M iterations are computed.

In this dissertation, the proposed models include a CatBoost classifier for binary prediction, while the Random Forest and LightGBM classifiers are components of the DSBHEP model for prediction multi-class attacks. Table 3.1 illustrates the search space for hyperparameters of the proposed ML-based models. The range values are intentionally selected to be less than the default values, aiming to construct lightweight models.

Table (3.1) The hyper parameters of the ML-based proposed models

Classifier	Hyper-parameter	Range	Best value
CatBoost	No. of trees	[200-500]	300
	Max-depth	[2-5]	5
	Learning rate	[0.01, 0.05, 0.1, 0.2, 0.3]	0.1

Classifier	Hyper-parameter	Range	Best value
RF	No. of trees	[60-95]	80
	Max-depth	[1-5]	4
	Max-feature	(sqrt, log2)	sqrt
LightGBM	No. of trees	[55-95]	85
	Max-depth	[2-5]	5
	Learning-rate	[0.01, 0.04, 0.05, 0.3]	0.04

B- Optimized DNN-based using RandomSearchCV

In the context of DNN-based model, RandomSearchCV (RSCV) is a valuable tool used to obtain the best model structure by efficiently exploring the hyper parameter space. In this dissertation, applying the RSCV technique allowed us to define the best structure with its parameters like the number of layers, number of neurons in each layer, learning rate, activation function, and optimization algorithm for FTDNN model. The RSCV technique randomly selects combinations of these hyper parameters and trains the proposed model using m-fold cross-validation. Afterwards, the best structure of the proposed FTDNN model are obtained. As a result, using the RSCV technique as well as optimization methods such as dropout layer, early stopping technique, and batch normalization layer aid in building reliable and accurate DNN-based model.

Table (3.2) The hyper parameters of the DNN-based proposed model

Hyper-parameters	Range	Best value
Number of layers	[2-5]	2
Number of neurons	(64,128,256)	128

Hyper-parameters	Range	Best value
Learning rate	(0.1,0.01,0.001,0.0001)	0.001
Activation function	(Relu, Tanah)	Relu
Optimizer algorithm	(Adam, RMSprop)	Adam

3.8 Prediction Stage

This stage denotes the significant step in the proposed framework, as it involves the building of the classification models. When the best subset of features is selected through the proposed HFR approach, and the best values for hyperparameters of proposed models are obtained, they are then fed into the classification phase (training stage). As shown in Figure (3.4) the prediction stage consists of two prediction models. The OCBM and FTDNN are used for binary detection of DoS/DDoS attacks.

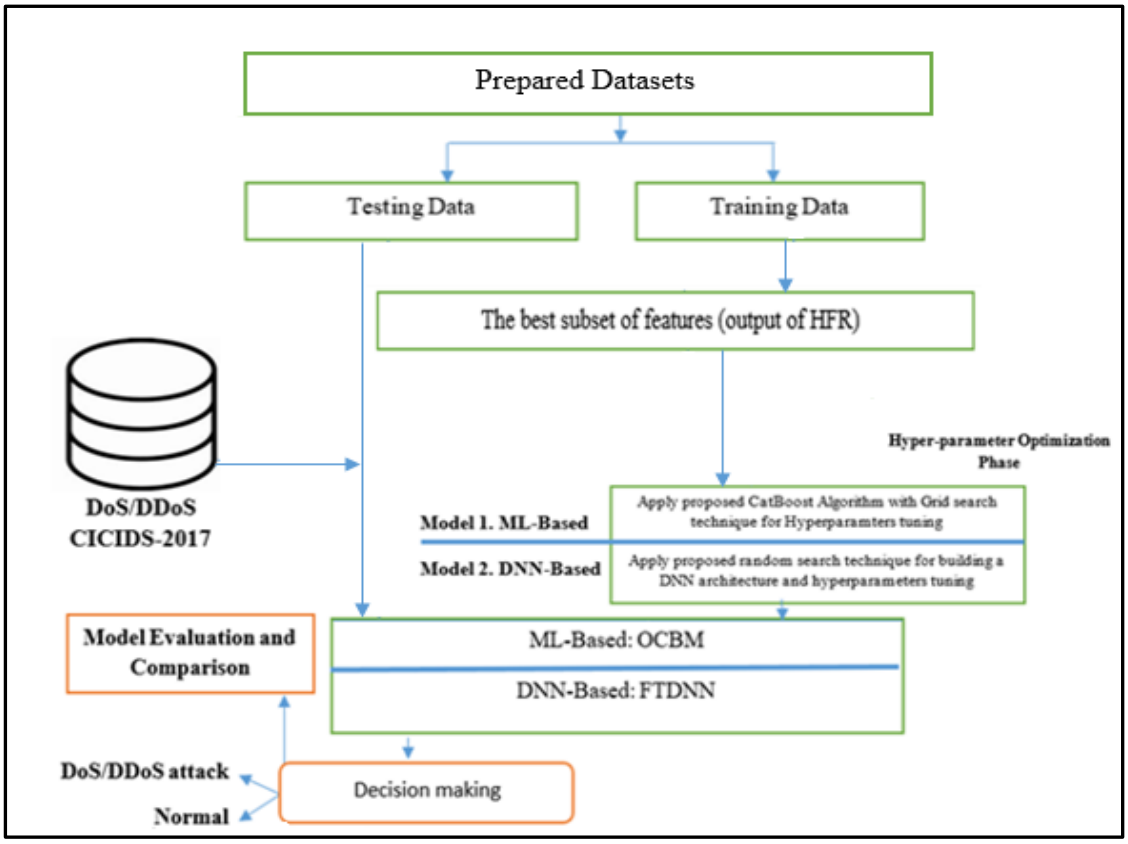


Figure (3.4): Prediction Stage.

3.8.1 An Optimized CatBoost Model

In order to improve the detection capability of IDS ML-based, the OCBM model was proposed in this dissertation for binary detection of DoS/DDoS attacks. Initially, the best subset of features which outputs from HFR approach was fed into the CatBoost for training, and the hyperparameters of the CatBoost were optimized using GSCV technique to obtain the OCBM classifier. Afterwards, the OCBM is trained on the InSDN, SNMP, CICIDS-2017, and CICDDoS-2019 datasets. Finally, the performance of the OCBM was assessed, and its outcomes were promising due to other ML algorithms and previous studies conducted on the same datasets. To the best of our knowledge, this is the first attempt to utilize the CatBoost model to construct an intrusion detection model for an SDN platform. Pseudo-code for the OCBM with GSCV technique is shown in Algorithm (3-4).

Algorithm (3-4) Optimized CatBoost Model
Input: *Two dimensional array* $D_{(i,j)}$ *, wherein i: number of examples and j: number of features* ***// Output of Algorithm (3-3)****, CatBoost algorithm (CBA);* Θ *(hyperparameters space), Number of folds (m), GSOT (Grid Search optimization technique).*
Output: *Evaluation metrics.*
// Getting the best hyper-parameters for ML-based model using GSOT
1 *hyperparameters space={* *No of trees: list(range(200,300,400,500))* ***//the number of trees.*** *Max-depth: list(range(2, 5))* ***// the maximum depth of tree.*** *Learning-rate: list(0.01, 0.05, 0.1, 0.2, 0.3)* *}*
2 besthyper= Null
3 for i= 1 to N do ***// N=40***
4 accuracy= GSOT (CBA, Θ_i *,* D_{Train} *, CV=m)*
5 if accuracy is the maximum so far then

6	*besthyper= Θ_i*
7	*end if*
8	*end for*
	*// **Applying an optimized CatBoost model***
9	*Train OCBM on D_{Train} using the best hyper-parameters (besthyper) obtained and compute train time.*
10	*Compute evaluation metrics for OCBM such as accuracy , detection rate, precision, FPR, F1-score, and test time on D_{Test}*
End.	

3.8.2 A Fine-Tuned Deep Neural Network Model

Developing an efficient security solution to safeguard SDN platform from DoS/DDoS threats is a challenge because DoS/DDoS utilizes diverse attack approaches. Moreover, most of the current DNN-based models suffer from high computational load or may not perform well in detecting the newly reported DoS/DDoS assault, which utilizes out-of-date datasets for building IDS. To address these issues, we propose FTDNN, a hybrid IDS approach that combines an HFR model of feature reduction with a DNN classifier, in addition to utilizing modern datasets for training and assessment.

However, various factors, such as the network structure and its configurations, play significant roles in constructing an effective DNN-based IDS model. Consequently, there is a need for techniques that optimize and develop the DNN model to achieve the best possible hyper-parameters and structure.

Therefore, one of the goals of this dissertation is to determine both the best structure of the DNN model and its features to attain the highest performance of the proposed FTDNN model using a RSCV technique.

Moreover, to further improve the robustness of DoS/DDoS detection model, the output of the proposed HFR approach which efficiently pick up the best subset of features integrated with the FTDNN model resulting in reduced training and testing times and improved model performance for binary detection of DoS/DDoS attacks, as will be described in Chapter 4, compared to previous works. For more details, see Algorithm (3-5).

Algorithm (3-5) Fine-Tuned DNN Model
Input: *Two dimensional array* $D_{(i,j)}$ *, wherein i: number of examples and j: number of features* ***// Output of Algorithm (3-3)****, DNN classifier (DNN); Θ (hyperparameters space), Number of folds (m), RSOT (Random Search optimization technique).*
Output: *Evaluation metrics.*
// Getting the best hyper-parameters for DNN-based model using RSOT
1 *hyperparameters space={* *Number of layers : list(range(2, 5))* *Number of neurons : list(64, 128, 256)* *Activation function : list(Relu, Tanah)* *Optimizer algorithm: list(Adam, RMSprop)* *Learning rate: list(1.0, 0.1, 0.01, 0.001, 0.0001)* ***}***
2 bestconfig= Null
3 bestaccuracy= 0
4 for i = 1 to max_iter do
5 h_i= select a random hyper-parameter from hyperparameters search space.
6 accuracy= fitness (RSOT, DNN, h_i , D_{Train} , CV=m)
7 if accuracy > bestaccuracy then
8 bestconfig = h_i
9 bestaccuracy = accuracy
10 end if
11 end for

*// **Constructing the FTDNN model using the best hyper-parameter (bestconfig)***
12 Feeding the best subset of features (D_{Train}) into fully connected layer (128 neurons) with ReLU
13 Adding dropout Layer (the probability is 0.02)
14 Adding Bach Normalization Layer()
15 Adding fully connected layer (128 neurons) with ReLU
16 Adding dropout layer (the probability is 0.02)
17 Adding Bach Normalization Layer()
18 Adding classification layer (N-neuron) with softmax function // N:No. of classes.
19 Updating parameters of the FTDNN model utilizing (binary cross-entropy loss) function with the Adam optimizer and train the DNN network.
20 Compute evaluation metrics for FTDNN such as accuracy , detection rate, precision, FPR, F1-score, and processing time on D_{Test}
21 Return Model-Performance
End.

Figure (3.5) describes the architecture of the proposed FTDNN classifier, which obtained utilizing the RSCV technique. Initially, it is essential to determine the number of neurons for each layer of the DNN model (FTDNNM). The number of neurons in the input layer is determined based on the size of the input data or features (the output of proposed HFR approach that is, the best set of features). The output layer (y_i) consists of neurons that represent the target labels of the training data (in our case normal and DoS/DDoS attack for binary prediction). The output (y_i) can be achieved by the following steps:

- The input values from the input layer are multiplied by the corresponding weights of the connections and passed through activation function (the ReLU is applied) in the hidden layers.

- The output from the last hidden layer is propagated to the output layer. Again, the output values are multiplied by the weights of the connections and passed through an activation function (the softmax function is implemented).
- The final output values from the output layer represent the predicted values (y_i) for the given input x_n.
- Through the training phase of supervised learning, a loss function (binarycross-entropy) is introduced to compute the error between the desired output and the predicted output.
- The backpropagation phase is a fundamental step in training DNN. It involves calculating and updating the gradients of the network's weights and biases by propagating the error backwards through the network.

During the training phase, it is common to face the challenge of overfitting. Consequently, the model that has been trained tends to exhibit high accuracy when evaluated on the training data, but it often performs poorly when presented with unseen data. To address this issue, various techniques have been introduced in this dissertation, including the use of dropout, batch normalization, and early stopping.

- Dropout technique is used to mitigate overfitting. It randomly drops out a certain percentage of neurons during each training iteration, forcing the deep network to learn more robust and generalized representations. Dropout helps to prevent the model from relying too heavily on specific neurons and encourages the learning of more diverse features.
- Batch normalization method was applied to normalize the corresponding output values to prevent values that are either too large or too small, thus speeding up the training process.

- Early stopping is a technique that monitors the model's performance during training process. Training is stopped early when the performance of the DNN starts to degrade or does not improve significantly. This prevents the model from over-fitting on the training data and allows it to generalize better to unseen data.

By employing these techniques, the overfitting problem can be alleviated, resulting in a model (FTDNN) that not only performs well on the training data but also generalizes effectively to unseen data.

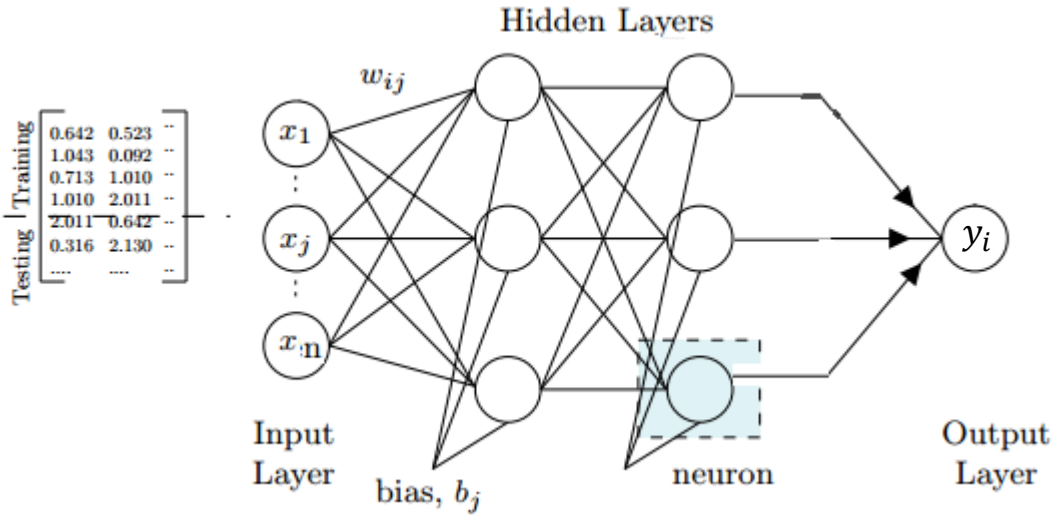


Figure (3.5) Deep neural network architecture of FTDNN.

3.9 Zero-Day DoS/DDoS Attack Prediction and Retraining

ML and DNN based models provide effective solutions for detecting DoS/DDoS attacks by attempting to identify patterns of malevolent traffic in SDN. Also, previous works have shown the power of these models in detecting attacks with known profiles, for instance, DoS/DDoS attack profiles with which the ML- and DNN-based techniques have been trained. Nevertheless, their detection performance against zero-day (or unknown) DoS/DDoS attacks or attacks with

dynamically altering profiles has not been widely examined. Given the rising complexity of DoS/DDoS attacks on SDN-based resources, it is vital to understand how the proposed models can be carried out in such situations and to what range can handle abnormalities from their training classifiers.

Thus, to prove the efficiency of the OCBM and FTDNN, for binary detection of DoS/DDoS and discover zero-day attacks. The proposed models were separately trained using two benchmark InSDN and CICDDoS-2019 datasets, and then the CICIDS-2017 dataset was utilized to simulate a real-life attack and evaluate the performance of the OCBM and FTDNN models. Afterwards, the proposed models were separately retrained by merging the CICIDS-2017 with InSDN datasets and CICIDS-2017 with CICDDoS-2019 datasets to verify the performance of the models. As a result, the retraining process should be iteratively carried out in order to enhance the performance of the proposed models against zero-day assaults, as illustrated in Cases (1-4) for each binary-based proposed model in Chapter 4.

3.10 Developed Soft-Based Heterogeneous Ensemble Detection Model

Figure (3.6) illustrates the general architecture of the proposed framework, which includes the developed model called DSBHED (Developed Soft-Based Heterogeneous Ensemble Detection). The goal of this model is to improve the efficiency of detection multi-class attacks such as UDP, SYN, and others against the SDN architecture. This model based on a blend of numerous ML-based classifiers for final detections.

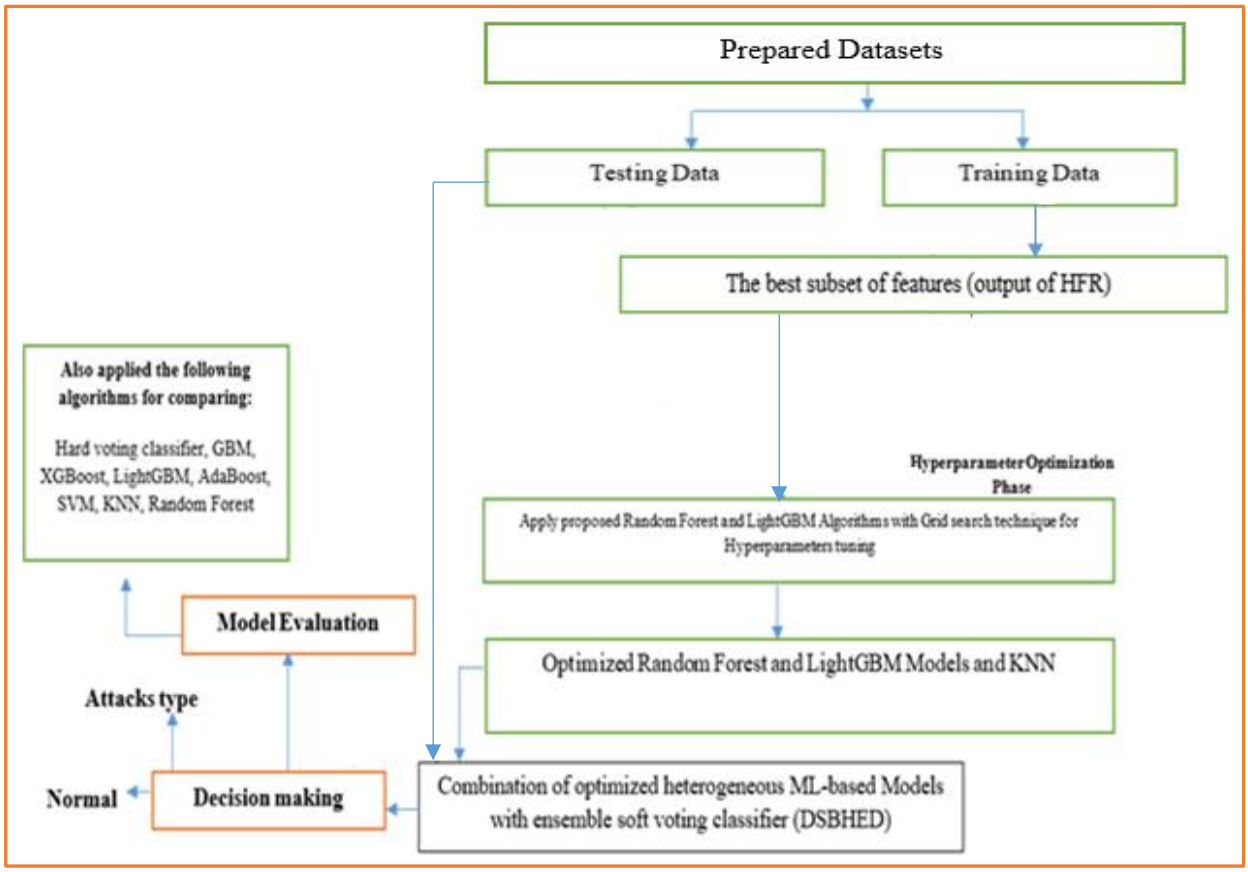


Figure (3.6) The Developed Soft-Based Heterogeneous Ensemble Framework.

There are two essential aspects are involved in constructing an efficient heterogeneous ensemble learning model. The first aspect concerns the selection of single classifiers, and the second involves a combination strategy of these single classifiers. These aspects are discussed below:

(i) The classifiers in an ensemble model should be diverse from one other; otherwise, there is no advantage to merging them. Therefore, the components of the DSBHED model include three different ML-based classifiers, namely, Random Forest (RF), K-Nearest Neighbor (KNN), and LightGBM.

(ii) The soft-based technique was selected as the combined strategy for all individual classifiers.

Since the InSDN, SNMP, and CICDDoS-2019 datasets cannot be utilized for building the DSBHED model directly, data preprocessing is required. As illustrated in Figure (3.1) through data pre-processing, we performed the techniques involving data cleansing and label encoding. The prepared datasets are then fed into the HFR stage, which consists of four steps. First, each dataset was divided into training set and test set, where the training set was utilized for feature reduction. Then, we applied the MI, WOA, and ETC techniques to assess the features separately. Second, the majority voting technique is implemented in this step to prune the subset of features. Third, because of the output of the previous steps, each feature has three weights. Thus, to unify these weights, the mean value was calculated for every feature in the reduced datasets.

After performing an HFR approach and obtaining a more concise and informative best subset of features, we utilized three different heterogeneous classifiers, ORF, KNN, and OLightGBM, in terms of individual classifier selection to predict the output labels. Furthermore, to increase the efficiency of the DSBHED model and make it more appropriate for real-time detection, its components, such as the ORF and OLGBM classifiers, were optimized using the GSCV technique. Finally, the output of every classifier fed into the soft-based technique, which predicts the output label for input instances, depends on the highest probability of the output labels across all classifiers. For more details, see Algorithm (3-6).

As a result, the DSBHED classifier focuses on combining the insight of multiple classifiers trained to solve the same issue, using their opinions to reach an accurate and efficient decision. The error of one classifier in the heterogeneous ensemble classifiers are compensated by the other individuals of ensemble model, thus ability of heterogeneous

ensemble model is usually much stronger than that of an individual classifier.

Algorithm (3-6) Developed Soft-based Heterogeneous Ensemble Detection Model

Input: *Two dimensional array* $D_{(i,j)}$ *, wherein i: number of examples and j: number of features* ***// Output of Algorithm (3-3)****, Random Forest algorithm (RFA), LightGBM algorithm (LGBMA), KNN algorithm;* Θ *(hyperparameters space), Number of folds (m), GSOT (Grid Search optimization technique).*

Output: *Evaluation metrics.*

// Getting the best hyper-parameters for ML-based model using GSOT

1 *hyperparameters space={*

RF classifier	*No. of trees:*	*[60-95]*
	Max-depth:	*[1-5]*
	Max-feature:	*(sqrt, log2)*
LightGBM classifier	*No. of trees:*	*[55-95]*
	Max-depth:	*[2-5]*
	Learning-rate:	*[0.01, 0.04, 0.05, 0.3]*

}

2 besthyper= Null

3 for i= 1 to C do *// C=2, No. of Classifiers*

4 for each hyperparameter h_i *in hyperparameters space* Θ_i *do*

5 $accuracy_i$*= GSOT (*$classifier_i$*,* h_i *,* D_{Train} *, CV=m)*

6 if $accuracy_i$ *is the maximum so far then*

7 $besthyper_i = h_i$

8 end if

9 end for

10 end for

// Applying Soft-based Heterogeneous Ensemble Detection Model

11 for i= 1 to C do *// C=3, No. of Classifiers*

12 Train $classifier_i$ *(ORF,KNN, and OLightGBM) on* D_{Train} *using the best hyper-parameters (*$besthyper_i$*)*

13 $Probability^{(i)}$*=null*

13 end for

for j= 1 to D_{Train} do
15 for i= 1 to C do
16 $Probability_j^{(i)} = Classifier_i(D_j)$
17 end for
18 $Probability_j = \frac{1}{C} \sum_{i=1}^{c} \boldsymbol{Probability_j^{(i)}}$
19 end for
20 Compute evaluation metrics for DSBHED such as accuracy , detection rate, precision, FPR, F1-score, testing time on D_{Test}
End.

3.11 Evaluation of the Proposed Models

The evaluation phase is the most vital part of the proposed models to decide the efficiency and quality of our proposed solutions in detecting attacks. According to the objectives of the proposed models for binary and multi-class classification, for these tasks, the following evaluation metrics were considered: Accuracy, Precision, F1 score, detection rate, and FPR, in addition to the training and testing times were also considers. The techniques like m-fold cross-validation and holdout train test splitting were employed to compute these evaluation metrics. In the proposed models, the datasets were separated as 70:30 to train and test the models.

Chapter Four

RESULTS AND DISCUSSION

4.1 Introduction

This chapter presents and discusses the experimental outcomes of the proposed model stages, which were discussed in the previous chapter. Experiments have been performed on four datasets available for research community: the InSDN, SNMP, CICDDoS-2019, and CICIDS-2017 datasets. Afterwards, the outcomes are arranged depending on their appearance in chapter three. However, this chapter stares with the hardware and software specifications to implement the proposed models.

4.2 Hardware and Software Specifications

Hardware: Processor Intel C-i5, RAM 8GB, Storage 500 GB.

Operating System: Windows8 (64) bit.

Programming Language: Python programming language version 3.8.

4.3 Description of Datasets

The quality of the training network datasets plays a key role in constructing an efficient ML and DNN based IDS models. However, the availability of high-quality datasets for network traffic and intrusion detection, in general, is a significant issue. In several application domains, such as computer vision and language translation, a lot of different datasets with high quality are online available for research community. In contrast, the network data traffic can contain sensitive client information, and it is against privacy or illegal to publish such information to the public.

Thus, this dissertation utilizes four datasets available to the research community. One of them the newly released InSDN dataset [20]. The InSDN dataset contains most recent popular attack types like DDoS, Probe, DoS, Botnet, Password-Guessing, Web attack, and Exploitation.

Besides, numerous attack scenarios are considered from several sources coming from both inside and outside the SDN network. Moreover, the normal network traffic in the dataset includes common applications like HTTP, HTTPS, Email, DNS, SSH, and FTP. The InSDN dataset contains 361,317 observations for attacks and normal traffic, wherein 292,893 for attack observations and 68,424 for normal observations. Table 4.1 shows how these data observations are distributed. Figure 4.2 illustrates the sample of dataset.

Table (4.1): The Distribution of Observations in an InSDN Dataset

Labels	Instances
legitimate	68,424
Exploitation (U2R)	17
DoS	69,044
Probe	98,129
Password-Guessing	1405
BOTNET	164
Web-Attack	192
DDoS	123,942
Total	361,317

Flow ID	Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	...	Fwd Seg Size Min	Ac
185.127.17.56-192.168.20.133-443-53648-6	185.127.17.56	443	192.168.20.133	53648	6	5/2/2020 13:58	245230	44	40	...	0	
185.127.17.56-192.168.20.133-443-53650-6	192.168.20.133	53650	185.127.17.56	443	6	5/2/2020 13:58	1605449	107	149	...	0	
192.168.20.133-192.168.20.2-35108-53-6	192.168.20.133	35108	192.168.20.2	53	6	5/2/2020 13:58	53078	5	5	...	0	
192.168.20.133-192.168.20.2-35108-53-6	192.168.20.2	53	192.168.20.133	35108	6	5/2/2020 13:58	6975	1	1	...	0	
154.59.122.74-192.168.20.133-443-60900-6	192.168.20.133	60900	154.59.122.74	443	6	5/2/2020 13:58	190141	13	16	...	0	

Figure (4.1): A Sample of InSDN Dataset.

The realistic SNMP dataset was used, which it constructed by [90]. This dataset includes a Brute Force attack and six types of DoS/DDoS attacks as showed in Table 4.2. In addition, the dataset

contains 4998 observations, where each instance record contains 34 features. Figure 4.2 demonstrates the sample of SNMP dataset.

Table (4.2): The Distribution of Instances in a SNMP Dataset.

No.	Class name	No. of observations
1.	Normal	600
2.	TCP - SYN	960
3.	UDP-flood	773
4.	ICMP - ECHO	632
5.	HTTP-flood	573
6.	Slowloris-attack	780
7.	Slowpost-attack	480
8.	Brute-Force-attack	200

ifInUcastPkts11	ifInNUcastPkts11	ifInDiscards11	ifOutUcastPkts11	ifOutNUcastPkts11	tcpOutRsts	tcpInSegs
52007310	16978	0	7197292	3968	1	682
52098054	16986	0	7227073	3968	1	682
52185853	16994	0	7255792	3969	1	682
52287097	17015	0	7291152	3975	1	701
52347521	17043	0	7313830	3977	1	709

Figure (4.2): A sample of SNMP dataset.

The CICDDoS-2019 dataset [21], released by the Canadian Cybersecurity Foundation in collaboration with the University of New Brunswick. The research community has been using high quality datasets obtained by these foundations for many years. The dataset has been constructed according to the limitations of previous datasets. It contains normal and the most up-to-date common DDoS attacks, categorized as exploitation-based and reflection-based DDoS attacks. The number of normal samples is 18906, while the number of different DDoS attacks is 354755. Figure (4.3) displays a sample of the CICIDS-2019 dataset.

Owing to the exploitation-based DDoS attacks, the intruders exploit the specific feature or weakness of a protocol or application to extremely exhaust and consume the target host's resource and bring it

shutdown. The UDP-Flood attack and SYN-Flood are examples of exploitation attack groups.

Due to the reflection-based DDoS attacks, the intruders hide their identity and exhaust the device resources utilizing a genuine third party node as the reflector server. The intruders initially send many packets to the reflector server, using the device's spoofed IP-address as the source address. According to the use of many reflector servers, the victim's capability to process the replay of these packets are reduced. The SNMP, NETBIOS, and LDAP are examples of reflection attack groups. For more details, Table 4.4 demonstrates the sample distribution of the divers attacks utilized in this dissertation.

Table (4.3) The Sample Distribution of CICDDoS-2019 Dataset.

Class name	Instances
legitimate	18906
UDP	60746
Syn	59522
Portmap	58171
SNMP	65093
MSSQL	56062
NTP	55161
Total	373661

Source IP	Source Port	Destination IP	Destination Port	Protocol	Timestamp	Flow Duration	...
192.168.50.253	0.0	224.0.0.5	0.0	0.0	22:42.0	5.0	...
8.6.0.1	0.0	8.0.6.4	0.0	0.0	22:43.9	2.0	...
192.168.50.254	0.0	224.0.0.5	0.0	0.0	22:48.5	3.0	...
192.168.50.8	59307.0	125.56.201.105	80.0	6.0	22:49.6	173888.0	...
192.168.50.6	57215.0	23.194.142.213	443.0	6.0	22:51.0	20586.0	...

Figure (4.3): A Sample of CICDDoS-2019 Dataset.

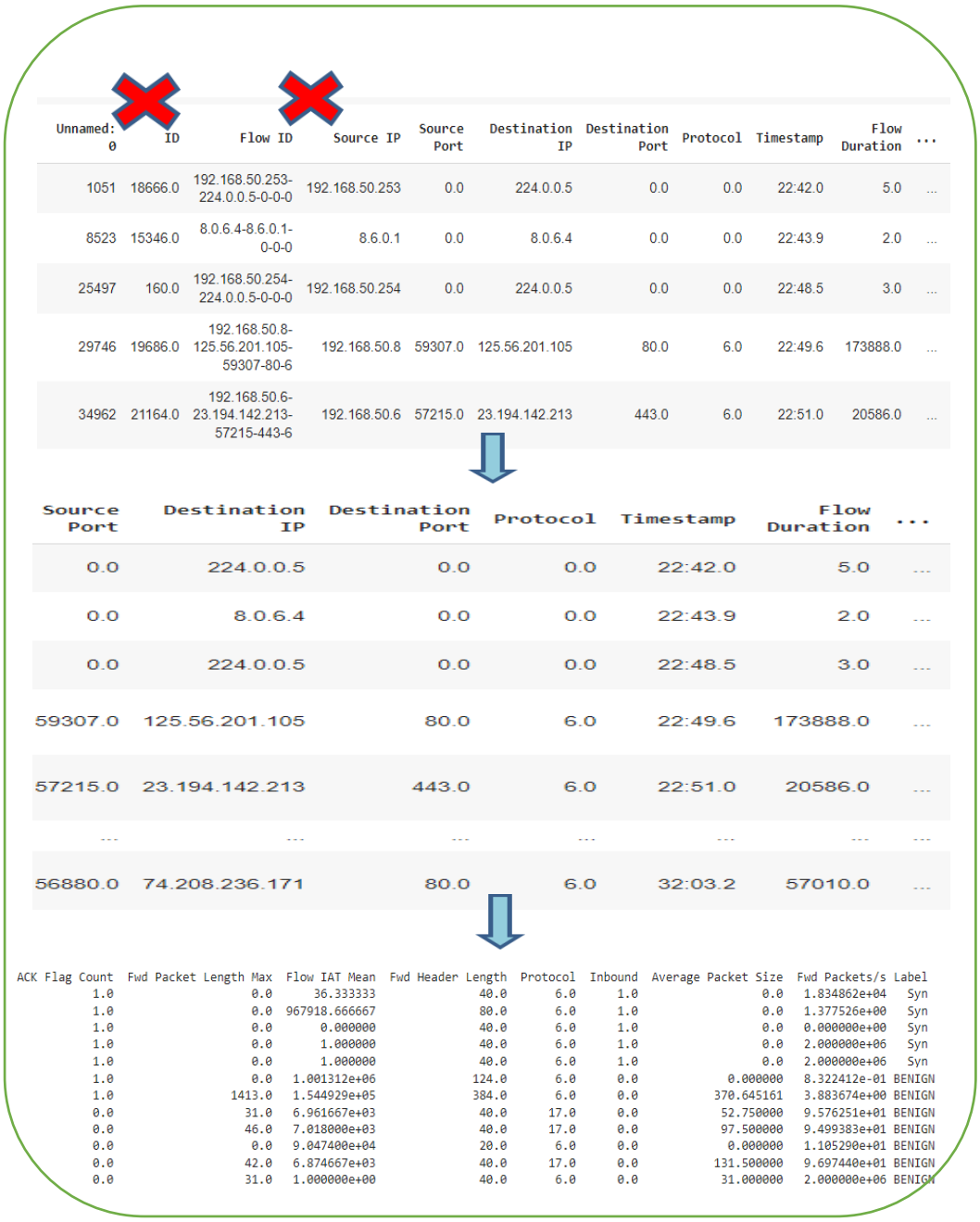
The CICIDS-2017 dataset was published via Canadian cyber security Institute [18]. It includes normal and most up to date attacks like DoS/DDoS that were derived depend on host characteristics of different protocols such as FTP, HTTP/S, SSH, etc. In this dataset, seventy-eight features were found, and for the current dissertation, the number of samples is 48180 samples out of which 9772 belong to normal class and 38408 samples belong to several DoS/DDoS attacks categories. Figure (4.4) shows a sample of the CICIDS-2017 dataset.

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	...
80	1146579	3	5	26	11601	20	0	8.666667	10.263203	...
80	892791	3	5	26	11607	20	0	8.666667	10.263203	...
80	81155768	8	4	56	11601	20	0	7.000000	5.656854	...
80	5781854	5	0	30	0	6	6	6.000000	0.000000	...
80	1898989	3	6	26	11607	20	0	8.666667	10.263203	...

Figure (4.4) A Sample of CICIDS-2017 Dataset.

4.4 Results of Data Pre-processing

The aim of this stage is to prepare the data to be suitable for the training ML models directly. The Pre-processing steps which are stated in Section (3.3) are demonstrated in Figure (4.5). This Figure illustrated, the overall framework for the data pre-processing stage, where further steps rely on the dataset's nature. For instance, remove of duplicate features is performed on the CICIDS-2017 dataset and not performed on other datasets.



Unnamed: 0	ID	Flow ID	Source IP	Source Port	Destination IP	Destination Port	Protocol	Timestamp	Flow Duration	...
1051	18666.0	192.168.50.253-224.0.0.5-0-0-0	192.168.50.253	0.0	224.0.0.5	0.0	0.0	22:42.0	5.0	...
8523	15346.0	8.0.6.4-8.6.0.1-0-0-0	8.6.0.1	0.0	8.0.6.4	0.0	0.0	22:43.9	2.0	...
25497	160.0	192.168.50.254-224.0.0.5-0-0-0	192.168.50.254	0.0	224.0.0.5	0.0	0.0	22:48.5	3.0	...
29746	19686.0	192.168.50.8-125.56.201.105-59307-80-6	192.168.50.8	59307.0	125.56.201.105	80.0	6.0	22:49.6	173888.0	...
34962	21164.0	192.168.50.6-23.194.142.213-57215-443-6	192.168.50.6	57215.0	23.194.142.213	443.0	6.0	22:51.0	20586.0	...

Source Port	Destination IP	Destination Port	Protocol	Timestamp	Flow Duration	...
0.0	224.0.0.5	0.0	0.0	22:42.0	5.0	...
0.0	8.0.6.4	0.0	0.0	22:43.9	2.0	...
0.0	224.0.0.5	0.0	0.0	22:48.5	3.0	...
59307.0	125.56.201.105	80.0	6.0	22:49.6	173888.0	...
57215.0	23.194.142.213	443.0	6.0	22:51.0	20586.0	...
...	...	...	...	...	...	...
56880.0	74.208.236.171	80.0	6.0	32:03.2	57010.0	...

ACK Flag Count	Fwd Packet Length Max	Flow IAT Mean	Fwd Header Length	Protocol	Inbound	Average Packet Size	Fwd Packets/s	Label
1.0	0.0	36.333333	40.0	6.0	1.0	0.0	1.834862e+04	Syn
1.0	0.0	967918.666667	80.0	6.0	1.0	0.0	1.377526e+00	Syn
1.0	0.0	0.000000	40.0	6.0	1.0	0.0	0.000000e+00	Syn
1.0	0.0	1.000000	40.0	6.0	1.0	0.0	2.000000e+06	Syn
1.0	0.0	1.000000	40.0	6.0	1.0	0.0	2.000000e+06	Syn
1.0	0.0	1.001312e+06	124.0	6.0	0.0	0.000000	8.322412e-01	BENIGN
1.0	1413.0	1.544929e+05	384.0	6.0	0.0	370.645161	3.883674e+00	BENIGN
0.0	31.0	6.961667e+03	40.0	17.0	0.0	52.750000	9.576251e+01	BENIGN
0.0	46.0	7.018000e+03	40.0	17.0	0.0	97.500000	9.499383e+01	BENIGN
0.0	0.0	9.047400e+04	20.0	6.0	0.0	0.000000	1.105290e+01	BENIGN
0.0	42.0	6.874667e+03	40.0	17.0	0.0	131.500000	9.697440e+01	BENIGN
0.0	31.0	1.000000e+00	40.0	6.0	0.0	31.000000	2.000000e+06	BENIGN

Figure (4.5): (a) Data Preprocessing Steps.

ACK Flag Count	Fwd Packet Length Max	Flow IAT Mean	Fwd Header Length	Protocol	Inbound	Average Packet Size	Fwd Packets/s	Label
1.0	0.0	36.333333	40.0	6.0	1.0	0.0	1.834862e+04	0
1.0	0.0	967918.666667	80.0	6.0	1.0	0.0	1.377526e+00	0
1.0	0.0	0.000000	40.0	6.0	1.0	0.0	0.000000e+00	0
1.0	0.0	1.000000	40.0	6.0	1.0	0.0	2.000000e+06	0
1.0	0.0	1.000000	40.0	6.0	1.0	0.0	2.000000e+06	0
1.0	0.0	1.001312e+06	124.0	6.0	0.0	0.000000	8.322412e-01	1
1.0	1413.0	1.544929e+05	384.0	6.0	0.0	370.645161	3.883674e+00	1
0.0	31.0	6.961667e+03	40.0	17.0	0.0	52.750000	9.576251e+01	1
0.0	46.0	7.018000e+03	40.0	17.0	0.0	97.500000	9.499383e+01	1
0.0	0.0	9.047400e+04	20.0	6.0	0.0	0.000000	1.105290e+01	1
0.0	42.0	6.874667e+03	40.0	17.0	0.0	131.500000	9.697440e+01	1
0.0	31.0	1.000000e+00	40.0	6.0	0.0	31.000000	2.000000e+06	1

Destination Port	Total Length of Fwd Packets	Fwd Packet Length Max	Fwd Packet Length Mean	Flow Bytes/s
4435.0	0	0.0	0.0	0.0
22966.0	458	229.0	229.0	9541666.667
53835.0	1118	559.0	559.0	559000000.0
26122.0	458	229.0	229.0	458000000.0
28679.0	1298	649.0	649.0	1298000000.0
0.0	78	0.0	0.0	7856.247388
38137.0	43120	440.0	440.0	2807291.667
0.0	458	0.0	0.0	0.0
53143.0	0	0.0	0.0	0.0
42617.0	1438	389.0	359.5	13602.73947
55914.0	458	229.0	229.0	229000000.0
0.0	494	0.0	0.0	494000000.0
35569.0	766	383.0	383.0	766000000.0
43452.0	2464	1232.0	1232.0	2464000000.0
1607.0	26256	440.0	437.6	5868573.983
59220.0	0	0.0	0.0	0.0

Destination Port	Total Length of Fwd Packets	Fwd Packet Length Max	Fwd Packet Length Mean	Flow Bytes/s
0.06767376211184863	0.0	0.0	0.0	0.0
0.35043869687953	0.002602727272727	0.0641636312692631	0.12200319659030368	0.003241055253736413
0.8214694438086518	0.006352727272727	0.1566265060240964	0.297815663292488	0.18987771739130435
0.39859616998550396	0.002602727272727	0.0641636312692631	0.12200319659030368	0.15557065217391305
0.4376134889753567	0.007375	0.18184365368450547	0.3457645178476292	0.4408967391304348
0.0	0.00044318181818181815	0.0	0.0	2.6685622921195652e-06
0.5819333180743115	0.245	0.1232838330064444	0.2344166222695791	0.0009535637455842391
0.0	0.002602727272727	0.0	0.0	0.0
0.810910200656138	0.0	0.0	0.0	0.0
0.6502937361715114	0.008170454545454545	0.10899411599887925	0.1915290356952584	4.62049574388587e-06
0.8531929503318837	0.002602727272727	0.0641636312692631	0.12200319659030368	0.07778532608695653
0.0	0.0028068181818181816	0.0	0.0	0.16779891304347827
0.542748149843595	0.004352727272727	0.10731297282151864	0.20404901438465636	0.2601902173913044
0.6630350194552529	0.013999999999999999	0.34519473241804427	0.6563665423548215	0.8369565217391305

Figure (4.5): (b) Data Preprocessing Steps.

In the figure (4.5) above, the steps of data pre-processing on datasets used in this dissertation are shown, as follows:

The pre-processing step begins by eliminating inappropriate features. Features like "Unnamed: 0", "Flow .ID" and “ID” do not hold

any important information to the issue at hand. In addition, infinity values, Not a Number (NaN) values, and duplicate features like "Fwd-Header-Length" in CICIDS-2017 dataset were excluded from the dataset.

Afterwards, the label conversion step has been done to deal with non-numerical values where the utilized datasets include binary, continuous, and symbolic values. For example, the attribute 'protocol' in the InSDN dataset comprises symbolic value like “udp”, “icmp”, and “tcp”. A label encoder method has been performed to solve this issue. Moreover, since this dissertation considers both multi-class and binary prediction of DoS/DDoS attacks. This required to map the dataset's label from multi label into binary label (i.e., 0: DoS/DDoS attack and 1: Normal).

Different ranges between feature values can degrade the performance of the proposed models. Thus, a normalization technique, the min-max method, has been utilized after feature reduction stage to map the range of feature values into (0 to 1).

4.5 Results of Hybrid Feature Reduction

Not all features contribute equally to the detection of attacks. Some features may have a more significant impact on the performance of the detection model, while others may have little or no effect. Furthermore, certain features may even degrade the model's performance and increase processing time. Thus, feature reduction is an important step in ML techniques because of its ability to eliminate irrelevant features, reduce the dimensionality of the dataset, decrease computational complexity, and improve overall performance detection.

Therefore, we propose an HFR approach that leverages multiple feature reduction approaches to obtain the fewest possible features. This

significantly reduced feature set aims to effectively detect both DoS/DDoS attacks and multi-class attack.

The HFR consists of three steps. In the first step, three different approaches: Filter, embedded, and wrapper, were applied separately to the InSDN, SNMP, and CICDDoS-2019 datasets. In the filter technique, the MI method has been performed to assess features according to their importance to the target label. Based on this approach, the number of features was reduced for the aforementioned datasets for both the binary-based and multiclass detection tasks.

In the wrapper-based approach, which evaluates feature subsets, relies on the performance of the prediction model. Thus, in this dissertation, the WOA was combined with the KNN classifier (k=7) as a fitness function to find a feature subset. Through this approach, the number of features was also decreased for the InSDN, SNMP, and CICDDoS-2019 datasets.

In the embedded-based approach, the features were evaluated implicitly through an extra tree classifier, where the evaluation depends on the decrease in impurity when selecting a feature as the splitting node. The contribution of each feature in reducing impurity was measured, and the average contribution across all trees is used as a weight or importance for that feature. Due this way, the number of features was diminished for utilized datasets.

In the second step, the majority-voting technique has been utilized. Throughout this step, a feature has been selected if and only if it is selected by any two out of the three approaches. Due to this way, the number of features was reduced for the binary-based and multiclass detection on all datasets.

In the third step, according to the three above approaches which gives a diverse weights to the same feature, that is, every feature has three weights. Thus, to have only one weight for each feature, the mean value has been computed. Afterwards, the obtained values are arranged from high to low, and each feature's weight less than an experimental threshold was removed. After this process, only 14 (InSDN), 9 (SNMP), and 12 (CICDDoS-2019) features remained which denotes the best subset of features for binary-based models detection attacks. While, for multiclass-based model detection attacks, only 14 (InSDN), 10 (SNMP), and 15 (CICDDoS-2019) features which denotes the best subset of features. The outcomes of these feature reduction steps for the InSDN, SNMP, and CICDDoS-2019 datasets are presented in Tables (4.4) and (4.8). These tables provide a detailed information of the best subset of features and their corresponding weights.

Table (4.4): Feature Reduction Results for Binary-based Models Detection

Dataset	Feature reduction method	Threshold value	Selected features
InSDN	MI	0.01	65 out of 83
	ETC	0	57 out of 83
	WOA	0	33 out of 83
	Majority-voting + Mean	max-vote + 0.3	14
SNMP	MI	0.07	18 out of 34
	ETC	0.01	25 out of 34
	WOA	0	16 out of 34
	Majority-voting + Mean	max-vote + 0.36	9
CICDDoS-2019	MI	0.02	54 out of 89
	ETC	0	66 out of 89

	WOA	0	22 out of 89
	Majority voting + Mean	max vote + 0.35	12

Table (4.5): The Best Subset of Features for Binary-based Detection InSDN dataset

Feature Name	MI	ETC	WOA	Final Weight
Init Bwd Win Byts	0.2141	0.2600	1	0.4914
Bwd Header Len	0.4234	0.0123	1	0.4786
Fwd Header Len	0.4229	0.0037	1	0.4755
TotLen Bwd Pkts	0.3417	0.006	1	0.4492
Bwd Pkt Len Max	0.3392	0.0061	1	0.4484
Fwd Pkt Len Mean	0.3282	0.0092	1	0.4458
Fwd Seg Size Avg	0.3281	0.0071	1	0.4451
Subflow Fwd Pkts	0.3111	0.0077	1	0.4396
TotLen Fwd Pkts	0.2911	0.0037	1	0.4316
Tot Bwd Pkts	0.2748	0.0096	1	0.4281
Down/Up Ratio	0.1793	0.0538	1	0.4110
Pkt Len Min	0.1831	0.0299	1	0.4043
Bwd IAT Mean	0.2117	0.0009	1	0.4042
Flow IAT Max	0.2004	0.0034	1	0.4013

Table (4.6): The Best Subset of Features for Binary-based Detection SNMP dataset

Feature Name	MI	ETC	WOA	Final Weight
icmpOutMsgs	0.1617	0.0583	1	0.4067
icmpInDestUnreachs	0.1465	0.0633	1	0.4033
tcpInSegs	0.1644	0.0451	1	0.4032
tcpRetransSegs	0.1056	0.0647	1	0.3901
ipInReceives	0.1168	0.0406	1	0.3858
icmpOutDestUnreachs	0.1047	0.044	1	0.3829
ifOutNUcastPkts11	0.0982	0.0446	1	0.3809
ipForwDatagrams	0.1106	0.0308	1	0.3805
ipOutNoRoutes	0.0951	0.0447	1	0.3799

Table (4.7): The Best Subset of Features for Binary-based Detection

CICDDoS-2019 dataset

Feature Name	MI	ETC	WOA	Final Weight
Bwd Packet Length Min	0.0631	0.097	1	0.3867
Max Packet Length	0.1258	0.0145	1	0.3801
Bwd Packet Length Max	0.0971	0.042	1	0.3797
Fwd Packet Length Mean	0.1224	0.0166	1	0.3797
URG Flag Count	0.0399	0.0959	1	0.3786
Packet Length Mean	0.1228	0.0116	1	0.3781
Average Packet Size	0.1218	0.0122	1	0.3780
Flow IAT Mean	0.1259	0.0044	1	0.3768
Flow IAT Std	0.0033	0.1154	1	0.3729
Packet Length Variance	0.1034	0.0071	1	0.3702
Fwd Packet Length Min	0.0964	0.0123	1	0.3696
Bwd IAT Min	0.0799	0.0009	1	0.3603

Table (4.8): Feature Reduction Results for Multiclass-based Model Detection

Dataset	Feature reduction method	Threshold value	Selected features
InSDN	MI	0.1	58 out of 83
	ETC	0	65 out of 83
	WOA	0	39 out of 83
	Majority-voting + Mean	max-vote + 0.52	14
SNMP	MI	0.1	15 out of 34
	ETC	0.01	25 out of 34
	WOA	0	18 out of 34
	Majority-voting + Mean	max-vote + 0.32	10
CICDDoS-2019	MI	0.1	51 out of 89
	ETC	0.01	53 out of 89
	WOA	0	44 out of 89

	Majority voting + Mean	max vote + 0.51	15

Table (4.9): The Best Subset of Features for Multi-based Detection InSDN Dataset

Feature Name	MI	ETC	WOA	Final Weight
Bwd Header Len	1.25054	0.0274	1	0.7593
Protocol	0.7788	0.1939	1	0.6576
Fwd Header Len	0.8501	0.0105	1	0.6202
Bwd IAT Tot	0.8116	0.0061	1	0.6059
Bwd Pkts/s	0.7721	0.0453	1	0.6058
Bwd IAT Mean	0.8102	0.0034	1	0.6045
Bwd IAT Max	0.8062	0.0057	1	0.6040
Flow Pkts/s	0.7631	0.0353	1	0.5995
Flow IAT Mean	0.7646	0.0024	1	0.5890
Tot Fwd Pkts	0.7235	0.0086	1	0.5774
Flow Duration	0.7203	0.0092	1	0.5765
Tot Bwd Pkts	0.7077	0.0125	1	0.5734
Pkt Len Max	0.6923	0.0154	1	0.5692
Pkt Len Var	0.6863	0.0045	1	0.5636

Table (4.10): The Best Subset of Features for Multi-based Detection SNMP Dataset

Feature Name	MI	ETC	WOA	Final Weight
tcpInSegs	0.1786	0.0451	1	0.4079
icmpInMsgs	0.1771	0.0379	1	0.4050
tcpPassiveOpens	0.1631	0.0384	1	0.4005
icmpInEchos	0.1495	0.0506	1	0.4000
ipOutDiscards	0.1492	0.0496	1	0.3996
tcpOutSegs	0.1637	0.0326	1	0.3988
ipInAddrErrors	0.115	0.0724	1	0.3958
tcpRetransSegs	0.1039	0.0647	1	0.3895
icmpOutDestUnreachs	0.0991	0.0441	1	0.3811
ipForwDatagrams	0.1084	0.0308	1	0.3797

Table (4.11): The Best Subset of Features for Multi-based Detection CICDDoS-2019 dataset

Feature Name	MI	ETC	WOA	Final Weight
Average Packet Size	1.5133	0.0756	1	0.8630
Max Packet Length	1.5164	0.0509	1	0.8558
Packet Length Mean	1.5118	0.0553	1	0.8557
Avg Fwd Segment Size	1.5023	0.0648	1	0.8557
Fwd Packet Length Mean	1.5016	0.0558	1	0.8525
Fwd Packet Length Max	1.5002	0.0445	1	0.8482
Total Length of Fwd Packets	1.5091	0.0247	1	0.8446
Fwd Packet Length Min	1.4763	0.0512	1	0.8425
Min Packet Length	1.4746	0.0504	1	0.8417
Flow Bytes/s	1.4144	0.0364	1	0.8169
Flow Packets/s	1.0661	0.0277	1	0.6979
Flow IAT Mean	1.0581	0.0042	1	0.6874
Flow Duration	1.0382	0.0093	1	0.6825
Protocol	0.8079	0.0874	1	0.6318
Bwd Packets/s	0.5143	0.0912	1	0.5352

4.6 Results of Optimized CatBoost Model

The OCBM is proposed to enhance efficiency and effectiveness binary prediction of DoS/DDoS attacks with a low FPR and therefore improve the security of SDN against such attacks. An HFR approach was utilized on the original datasets to identify the best subset of features. Once the best subset of features was determined as shown in section (4.5), the GSCV optimization technique was adopted to find the best hyperparameter values for the OCBM. The best values have been obtained through a GSCV technique were 300, 0.1, and 5 for the boosted of trees, learning rate, and max depth, respectively. This optimization process helps to reduce the complexity of the OCBM and made it more suitable for real-time detection, increasing its efficiency, and effectiveness in detecting DoS/DDoS attacks.

4.6.1 Experiments on InSDN Dataset using OCBM

To construct an intrusion detection model with high accuracy, high detection rate, low false alarm rate, and better reliability, as well as to have shorter testing time on InSDN dataset, the OCBM was optimized through the GSCV method. To validate the efficacy of the proposed model using the best subset of features for the InSDN dataset, as demonstrated in Table (4.6), we compared the results of our proposed model against the baseline model, other ML classifiers and the outcomes of previous studies conducting on the same dataset. In this dissertation, five major evaluation metrics such as Accuracy, Precision, Detection rate, F1 Score, and FPR were utilized to validate the proposed model, in addition to the training (training) time and testing time for each classifier were also consider. Tables (4.12) and (4.13) demonstrated the evaluation results. Figure (4.7) depicts the performance of OCBM on InSDN dataset.

Table (4.12): Comparison performance of the proposed model (OCBM with HFR) with baseline and other ML classifiers on InSDN dataset

Classifier	Accuracy	Precision	Detection rate	F1-Score	FPR
CatBoost(Baseline)	0.9995	0.9995	0.9996	0.9995	0.0007
AdaBoost	0.9994	0.9994	0.9994	0.9994	0.0014
KNN	0.9993	0.9993	0.9993	0.9993	0.0015
GBM	0.9991	0.9991	0.9992	0.9991	0.0021
SVM	0.8757	0.8685	0.8757	0.8688	0.2408
NB	0.8073	0.7824	0.8073	0.7883	0.3754
LR	0.8339	0.8171	0.8339	0.8049	0.3707
OCBM	**0.9997**	**0.9996**	**0.9997**	**0.9997**	**0.0004**
The study in [20]	0.875	0.89	0.93	0.91	-
The study in [25]	0.997	0.998	0.998	0.998	-

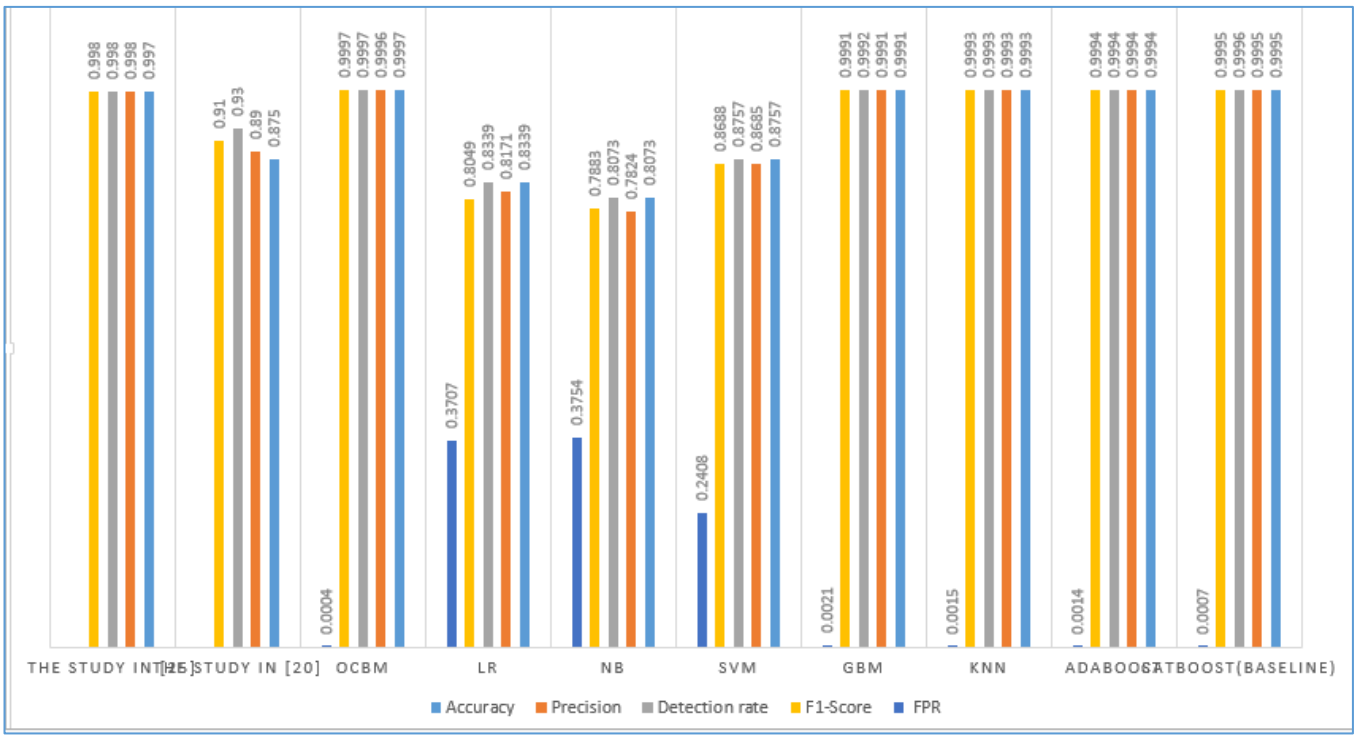


Figure (4.6): Performance of OCBM on InSDN Dataset

Table (4.12) and Figure (4.6) demonstrate that the OCBM with the best subset of features (14 f.) outperforms the baseline classifier and other ML classifiers such as AdaBoost, KNN, GBM, NB, LR, and SVM in terms of all evaluation measures. Although these classifiers have achieved good results, they utilized default hyperparameters in training their models. For instance, the baseline CatBoost model used 600 (boosted tree) and 7 (max-depth) in this experiment. In contrast, the OCBM achieved the highest accuracy (0.9997), precision (0.9996), detection rate (0.9997), F1 score (0.9997), and the lowest FPR (0.0004). Furthermore, the OCBM utilized fewer boosted trees (300) and a maximum depth of five. The OCBM also achieved promising results in comparison to other studies.

Table (4.13): Summary of training time and test time of the proposed model (OCBM) with baseline and other classifiers on InSDN dataset

Classifier	Training Time (sec.)	Testing Time (sec.)
CatBoost(Baseline)	60.890	0.126
AdaBoost	9.724	0.621
KNN	0.921	142.240
GBM	39.223	0.150
SVM	7.308	0.011
NB	0.069	0.036
LR	2.519	0.029
OCBM	**15.504**	**0.051**
The study in [20]	479.748	38.355

Table (4.13) illustrates the training time and testing time for the baseline model, proposed model, other ML-based classifiers, and pervious study on the InSDN dataset. According to the GSCV optimization technique, which specifies the best values for hyperparameters of the OCBM classifier and using the HFR approach to determine the best subset of features. Utilized these approaches resulted in improved the performance of proposed model (see Table 4.12) as well as sharply decreased the training and testing times overhead (see Table 4.13). The results have also shown that the proposed model, OCBM, is faster than the baseline model and previous study. While some ML like SVM, LR, and NB may be faster in terms of time complexity but the proposed model (OCBM) accomplished a good balance between prediction power and computational efficiency, making it more suitable for online detection processes.

4.6.2 Experiments on SNMP Dataset using OCBM

The proposed model, OCBM, was also verified on the SNMP dataset. First, the nine best features were obtained by the HFR approach from SNMP dataset, as shown in Table (4.6). Then, the proposed model was applied on these features. Although, the OCBM with baseline

classifiers have achieved the highest performance as compared to other ML classifiers and previous studies in terms of all evaluation measures, as demonstrated in Table (4.14) and Figure (4.7). However, the OCBM has a lowest detection (testing) time that represents a critical factor in terms of IDSs, as depicted in Table (4.15). Furthermore, the previous studies did not consider the computation time in their evaluation models. As a result, thanks to the GSCV technique that improves the hyper parameters of the proposed model without any effect on its performance.

Table (4.14): Comparison performance of the proposed with baseline model and other ML-based classifiers on SNMP dataset

Classifier	Accuracy	Precision	Detection rate	F1-Score	FPR
CatBoost(Baseline)	0.9999	0.9999	0.9999	0.9999	0.01
GBM	0.9996	0.9996	0.9995	0.9995	0.02
SVM	0.9913	0.9914	0.9913	0.9912	0.03
NB	0.9913	0.9882	0.9734	0.9807	0.02
LR	0.9193	0.9261	0.9193	0.9032	0.30
OCBM	**0.9999**	**0.9999**	**0.9999**	**0.9999**	**0.01**
The study in [91]-Random Forest	0.9998	-	-	-	-
The study in [91]-J48	0.9988	-	-	-	-
The study in [92]	0.993	-	-	-	0.4

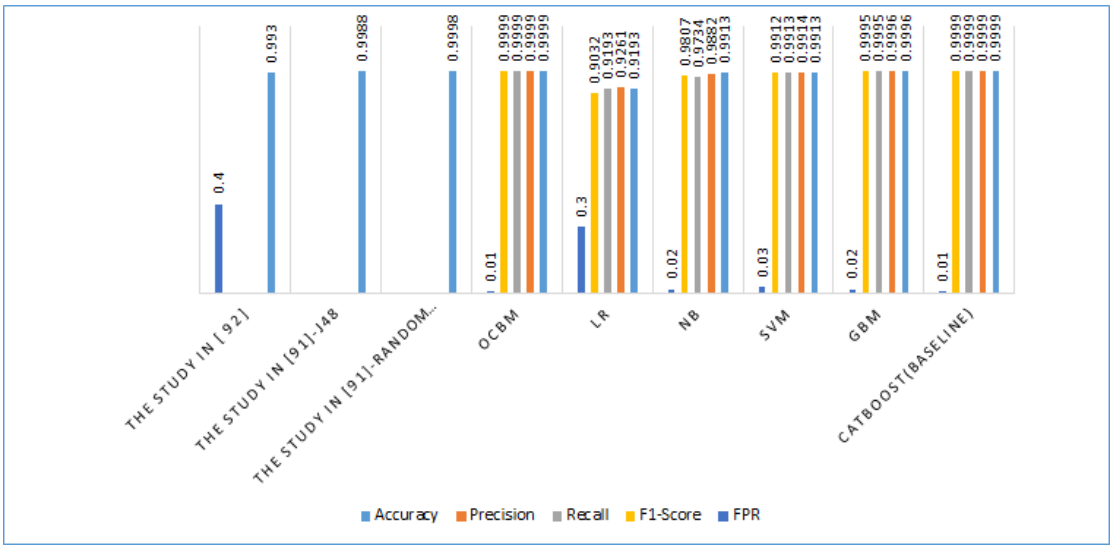


Figure (4.7) Performance of OCBM on SNMP Dataset

Table (4.15): Summary of training time and test time of the proposed model with other ML-based classifiers on SNMP dataset

Classifier	Training Time (sec.)	Testing Time (sec.)
CatBoost(Baseline)	3.994	0.015
GBM	0.850	0.064
SVM	0.045	0.047
NB	0.014	0.005
LR	0.047	0.003
OCBM	**0.450**	**0.002**

4.6.3 Experiments on CICDDoS-2019 Dataset using OCBM

The OCBM tested on CICDDoS-2019 dataset for binary prediction DoS/DDoS attack. The proposed model was developed in order to further boost the intrusion detection mechanism against such attacks. Initially, data preprocessing has been performed to enhance the generalizability of the classifier. Then, HFR approach was applied to elect the most relevant features (12 *features*), as described in Table (4.7), which aids in reducing the computational time and avoid overfitting, as well as improving overall prediction results. Next, the complexity of the OCBM has been enhanced by utilizing a GSCV technique. Lastly, both the best subset of features and hyper-parameters are fed to the OCBM. Moreover, to prove the efficiency and effectiveness of our proposed model, it was compared

with the baseline model, state-of-the-art ML-based classifiers, and recent studies that were carried out on the CICDDoS-2019 dataset. The final experimental outcomes are shown in Table (4.16) and Table (4.17). Figure (4.8) depicts the performance of OCBM on CICDDoS-2019 dataset.

Table (4.16): Comparison performance of the proposed model (OCBM) with other classifiers on CICDDoS-2019 dataset

Classifier	Accuracy	Precision	Detection rate	F1-Score	FPR
CatBoost(Baseline)	0.9998	0.9998	0.9998	0.9998	0.0009
AdaBoost	0.9994	0.9994	0.9994	0.9994	0.0016
KNN	0.9993	0.9993	0.9993	0.9993	0.0042
GBM	0.9995	0.9995	0.9995	0.9995	0.0016
LightGBM	0.9996	0.9996	0.9996	0.9996	0.0015
SVM	0.9527	0.9424	0.9527	0.9361	0.444
NB	0.9097	0.9614	0.9099	0.9277	0.0947
LR	0.9520	0.9404	0.9520	0.9347	0.4511
OCBM	**0.9998**	**0.9998**	**0.9998**	**0.9998**	**0.0009**
The study in [93]-NB	0.57	-	-	-	-
The study in [93]-RF	0.86	-	-	-	-
The study in [93]-DT	0.77	-	-	-	-
The study in [93]-LR	0.95	-	-	-	-
The study in [94]	0.9998	0.9851	0.9835	0.9842	-

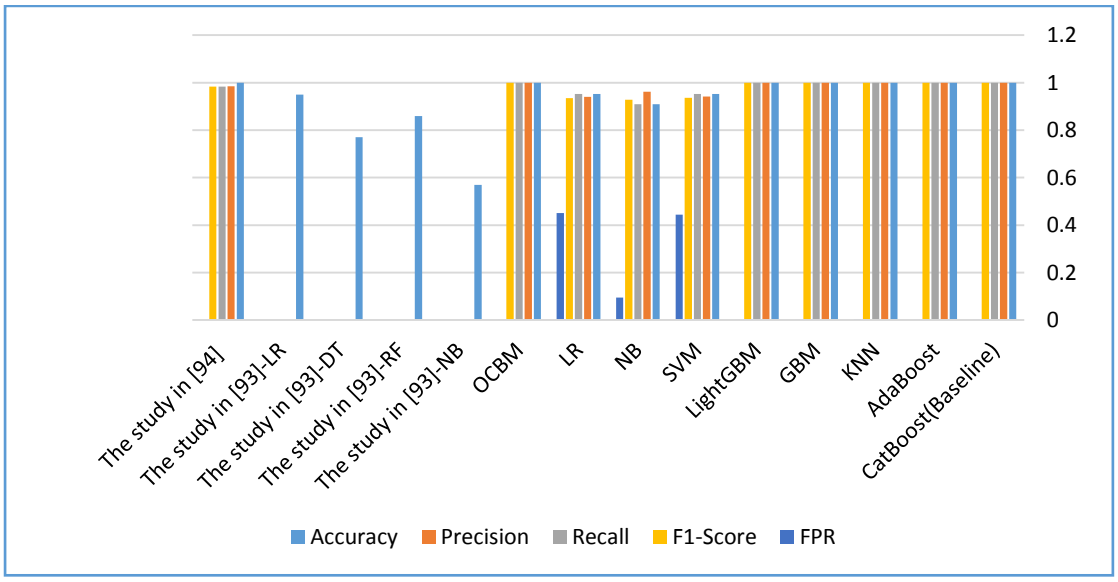


Figure (4.8) Performance of OCBM on CICDDoS-2019 Dataset.

Table (4.17): Summary of training time and test time of the proposed model (OCBM with HFR.) with other classifiers on CICDDoS-2019 dataset

Classifier	Training Time (sec.)	Testing Time (sec.)
CatBoost(Baseline)	64.868	0.147
AdaBoost	13.818	0.656
KNN	0.938	38.574
GBM	58.276	0.150
LightGBM	2.527	0.322
SVM	2.415	0.024
NB	0.069	0.059
LR	2.111	0.025
OCBM	**17.202**	**0.061**
The study in [94]	-	4.48 (m)

The experimental results presented in Tables (4.16) and (4.17) demonstrate that the proposed model, OCBM, achieved the highest performance in most of the evaluation metrics with a low false positive rate (FPR). Additionally, the OCBM classifier exhibited significantly reduced training and testing times compared to other ML classifiers and the study mentioned in [94]. Although some ML classifiers may have shorter testing times, the OCBM classifier proves its efficacy when considering their overall performance. Furthermore, the OCBM utilized only 300 boosted trees and a maximum depth of five, in contrast to the CatBoost (baseline) model's utilization of 600 boosted trees and a maximum depth of seven. As a result, the goal of detecting DoS/DDoS attacks was successfully achieved using the OCBM classifier.

4.7 Results of FTDNN Model

This dissertation proposes a reliable and accurate intrusion detection model for binary prediction of most recent DoS/DDoS attacks by developing the DNN structure and hyper-parameters using a RSCV technique. By leveraging this technique and HFR approach, the proposed model achieved improved performance in terms of assessment metrics,

making it a promising solution for detecting such attacks. The best structure of FTDNN obtained consists of the number of hidden layers was two, number of neurons for every hidden layer was 128, ReLU was used as activation function in two hidden layers, learning rate was 0.001, optimizer algorithm was Adam, and the last layer of the FTDNN model utilized the softmax function.

The FTDNN model was optimized through three techniques: Early stopping, dropout layer, and batch normalization layer. Early stopping depends on accuracy metric with patience of 6 was utilized to avoid too much training. Dropout layer with a specified probability of 0.02 was added after each hidden layer to evade overfitting issue. Batch normalization was also added to speed-up the DNN training process.

In addition to other parameters that play a vital role in DNN training convergence and non-fall in local minima. For instance, batch size was set to be 1000 for InSDN, CICDDoS-2019 datasets, and 500 for SNMP dataset. The number of epochs was 50 for all datasets.

The FTDNN model was utilized the following procedures with all datasets (InSDN, CICDDoS-2019, and SNMP): (1) the Hold-out method was utilized to train and test the proposed model, wherein all datasets were separated into 70% for training and 30% for testing; (2) the binary-crossentropy was used as loss function; (3) unlike most other studies, this dissertation assesses the performance of FTDNN model by computing various evaluation metrics including detection rate, accuracy, precision, F1-score, and FPR. Moreover, the time complexity for FTDNN model was also considered in evaluated its performance efficacy.

4.7.1 Experiments on InSDN Dataset using FTDNN

The proposed model, FTDNN, was trained and validated on the InSDN dataset after pre-processing it to be in a suitable form when fed into the feature reduction stage (see Sections 4.4 and 4.5). Subsequently, the FTDNN model utilized the HFR approach to select the best subset of features (14 *f.*) from the InSDN dataset, as shown in Table (4.5). Then, the structure of the FTDNN was constructed with fine-tuned hyperparameters using a RSCV technique. Finally, Tables (4.18) and (4.19) show that the outcomes of the FTDNN model, FTDNN without utilized HFR approach (with no feature reduction), shallow DNN model, LSTM model, and the results of existing studies curried out on InSDN dataset. Figure (4.9) shows the performance of FTDNN on InSDN dataset.

Table (4.18): Comparison performance of the proposed model (FTDNN) on InSDN dataset

Classifier	Accuracy	Precision	Recall	F1-Score	FPR
FTDNN without HFR.	0.9895	0.9896	0.9895	0.9895	0.0225
Shallow-DNN	0.9691	0.9688	0.9691	0.9687	0.0592
The proposed model (FTDNN)	**0.9908**	**0.9907**	**0.9908**	**0.9907**	**0.0101**
LSTM	0.9900	0.9900	0.9900	0.9899	0.0237
The study in [20]	0.905	0.93	0.93	0.93	-
The study in [25]	0.977	-	-	-	-

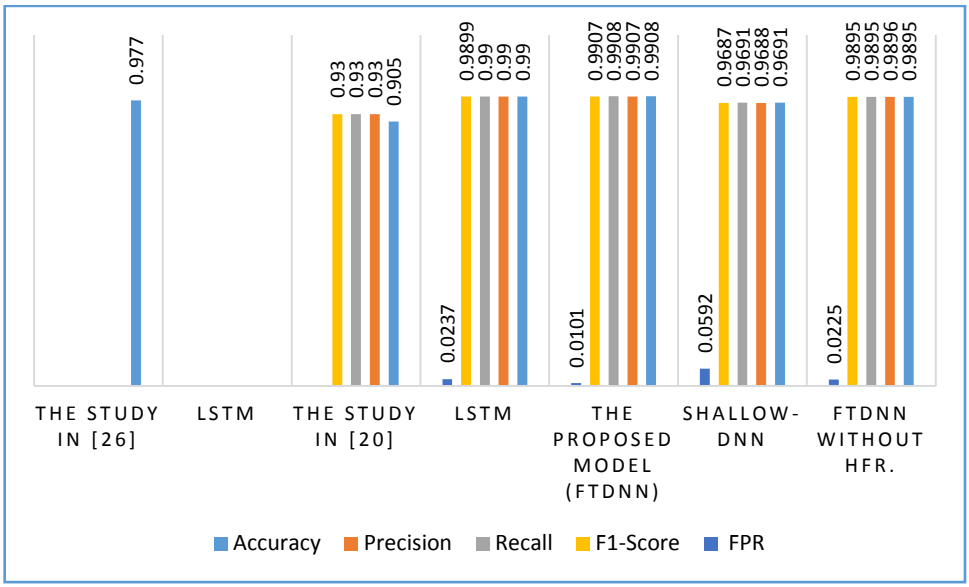


Figure (4.9) Performance of FTDNN on InSDN Dataset

From Table (4.18), it can be seen that our proposed model achieves the highest performance in all evaluation metrics with the lowest FPR compared to other DNN-based models and earlier studies using InSDN. The HFR and RSCV approaches followed for the FTDNN play a vital role in accurately attack detection.

Table (4.19): Summary of training time and test time of the proposed model (FTDNN) on InSDN dataset

Classifier	Training Time (sec.)	Testing Time (sec.)
FTDNN without HFR.	324.588	8.967
Shallow-DNN	80.911	0.0592
The proposed model (FTDNN)	**239.442**	**6.241**
LSTM	483.538	10.390
The study in [20]	147.548	13.546

The efficiency of the FTDNN model is evaluated by comparing its processing time to that of the FTDNN classifier with full features, shallow-DNN model, LSTM model, and existing work done on the same dataset. The FTDNN model with fourteen best features significantly decreases the time complexity for training and response (test) time, as shown in Table (4.19). A previous study [20] achieved a low training

time but did not have significant results, as shown in Table (4.18), which proves the importance of our results. As a result, this model proved to be the best performing model for the InSDN dataset, with reduced processing time as well.

4.7.2 Experiments on SNMP Dataset using FTDNN

For the further evolution of the FTDNN model, the SNMP dataset was used in the training and testing phases. The importance of the FTDNN depends on its ability to correctly classify network traffic into the correct type (normal or DoS/DDoS attack). Table (4.6) displays the best selected features for SNMP dataset (9 *f.*). The evaluation results are presented in Table (4.20) as well as all other models and existing work performed on SNMP dataset. Also, Table (4.21) shows the processing time (training and testing) for these models. Figure (4.10) displays the performance of FTDNN on SNMP dataset.

Table (4.20): Comparison performance of the proposed model (FTDNN) on SNMP dataset

Classifier	Accuracy	Precision	Recall	F1-Score	FPR
FTDNN without HFR.	0.9973	0.9875	0.9873	0.9874	0.02
Shallow-DNN	0.9860	0.9862	0.9860	0.9861	0.05
The proposed model (FTDNN with HFR)	**0.9998**	**0.9996**	**0.9997**	**0.9996**	**0.01**
LSTM	0.9960	0.9960	0.9960	0.9959	0.03
The Study in [95]	0.9886	-	-	-	-
The study in [96]	0.9838	-	-	-	-

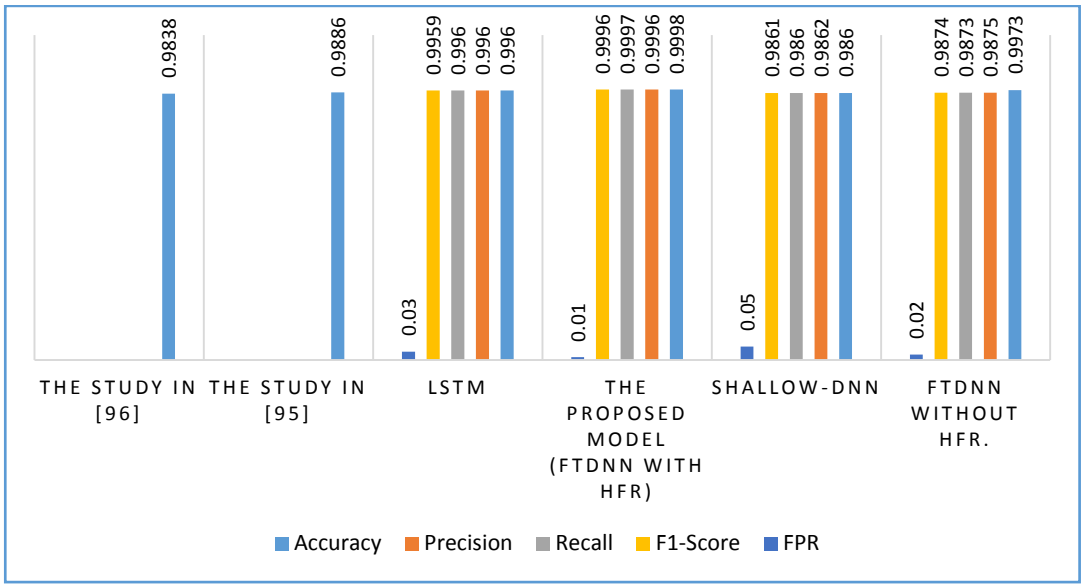


Figure (4.10) Performance of FTDNN on SNMP Dataset.

By looking at Table (4.20) and figure (4.10), it was found that the integrating output of the proposed HFR approach with the FTDNN classifier resulted in the highest detection rate of 0.9997, accuracy of 0.9998, precision of 0.9996, F1-score of 0.9996, with a very low FPR of 0.01 compared to other models and existing works. Furthermore, this integration leads to significant reduction in the training and testing times compared to the FTDNN model with full features of the dataset as shown in Table (4.21). In contrast, the Shallow-DNN classifier exhibited the shortest training and testing times among all other models. However, despite its efficiency in terms of computational speed, the Shallow-DNN model did not perform well when considering accuracy, precision, recall, F1-Score, and FPR rates.

Table (4.21) Summary of training time and test time of the proposed model (FTDNN) on SNMP dataset

Classifier	Training Time (sec.)	Testing Time (sec.)
FTDNN without HFR.	34.248	0.498
Shallow-DNN	10.977	0.212
The proposed model (FTDNN with HFR)	**11.207**	**0.270**
LSTM	29.566	0.323

4.7.3 Experiments on CICDDoS-2019 Dataset using FTDNN

The FTDNN model was evaluated using the CICDDoS-2019 dataset with the aim of improving the detection of DoS/DDoS attacks while minimizing the FPR and detection time. To achieve this, an efficient HFR approach was used for feature reduction, and a RSCV technique was employed to construct an appropriate DNN structure. The HFR approach selected 12 out of all features, as shown in Table (4.7). These selected features were utilized for training and testing the FTDNN classifier.

To measure the performance of the proposed FTDNN model, various assessment metrics such as accuracy, precision, detection rate, F1-score, and FPR were used, which are presented in Table (4.22) and figure (4.11). In addition to these metrics, the training time and testing time were also considered in this model using CICDDoS-2019 dataset, as shown in Table (4.23).

Table (4.22): Comparison performance of the proposed model (FTDNN) on CICDDoS-2019 dataset

Classifier	Accuracy	Precision	Detection rate	F1-Score	FPR
FTDNN with no HFR.	0.9982	0.9984	0.9980	0.9982	0.0060
Shallow-DNN	0.9902	0.9901	0.9902	0.9901	0.0873
The proposed (FTDNN with HFR)	**0.9994**	**0.9995**	**0.9994**	**0.9994**	**0.0041**
LSTM	0.9941	0.9946	0.9939	0.9942	0.0444
The study in [21]	0.990	0.995	0.990	0.990	-
The study in [22]	0.9438	0.9408	0.9789	0.9594	-
The study in [97]	0.9834	0.9791	0.9848	0.9818	-
The study in [98]-CNN	0.954	0.933	0.924	0.890	-
The study in [98]-MLP	0.925	0.844	0.942	0.890	-

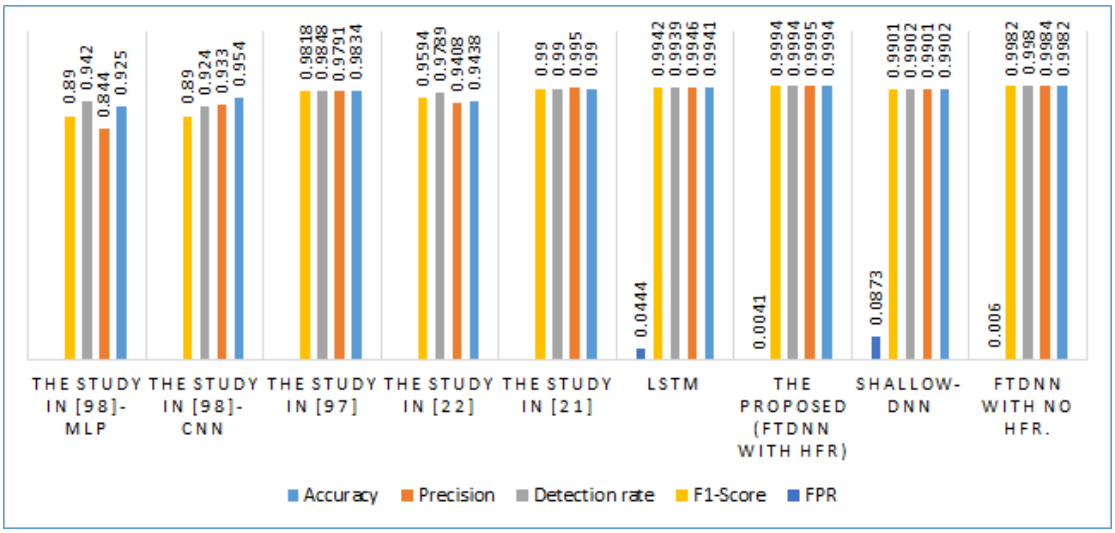


Figure (4.11) Performance of FTDNN in SNMP Dataset.

Table (4.23): Summary of training time and test time of the proposed model (FTDNN) on CICDDoS-2019 dataset

Classifier	Training Time (sec.)	Testing Time (sec.)
FTDNN without HFR.	218.976	12.434
Shallow-DNN	79.898	5.893
The proposed (FTDNN with HFR)	**91.211**	**7.110**
LSTM	287.399	15.588

The empirical results in Tables (4.22) and (4.23) show the effect of the HFR approach to achieve the FTDNN model with an accuracy of 99.94%, detection rate (recall) of 99.94%, precision of 99.95%, and F1 of 99.94% with low FPR of 0.0041, and outperformed other approaches. In addition, these results indicate that the FTDNN model has achieved better performance compared to the earlier studies. Furthermore, the testing time was significantly reduced from 12.434 s to 7.110 s and training time from 218.976 s to 91.211 s in compassion between the HFR-FTDNN and FTDNN with no feature reduction approach HFR. Despite exhibited the shortest training and testing times compared to all models, the Shallow-DNN classifier's computational efficiency did not align with robust performance in accuracy, precision, recall, F1-Score, and FPR rates.

Consequently, the Shallow-DNN classifier is deemed not beneficial for the attack detection system.

4.8 Results of Zero-Day DoS/DDoS attack prediction

Due to the absence of real-world datasets, researchers usually assess their classifiers on the similar dataset which they utilized for training stage. Therefore, their performance models are not adequate in identifying a zero-day attack (in this dissertation DoS/DDoS attacks). Zero-day attacks are dangerous and pose serious threats to SDNs for the reason that it has a diverse feature distribution when compared to what ML and DNN based models have been trained on.

Thus, unlike most other studies which it focus only on detection known attacks based on previously known profiles. To end this gap and to simulate unknown (or zero-day) attacks in a real-world environment, this dissertation evaluates the performance of both proposed ML-based model (OCBM) and DNN-based model (FTDNN) in detecting zero-day DoS/DDoS attacks. First, the proposed models are individually trained on the prepared InSDN and CICDDoS-2019 datasets. Then, the trained models are individually assessed on the CICIDS-2017 dataset to ensure their effectiveness's. Finally, the CICIDS-2017 dataset merged with the InSDN dataset in the first case, and with CICDDoS-2019 dataset in second case, respectively. Afterwards, the proposed models are re-trained again to verify their performances. All experiments for both proposed OCBM and FTDNN models on these datasets are illustrated in Cases 1-4 for each model in Table (4.25) and (4.26).

Table (4.24): Results of the proposed model (OCBM) before and after training on Zero-Day attacks

Cases	Training Dataset	Evaluation Dataset	Accuracy	Precision	Recall	F1-Score	FPR
Case -1-	InSDN	CICIDS 2017	0.9727	0.9734	0.9727	0.9720	0.0668
Case -2-	CICDDoS2019	CICIDS 2017	0.9914	0.9914	0.9914	0.9914	0.0151
Case -3-	CICIDS2017+InSDN	InSDN+ CICDDoS2017	0.9833	0.9841	0.9833	0.9826	0.0568
Case -4-	CICDDoS2019+ CICIDS2017	CICIDS 2017+ CICDDoS2019	0.9991	0.9991	0.9991	0.9991	0.0036

All utilized datasets are divided into 70% for training and 30% for testing. Table (4.24) depicts the experimental outcomes for the OCBM model on zero-day attacks. In case 1, the results indicate that the proposed model, OCBM, archives for example, the detection accuracy of 97.27% for training the InSDN dataset and validation on the CICIDS-2017 dataset that has diverse distribution than the ones which the OCBM model was trained earlier. Whereas, for case 3 in the same Table (4.24), we observed that the detection accuracy has increased to 98.33% when combined the InSDN with CICIDS-2017 datasets. In the same manner for CICDDoS-2019 dataset, in case 2, the score accuracy of 99.14% for training the CICDDoS-2019 dataset and validation on the CICIDS-2017 dataset. While, in case 4, the score accuracy has increased to 99.91% for evaluation combined the CICDDoS-2019 with CICIDS-2017 datasets. Therefore, due to experimental outcomes, the performance of OCBM model for detection normal and DoS/DDoS attack for validation dataset has improved after re-training of the model.

Table (4.25): Results of the proposed model (FTDNN) before and after training on Zero-Day attacks

Cases	Training Dataset	Evaluation Dataset	Accuracy	Precision	Recall	F1-Score	FPR
Case -1-	InSDN	CICIDS2017	0.9684	0.9688	0.9684	0.9676	0.0729
Case -2-	CICDDoS2019	CICIDS2017	0.9535	0.9560	0.9535	0.9512	0.1146
Case -3-	CICIDS2017 +InSDN	InSDN+CICDDoS2017	0.9749	0.9756	0.9749	0.9742	0.0630
Case -4-	CICDDoS2019+ CICIDS2017	CICIDS2017+ CICDDoS2019	0.9925	0.9924	0.9925	0.9924	0.0376

Table (4.25) illustrates the experimental results for the FTDNN model on unknown DoS/DDoS attacks. In case 1, the outcomes investigate that the proposed model, FTDNN, accomplishes for instance, the score accuracy of 96.84% for training the InSDN dataset and validation on the CICIDS-2017 dataset. While, for case 3 in the same Table (4.25), we observed that the detection accuracy has increased to 97.49% for evaluation combined the InSDN with CICIDS-2017 datasets. In the same manner for CICDDoS-2019 dataset, in case 2, the score accuracy of 95.35% for training the CICDDoS-2019 dataset and validation on the CICIDS-2017 dataset. Whereas, in case 4, the score accuracy has increased to 99.25% for evaluation combined the CICDDoS-2019 with CICIDS-2017 datasets. Thus, owing to experimental analysis results, the performance of FTDNN model for detection benign and DoS/DDoS attack for evaluation dataset has enhanced after re-training of the FTDNN model.

In summary, these experiments was performed to mimic the real-world assault situation and assess the detection performance of the

proposed models, OCBM and FTDNN, when exposed to zero-day or new unknown attacks.

4.9 Results of DSBHED Model

This dissertation proposes a new soft-based heterogeneous ensemble detection (DSBHED) model that relies on a combination of hybrid feature reduction and optimized ML-based classifiers to improve detection of multi-class attacks. The proposed DSBHED model is constructed via several stages. First, the initial datasets are prepared. Second, three different feature reduction approaches are applied, and their outcomes are combined based on majority voting and mean techniques. In other words, this model also uses an HFR approach to obtain the best subset of features, but for a multi-class detection task.

In the third stage of constructing the DSBHED model, three different heterogeneous classification algorithms were performed, including RF, KNN, and LightGBM. These algorithms, including RF and LightGBM classifiers, were optimized using the GSCV method to select their best hyperparameters, as illustrated in Table (3.1), and produce the ORF and OLighGBM classifiers. This optimization process with the HFR approach helped to produce developed model that could better adapt to detect multi-class attacks and enhance the accuracy and reliability of intrusion detection. The results of all classifiers (ORF, KNN, and OLightGBM) were then combined using a soft-voting technique. Finally, the performance of the proposed DSBHED model was evaluated using the InSDN, SNMP, and CICDDoS-2019 datasets.

4.9.1 Experiments on the InSDN dataset using DSBHED

In the first step, the original InSDN dataset is not appropriate to directly construct ML-based model, DSBHED; therefore some pre-

processing methods are performed to transform InSDN dataset into a readable and understandable format, as described in Section (4.4).

In the second step, the prepared InSDN dataset contains 83 features but not all of these features can be utilized to classify multi-class assaults. Therefore, the proposed HFR approach is applied to select the best subset of features; this approach obtained only 14 out of 83 features which then used in the next stage.

Finally, the DSBHED was validated on InSDN dataset and its performance was measured according to the evaluation metrics. Our results were compared with the baseline models and previous studies conducted on this dataset. As a result, the proposed model achieved the best results in detecting multi-class attacks, as shown in Table (4.26) and Figure (4.12). In addition, the training and testing time were also calculated and presented in Table (4.27). Figure (4.13) shows the confusion matrix of the DSBHED model.

Table (4.26): The Outcomes of the Proposed Model (DSBHED) based on InSDN Dataset

Classifier	Accuracy	Precision	Recall	F1-Score	FPR
KNN	0.9841	0.9842	0.9841	0.9840	0.0026
LightGBM	0.9419	0.9481	0.9419	0.9403	0.0108
LR	0.8650	0.8812	0.8650	0.8579	0.0230
Hard-ensemble	0.9981	0.9982	0.9981	0.9981	0.0004
SVM	0.9347	0.9407	0.9347	0.9351	0.0549
DSBHED	**0.9989**	**0.9989**	**0.9988**	**0.9988**	**0.0002**
The study in [25]	0.955	-	-	-	-
The study in [99]	0.9980	0.9980	0.9980	-	-

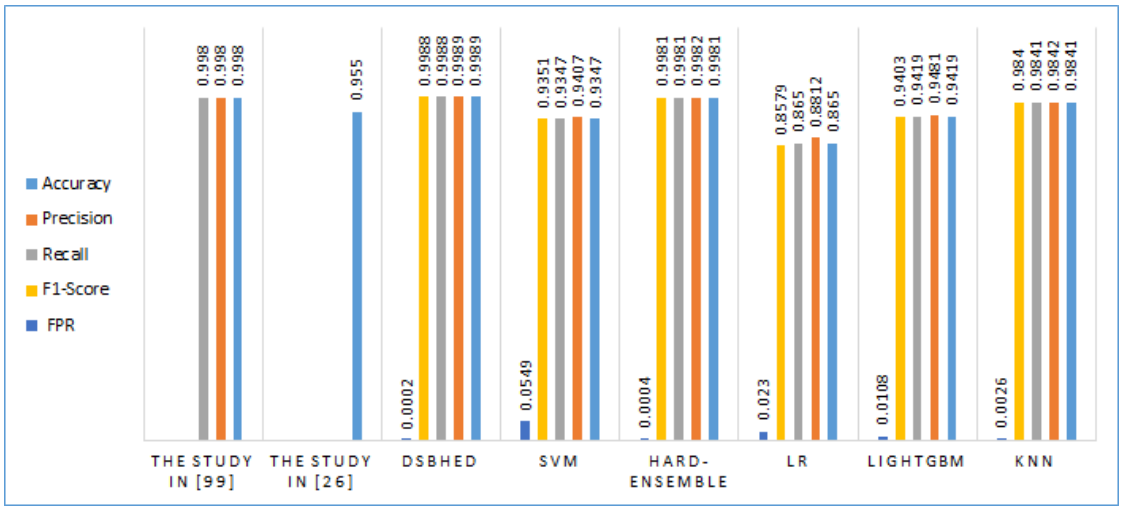


Figure (4.12) Performance of DSBHED on InSDN Dataset.

Confusion Matrix for the proposed model (DSBHED) on InSDN:

	BFA	DDoS	DoS	Normal	Probe
BFA	337	0	9	1	51
DDoS	0	36740	0	2	0
DoS	3	0	16019	7	11
Normal	1	0	3	20530	20
Probe	5	0	19	10	29287

Figure (4.13): The DSBHED Confusion-Matrix of InSDN Dataset.

Table (4.27): The Processing Time of the Proposed Model (DSBHED) based on the InSDN Dataset

Classifier	Training Time (sec.)	Testing Time (sec.)
KNN	9.263	49.179
LightGBM	64.627	6.569
LR	11.020	0.039
Hard-ensemble	52.731	48.427
SVM	15.502	0.087
DSBHED	41.086	40.936
The study in [25]	1902	-

Table (4.27) shows that the proposed DSBHED model takes more time than LR, LightGBM, and SVM in terms of training and testing times. However, when taking into account its detection rate, precision, F-measure, FPR, and accuracy rates, the time taken by the proposed model appears to be acceptable.

4.9.2 Experiments on the SNMP dataset using DSBHED

The proposed DSBHED model was validated on the SNMP dataset, which included the BFA attack and six types of DoS/DDoS assaults. A subset of important features was determined using a multi-class based HFR approach (10 *features*). To validate the effectiveness of the model, we used all assessment measures as indicators. Tables (4.28) and (4.29) demonstrate the effectiveness and efficiency of the DSBHED model on the SNMP dataset. Figure (4.14) shows the performance of DSBHED on SNMP dataset. While, Figure (4.15) depicts the confusion matrix of the DSBHED model for multi-class assault detection.

Table (4.28): The Outcomes of the Proposed Model (DSBHED) based on the SNMP Dataset

Classifier	Accuracy	Precision	Recall	-F1 Score	FPR
XGBoost	0.9997	0.9997	0.9998	0.9997	0.0002
GBM	0.9998	0.9997	0.9998	0.9997	0.0002
LR	0.9093	0.9123	0.9093	0.9089	0.0131
SVM	0.8801	0.8893	0.8802	0.8793	0.0171
DSBHED	**0.9999**	**0.9998**	**0.9999**	**0.9998**	**0.0001**

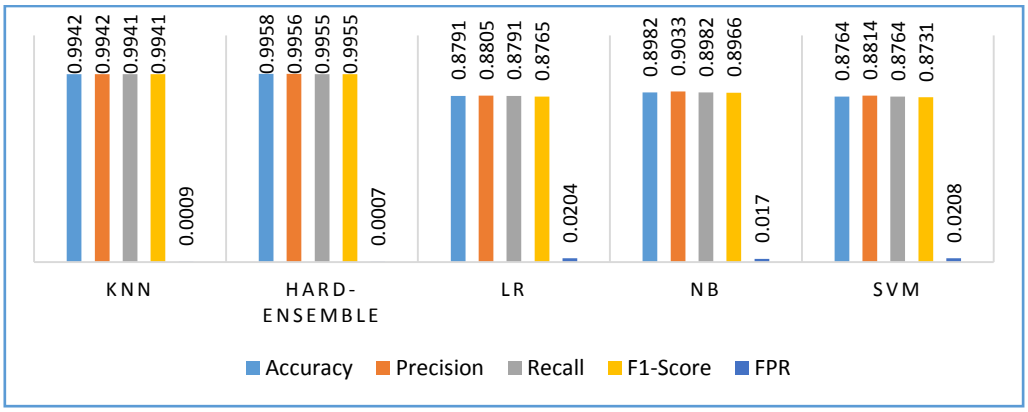


Figure (4.14) Performance of DSBHED on SNMP Dataset.

Confusion Matrix for the proposed model (DSBHED) on SNMP:

	bruteForce	httpFlood	icmp-echo	slowloris	slowpost	udp-flood	tcp-syn	normal
bruteForce	60	0	0	0	0	0	0	0
httpFlood	0	147	0	0	0	0	0	0
icmp-echo	0	0	199	0	0	0	0	0
slowloris	0	0	0	194	0	0	3	0
slowpost	0	0	0	0	246	0	0	0
udp-flood	0	0	0	0	0	143	0	0
tcp-syn	0	0	0	0	0	0	276	0
normal	0	0	0	0	0	0	0	232

Figure (4.15): The DSBHED Confusion-Matrix of SNMP Dataset.

Table (4.29): The Processing Time of the Proposed Model (DSBHED) based on the SNMP Dataset

Classifier	Training Time (sec.)	Testing Time (sec.)
XGBoost	1.179	0.037
GBM	6.013	0.130
LR	0.196	0.085
SVM	0.246	0.076
DSBHED	1.201	0.135

Table (4.28) and Figure (4.14) display the evaluation results of the DSBHED model, along with other classifiers such as XGBoost, GBM, and etc. in detecting multi-class assaults on the SNMP dataset. Thanks to the best subset of features obtained by our HFR approach, the DSBHED, XGBoost, and GBM classifiers achieved good results across all evaluation metrics. Table (4.29) displays the training time and testing time for all baseline classifiers, including DSBHED. It indicates that the testing time for DSBHED is slightly longer than other ML-based techniques. However, this slight increase in testing time is acceptable when considering the excellent performance of DSBHED across various evaluation metrics such as accuracy, detection rate (recall), precision, F1-Score, and FPR, as shown in Table (4.28). In addition, a model that achieves a good balance between accuracy detection and computational speed (training and testing times) ensures efficient and effective in terms of IDSs.

4.9.3 Experiment on CICDDoS-2019 Dataset using DSBHED

To highlight the superiority of the proposed model in this dissertation, we compared its performance with different ML-based classifiers and previous studies carried out on the same dataset. Most important of features was (15 *features*) obtained via our HFR. These important features are then fed to the DSBHED to effectively identify different DDoS attack types. It is observed from the empirical results shown in Table (4.30) and Figure (4.16) that the proposed model, DSBHED, performs better in terms of all evaluation measures. Whereas, among all the base classifiers, SVM performs the worst in terms of all classification measures. In addition, the proposed model achieves promising results in comparison with the earlier studies. Therefore, the DSBHED performs as the best classifier, which results in a more accurate and stable model. Figure (4.17) presents the confusion matrix of the DSBHED model for multi-class assaults detection on the CICDDoS-2019 dataset.

Table (4.30): The Outcomes of the Proposed Model (DSBHED) based on the CICDDoS-2019 Dataset

Classifier	Accuracy	Precision	Recall	-F1 Score	FPR
KNN	0.9942	0.9942	0.9941	0.9941	0.0009
Hard-ensemble	0.9958	0.9956	0.9955	0.9955	0.0007
LR	0.8791	0.8805	0.8791	0.8765	0.0204
NB	0.8982	0.9033	0.8982	0.8966	0.0170
SVM	0.8764	0.8814	0.8764	0.8731	0.0208
DSBHED	**0.9965**	**0.9964**	**0.9963**	**0.9963**	**0.0006**
The study in [23]	0.8493	0.8465	0.8409	-	-
The study in [100]	0.9834	0.9791	0.9848	0.9818	-
The study in [101]	0.9625	0.96	0.96	0.96	-

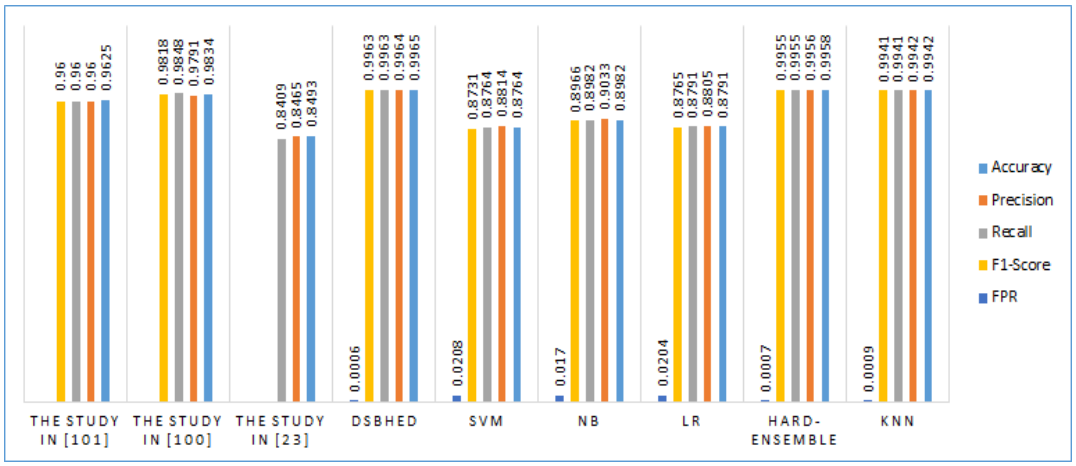


Figure (4.16) Performance of DSBHED on CICDDoS-2019 Dataset.

Confusion Matrix for the proposed model (DSBHED) on CICDDoS-2019:

	BENIGN	MSSQL	NTP	SNMP	UDP	Portmap	SYN
BENIGN	5675	0	0	0	0	8	2
MSSQL	0	16665	22	0	14	0	5
NTP	2	1	16341	1	3	0	95
SNMP	0	1	1	19716	1	0	0
UDP	1	26	150	0	17837	0	52
Portmap	19	0	0	0	0	17568	0
SYN	0	0	1	0	0	0	17892

Figure (4.17): The DSBHED Confusion-Matrix of CICDDoS-2019 Dataset.

Table (4.31): The Processing Time of the Proposed Model (DSBHED) based on the CICDDoS-2019 Dataset

Classifier	Training Time (sec.)	Testing Time (sec.)
KNN	2.020	63.183
Hard-ensemble	40.418	65.749
LR	12.062	0.027
NB	0.113	0.073
SVM	40.663	0.023
DSBHED	39.092	40.504

To further evaluate the effectiveness of the proposed model and highlight the impact of the HFR approach, we calculated the training and testing times for all ML models, as presented in Table (4.31). The findings indicate that the DSBHED model requires more testing time compared to the NB, SVM, and LR models. However, considering the

evaluation metrics of detection rate, accuracy, precision, F1-score, and FPR rates, this increase in testing time is reasonable and acceptable.

Chapter Five

Conclusions and Future Works

Chapter 5

Conclusions and Future Works

5.1 Conclusions

Although previous attempts aimed to enhance the effectiveness of IDSs by utilizing different ML techniques, existing IDSs are remain insufficient in terms of some measures. In this dissertation, we proposed efficient and effective ML-based (OCBM) and DNN-based (FTDNN) IDS models for binary detection of DoS/DDoS attacks against an SDN platform. In addition, a developed soft-based heterogeneous ensemble detection (DSBHED) model was proposed to detect multi-class attacks and attain sufficient complementarity with strong generalization.

Due to empirical outcomes of proposed models, many conclusions can be drawn:

1- The proposed HFR approach can significantly reduce the number of features required for binary-based detection task. Specifically, it reduced 83% of the InSDN features, 74% of the SNMP features, and 87% of the CICDDoS-2019 features. For the multiclass detection task, it was also effective in reducing the number of required features. It reduced 83% of the InSDN features, 71% of the SNMP features, and 83% of the CICDDoS-2019 features. These results demonstrate the effectiveness of the HFR approach in reducing the number of features required for effective detection, while maintaining high levels of performance.

2- The proposed OCBM and FTDNN models have been validated on four datasets, namely InSDN, SNMP, CICDDoS-2019, and CICIDS-2017. The results demonstrated that these models are highly effective in identifying attacks across all evaluation metrics, including accuracy, precision, detection rate, and F-measure with

low FPRs while maintaining fast execution times. Hence, these findings highlight the effectiveness of the proposed models in detecting attacks on different datasets.

3- The performance of the DSBHED model was found to be very stable and excellent for detecting multiclass attacks across the InSDN, SNMP, and CICDDoS-2019 datasets. Compared with other studies, the DSBHED model consistently demonstrated superior performance. The ability to effectively detect multiclass attacks makes it a valuable tool for intrusion detection.
4- In this dissertation, the best hyperparameters for the ML-based proposed OCBM and DSBHED models are tuned using the grid search optimization technique, GSCV. In contrast, the RandomSearchCV (RSCV) technique is employed to design the best structure of the FTDNN model along with its hyperparameters.
5- Carried out data preparing especially, removed non-essential attributes, eliminated missing values, and noise, thereby providing better training data.
6- In the context of DNN, using the three methods of dropout, batch normalization, and early stopping helped to improve the performance of the FTDNN model.
7- The stable performance of proposed models (OCBM, FTDNN, and DSBHED) across different datasets make them strong candidates for real-time detection processes.

5.2 Future works

Some future works can be viewed below:

1- Dataset: we plan to incorporate additional IDS datasets, including the newly available IoTdataset 2023, to expand the scope of our research and enhance the diversity of attack scenarios explored.
2- Feature reduction: Trying to use other feature reduction algorithms to select fewer important features as much as possible and compare them with HFR model.
3- Classification approach: We intend to use other field of classification such as unsupervised algorithms to compare it with the proposed models on the same datasets.
4- Optimization techniques: Studying other algorithms such as (Bayesian optimization algorithm, Genetic algorithm, and so on) to optimize hyper parameters of the proposed models (ML-based and DNN-based).
5- We plan to apply the proposed models in data centers which utilize real SDN network. Furthermore, we aim to expand upon the current study by enhancing its capabilities to include preventing multiple types of attacks based on different rules.

References

References

[1] H. Kim and N. Feamster, “Improving network management with software defined networking,” IEEE Communications Magazine, vol. 51, no. 2, pp. 114–119, Feb. 2013, doi: 10.1109/mcom.2013.6461195.

[2] D. Li, C. Yu, Q. Zhou, and J. Yu, “Using SVM to Detect DDoS Attack in SDN Network,” IOP Conference Series: Materials Science and Engineering, vol. 466, p. 012003, Dec. 2018, doi: 10.1088/1757-899x/466/1/012003.

[3] K. Bhushan and B. B. Gupta, “Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment,” Journal of Ambient Intelligence and Humanized Computing, vol. 10, no. 5, pp. 1985–1997, Apr. 2018, doi: 10.1007/s12652-018-0800-9.

[4] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, “A survey of intrusion detection in Internet of Things,” Journal of Network and Computer Applications, vol. 84, pp. 25–37, Apr. 2017, doi: 10.1016/j.jnca.2017.02.009.

[5] M. I. Kareem and M. N. Jasim, “The Current Trends of DDoS Detection in SDN Environment,” 2021 2nd Information Technology To Enhance e-learning and Other Application (IT-ELA), Dec. 2021, doi: 10.1109/it-ela52201.2021.9773744.

[6] P. Kumar, M. Tripathi, A. Nehra, M. Conti, and C. Lal, “SAFETY: Early Detection and Mitigation of TCP SYN Flood Utilizing Entropy in SDN,” IEEE Transactions on Network and Service Management, vol. 15, no. 4, pp. 1545–1559, Dec. 2018, doi: 10.1109/tnsm.2018.2861741.

[7] S. Yu, J. Zhang, J. Liu, X. Zhang, Y. Li, and T. Xu, “A cooperative DDoS attack detection scheme based on entropy and ensemble learning in SDN,” EURASIP Journal on Wireless Communications and

Networking, vol. 2021, no. 1, Apr. 2021, doi: 10.1186/s13638-021-01957-9.

[8] A. Mishra, N. Gupta, and B. B. Gupta, “Defense mechanisms against DDoS attack based on entropy in SDN-cloud using POX controller,” Telecommunication Systems, vol. 77, no. 1, pp. 47–62, Jan. 2021, doi: 10.1007/s11235-020-00747-w.

[9] K. S. Sahoo, D. Puthal, M. Tiwary, J. J. P. C. Rodrigues, B. Sahoo, and R. Dash, “An early detection of low rate DDoS attack to SDN based data center networks using information distance metrics,” Future Generation Computer Systems, vol. 89, pp. 685–697, Dec. 2018, doi: 10.1016/j.future.2018.07.017.

[10] N. Meti, D. G. Narayan, and V. P. Baligar, “Detection of distributed denial of service attacks using machine learning algorithms in software defined networks,” 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Sep. 2017, doi: 10.1109/icacci.2017.8126031.

[11] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, “Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks,” 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), Jun. 2018, doi: 10.1109/netsoft.2018.8460090.

[12] S. Vimala and J. Dhas, “SDN Based DDoS Attack Detection System by Exploiting Ensemble Classification for Cloud Computing,” International Journal of Intelligent Engineering and Systems, vol. 11, no. 6, pp. 282–291, Dec. 2018, doi: 10.22266/ijies2018.1231.28.

[13] A. Bansal and S. Kaur, “Extreme Gradient Boosting Based Tuning for Classification in Intrusion Detection Systems,” Advances in Computing and Data Sciences, pp. 372–380, 2018, doi: 10.1007/978-981-13-1810-8_37.

[14] V. Deepa, K. M. Sudar, and P. Deepalakshmi, “Design of Ensemble Learning Methods for DDoS Detection in SDN Environment,” 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), Mar. 2019, doi: 10.1109/vitecon.2019.8899682.

[15] M. Catillo, M. Rak, and U. Villano, “Discovery of DoS attacks by the ZED-IDS anomaly detector,” Journal of High Speed Networks, vol. 25, no. 4, pp. 349–365, Nov. 2019, doi: 10.3233/jhs-190620.

[16] L. Tan, Y. Pan, J. Wu, J. Zhou, H. Jiang, and Y. Deng, “A New Framework for DDoS Attack Detection and Defense in SDN Environment,” IEEE Access, vol. 8, pp. 161908–161919, 2020, doi: 10.1109/access.2020.3021435.

[17] S. Dong and M. Sarem, “DDoS Attack Detection Method Based on Improved KNN With the Degree of DDoS Attack in Software-Defined Networks,” IEEE Access, vol. 8, pp. 5039–5048, 2020, doi: 10.1109/access.2019.2963077.

[18] J. A. Perez-Diaz, I. A. Valdovinos, K.-K. R. Choo, and D. Zhu, “A Flexible SDN-Based Architecture for Identifying and Mitigating Low-Rate DDoS Attacks Using Machine Learning,” IEEE Access, vol. 8, pp. 155859–155872, 2020, doi: 10.1109/access.2020.3019330.

[19] R. Swami, M. Dave, and V. Ranga, “Voting-based intrusion detection framework for securing software-defined networks,” Concurrency and Computation: Practice and Experience, vol. 32, no. 24, Jul. 2020, doi: 10.1002/cpe.5927.

[20] M. Said Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, “Network Anomaly Detection Using LSTM Based Autoencoder,” Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks, Nov. 2020, doi: 10.1145/3416013.3426457.

[21] D. Srilatha and N. Thillaiarasu, “DDoSNet: A Deep Learning Model for detecting Network Attacks in Cloud Computing,” 2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA), Sep. 2022, doi: 10.1109/icirca54612.2022.9985524.

[22] M. P. Novaes, L. F. Carvalho, J. Lloret, and M. L. Proença, “Adversarial Deep Learning approach detection and defense against DDoS attacks in SDN environments,” Future Generation Computer Systems, vol. 125, pp. 156–167, Dec. 2021, doi: 10.1016/j.future.2021.06.047.

[23] S. Rajagopal, P. P. Kundapur, and H. K. S., “Towards Effective Network Intrusion Detection: From Concept to Creation on Azure Cloud,” IEEE Access, vol. 9, pp. 19723–19742, 2021, doi: 10.1109/access.2021.3054688.

[24] T. T. Khoei, G. Aissou, W. C. Hu, and N. Kaabouch, “Ensemble Learning Methods for Anomaly Intrusion Detection System in Smart Grid,” 2021 IEEE International Conference on Electro Information Technology (EIT), May 2021, doi: 10.1109/eit51626.2021.9491891.

[25] L. Kou, S. Ding, T. Wu, W. Dong, and Y. Yin, “An Intrusion Detection Model for Drone Communication Network in SDN Environment,” Drones, vol. 6, no. 11, p. 342, Nov. 2022, doi: 10.3390/drones6110342.

[26] N. Ahuja, G. Singal, D. Mukhopadhyay, and N. Kumar, “Automated DDOS attack detection in software defined networking,” Journal of Network and Computer Applications, vol. 187, p. 103108, Aug. 2021, doi: 10.1016/j.jnca.2021.103108.

[27] J. C. Correa Chica, J. C. Imbachi, and J. F. Botero Vega, “Security in SDN: A comprehensive survey,” Journal of Network and Computer

Applications, vol. 159, p. 102595, Jun. 2020, doi: 10.1016/j.jnca.2020.102595.

[28] Kupreev, O.; Badovskaya, E.; Gutnikov, "A. DDoS Attacks in Q2 2020|Securelist." 2020. Available online: https://securelist.com/ddos-attacks-in-q2-2020/98077/ (accessed on 3 March 2021).

[29] N. G. B. Amma and S. Selvakumar, "Optimization of vector convolutional deep neural network using binary real cumulative incarnation for detection of distributed denial of service attacks," Neural Computing and Applications, vol. 34, no. 4, pp. 2869–2882, Oct. 2021, doi: 10.1007/s00521-021-06565-8.

[30] J. Singh and S. Behal, "Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions," Computer Science Review, vol. 37, p. 100279, Aug. 2020, doi: 10.1016/j.cosrev.2020.100279.

[31] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-Defined Networking: Challenges and research opportunities for Future Internet," Computer Networks, vol. 75, pp. 453–471, Dec. 2014, doi: 10.1016/j.comnet.2014.10.015.

[32] R. U and K. E, "Distributed Denial of Service Attacks Prevention, Detection and Mitigation – A Review," SSRN Electronic Journal, 2021, doi: 10.2139/ssrn.3852902.

[33] Y. Xu and Y. Liu, "DDoS attack detection under SDN context," IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, Apr. 2016, doi: 10.1109/infocom.2016.7524500.

[34] Yujie Xie and Pankoo Kim, "Novel privacy vulnerabilities and challenges of OpenFlow-based SDN in network security," Research Briefs on Information and Communication Technology Evolution, vol. 3, pp. 84–90, Oct. 2017, doi: 10.56801/rebicte.v3i.47.

[35] B. Pfaff, “The Open vSwitch Database Management Protocol,” Dec. 2013, doi: 10.17487/rfc7047.

[36] S. Ogasawara and Y. Takahashi, “Performance analysis of traffic classification in an OpenFlow protocol,” 2016 Cloudification of the Internet of Things (CIoT), Nov. 2016, doi: 10.1109/ciot.2016.7872908.

[37] B. P. R. Killi and S. V. Rao, “Controller placement in software defined networks: A Comprehensive survey,” Computer Networks, vol. 163, p. 106883, Nov. 2019, doi: 10.1016/j.comnet.2019.106883.

[38] W. Braun and M. Menth, “Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices,” Future Internet, vol. 6, no. 2, pp. 302–336, May 2014, doi: 10.3390/fi6020302.

[39] N. McKeown et al., “OpenFlow,” ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69–74, Mar. 2008, doi: 10.1145/1355734.1355746.

[40] N. Xue, D. Guo, J. Zhang, J. Xin, Z. Li, and X. Huang, “OpenFunction for Software Defined IoT,” in 2021 International Symposium on Networks, Computers and Communications (ISNCC), 2021, pp. 1–8.

[41] J. Benabbou, K. Elbaamrani, and N. Idboufker, “Security in OpenFlow based SDN, opportunities and challenges,” Photonic Netw. Commun., vol. 37, no. 1, pp. 1–23, 2019.

[42] K. Nam and K. Kim, “A study on sdn security enhancement using open source ids/ips suricata,” in 2018 International Conference on Information and Communication Technology Convergence (ICTC), 2018, pp. 1124– 1126.

[43] A. Mendiola, J. Astorga, E. Jacob, and M. Higuero, “A Survey on the Contributions of Software-Defined Networking to Traffic

Engineering," IEEE Communications Surveys & Tutorials, vol. 19, no. 2, pp. 918–953, 2017, doi: 10.1109/comst.2016.2633579.

[44] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, "Software-defined networking (SDN): a survey," Security and Communication Networks, vol. 9, no. 18, pp. 5803–5833, Dec. 2016, doi: 10.1002/sec.1737.

[45] A. Dawoud, S. Shahristani, and C. Raun, "Software-Defined Network Security," Networks of the Future, pp. 89–100, Oct. 2017, doi: 10.1201/9781315155517-5.

[46] P. Dong, X. Du, H. Zhang, and T. Xu, "A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows," 2016 IEEE International Conference on Communications (ICC), May 2016, doi: 10.1109/icc.2016.7510992.

[47] D. Yuan and J. Zhong, "A lab implementation of SYN flood attack and defense," Proceedings of the 9th ACM SIGITE conference on Information technology education, Oct. 2008, doi: 10.1145/1414558.1414575.

[48] A. Akhunzada, E. Ahmed, A. Gani, M. K. Khan, M. Imran, and S. Guizani, "Securing software defined networks: taxonomy, requirements, and open issues," IEEE Communications Magazine, vol. 53, no. 4, pp. 36–44, Apr. 2015, doi: 10.1109/mcom.2015.7081073.

[49] R. Hofstede, A. Pras, A. Sperotto, and G. D. Rodosek, "Flow-Based Compromise Detection: Lessons Learned," IEEE Security & Privacy, vol. 16, no. 1, pp. 82–89, Jan. 2018, doi: 10.1109/msp.2018.1331021.

[50] C. V. Neu, C. G. Tatsch, R. C. Lunardi, R. A. Michelin, A. M. S. Orozco, and A. F. Zorzo, "Lightweight IPS for port scan in OpenFlow SDN networks," NOMS 2018 - 2018 IEEE/IFIP Network Operations

[51] N. Dayal, P. Maity, S. Srivastava, and R. Khondoker, “Research Trends in Security and DDoS in SDN,” Security and Communication Networks, vol. 9, no. 18, pp. 6386–6411, Dec. 2016, doi: 10.1002/sec.1759.

[52] Z. Ahmed, N. Afaqui, and O. Humayun, “Detection and Prevention of DDoS attacks on Software Defined Networks Controllers for Smart Grid,” International Journal of Computer Applications, vol. 181, no. 45, pp. 16–21, Mar. 2019, doi: 10.5120/ijca2019918572.

[53] “Software Defined Networking based Intrusion Detection System,” International Journal of Recent Trends in Engineering and Research, vol. 3, no. 5, pp. 475–480, Jun. 2017, doi: 10.23883/ijrter.2017.3252.m8yed.

[54] T. Saranya, S. Sridevi, C. Deisy, T. D. Chung, and M. K. A. A. Khan, “Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review,” Procedia Computer Science, vol. 171, pp. 1251–1260, 2020, doi: 10.1016/j.procs.2020.04.133.

[55] K. Gibert, M. Sànchez–Marrè, and J. Izquierdo, “A survey on pre-processing techniques: Relevant issues in the context of environmental data mining,” AI Communications, vol. 29, no. 6, pp. 627–663, Dec. 2016, doi: 10.3233/aic-160710.

[56] A. Jovic, K. Brkic, and N. Bogunovic, “A review of feature selection methods with applications,” 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), May 2015, doi: 10.1109/mipro.2015 .7160458.

[57] L. V. Filter and P. C. A. Filter, “Seven Techniques for Dimensionality Reduction,” 2014.

[58] U. Stańczyk, “Feature Evaluation by Filter, Wrapper, and Embedded Approaches,” Studies in Computational Intelligence, pp. 29–44, Dec. 2014, doi: 10.1007/978-3-662-45620-0_3.

[59] W. Qian, J. Huang, Y. Wang, and W. Shu, “Mutual information-based label distribution feature selection for multi-label learning,” Knowledge-Based Systems, vol. 195, p. 105684, May 2020, doi: 10.1016/j.knosys.2020.105684.

[60] G. Chandrashekar and F. Sahin, “A survey on feature selection methods,” Computers & Electrical Engineering, vol. 40, no. 1, pp. 16–28, Jan. 2014, doi: 10.1016/j.compeleceng.2013.11.024.

[61] M. Mafarja and S. Mirjalili, “Whale optimization approaches for wrapper feature selection,” Applied Soft Computing, vol. 62, pp. 441–453, Jan. 2018, doi: 10.1016/j.asoc.2017.11.006.

[62] V. Kumar, “Feature Selection: A literature Review,” The Smart Computing Review, vol. 4, no. 3, Jun. 2014, doi: 10.6029/smartcr.2014.03.007.

[63] Y. A. Alsariera, V. E. Adeyemo, A. O. Balogun, and A. K. Alazzawi, “AI Meta-Learners and Extra-Trees Algorithm for the Detection of Phishing Websites,” IEEE Access, vol. 8, pp. 142532–142542, 2020, doi: 10.1109/access.2020.3013699.

[64] S. G. K. Patro and K. K. sahu, “Normalization: A Preprocessing Stage,” IARJSET, pp. 20–22, Mar. 2015, doi: 10.17148/iarjset.2015.2305.

[65] P. Amudha, S. Karthik, and S. Sivakumari, “Classification Techniques for Intrusion Detection An Overview,” International Journal of Computer Applications, vol. 76, no. 16, pp. 33–40, Aug. 2013, doi: 10.5120/13334-0928.

[66] P. Sapate and S. A.Raut, “Survey on Classification Techniques for Intrusion Detection,” Computer Science & Information Technology (CS & IT), May 2014, doi: 10.5121/csit.2014.4523.

[67] J. T. Hancock and T. M. Khoshgoftaar, “CatBoost for big data: an interdisciplinary review,” Journal of Big Data, vol. 7, no. 1, Nov. 2020, doi: 10.1186/s40537-020-00369-8.

[68] C. C. Aggarwal, “Data Classification,” Data Mining, pp. 285–344, 2015, doi: 10.1007/978-3-319-14142-8_10

[69] D. Kirkpatrick and S. Zilles, “Competitive Search in Symmetric Trees,” Algorithms and Data Structures, pp. 560–570, 2011, doi: 10.1007/978-3-642-22300-6_47.

[70] Y M. Schonlau and R. Y. Zou, “The random forest algorithm for statistical learning”, vol. 20, no. 1, Mar. 2020, doi: 10.1177/1536867X20909688.

[71] Y. Liao and V. R. Vemuri, “Use of K-Nearest Neighbor classifier for intrusion detection,” Computers & Security, vol. 21, no. 5, pp. 439–448, Oct. 2002, doi: 10.1016/s0167-4048(02)00514-x.

[72] G. Toussaint, “Geometric Proximity Graphs For Improving Nearest Neighbor Methods In Instance-Based Learning And Data Mining,” International Journal of Computational Geometry & Applications, vol. 15, no. 02, pp. 101–150, Apr. 2005, doi: 10.1142/s0218195905001622.

[73] R. Khairy, A. Hussein, and H. ALRikabi, “The Detection of Counterfeit Banknotes Using Ensemble Learning Techniques of AdaBoost and Voting,” International Journal of Intelligent Engineering and Systems, vol. 14, no. 1, pp. 326–339, Feb. 2021, doi: 10.22266/ijies2021.0228.31.

[74] Ke, Guolin, et al. "Lightgbm: A highly efficient gradient boosting decision tree." 2017 Advances in neural information processing systems 30.

[75] Y. Hua, “An Efficient Traffic Classification Scheme Using Embedded Feature Selection and LightGBM,” 2020 Information Communication Technologies Conference (ICTC), May 2020, doi: 10.1109/ictc49638.2020.9123302.

[76] S. Yue, P. Li, and P. Hao, “SVM classification: Its contents and challenges,” Applied Mathematics-A Journal of Chinese Universities, vol. 18, no. 3, pp. 332–342, Sep. 2003, doi: 10.1007/s11766-003-00595.

[77] “Linear Regression and the Logistic Regression Model,” Applied Logistic Regression Analysis, pp. 2–17, 2002, doi: 10.4135/9781412983433.n1.

[78] M. M. Saritas and A. Yasar, “Performance Analysis of ANN and Naive Bayes Classification Algorithm for Data Classification,” International Journal of Intelligent Systems and Applications in Engineering, vol. 7, no. 2, pp. 88–91, Jan. 2019, doi: 10.18201/ijisae.2019252786.

[79] H. Liu and H. Motoda, “Feature Selection Methods,” Feature Selection for Knowledge Discovery and Data Mining, pp. 73–95, 1998, doi: 10.1007/978-1-4615-5689-3_4.

[80] A. S. Tomar, M. Singh, G. Sharma, and K. V. Arya, “Traffic Management using Logistic Regression with Fuzzy Logic,” Procedia Computer Science, vol. 132, pp. 451–460, 2018, doi: 10.1016/j.procs.2018.05.159.

[81] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, “A comparative analysis of gradient boosting algorithms,” Artificial Intelligence Review, vol. 54, no. 3, pp. 1937–1967, Aug. 2020, doi: 10.1007/s10462-020-09896-5.

[82] D. Patil and T. Pattewar, “Majority Voting and Feature Selection Based Network Intrusion Detection System,” ICST Transactions on Scalable Information Systems, p. 173780, Jul. 2018, doi: 10.4108/eai.4-4-2022.173780.

[83] R. Polikar, “Ensemble based systems in decision making,” IEEE Circuits and Systems Magazine, vol. 6, no. 3, pp. 21–45, 2006, doi: 10.1109/mcas.2006.1688199.

[84] R. Swami, M. Dave, and V. Ranga, “Voting-based intrusion detection framework for securing software-defined networks,” Concurrency and Computation: Practice and Experience, vol. 32, no. 24, Jul. 2020, doi: 10.1002/cpe.5927.

[85] Y. N. Kunang, S. Nurmaini, D. Stiawan, and B. Y. Suprapto, “Attack classification of an intrusion detection system using deep learning and hyperparameter optimization,” Journal of Information Security and Applications, vol. 58, p. 102804, May 2021, doi: 10.1016/j.jisa.2021.102804.

[86] K. M. Hamdia, X. Zhuang, and T. Rabczuk, “An efficient optimization approach for designing machine learning models based on genetic algorithm,” Neural Computing and Applications, vol. 33, no. 6, pp. 1923–1933, Jun. 2020, doi: 10.1007/s00521-020-05035-x.

[87] Z. Wang, J. Liu, and L. Sun, “EFS-DNN: An Ensemble Feature Selection-Based Deep Learning Approach to Network Intrusion Detection System,” Security and Communication Networks, vol. 2022, pp. 1–14, Apr. 2022, doi: 10.1155/2022/2693948.

[88] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, “On the Properties of Neural Machine Translation: Encoder–Decoder Approaches,” Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, 2014, doi: 10.3115/v1/w14-4012.

[89] J. Kim, J. Kim, H. L. Thi Thu, and H. Kim, “Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection,” 2016 International Conference on Platform Technology and Service (PlatCon), Feb. 2016, doi: 10.1109/platcon.2016.7456805.

[90] A. Aldweesh, A. Derhab, and A. Z. Emam, “Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues,” Knowledge-Based Systems, vol. 189, p. 105124, Feb. 2020, doi: 10.1016/j.knosys.2019.105124.

[91] G. Al Naymat, M. Al Kasassbeh, and E. Al Harwari, “Using machine learning methods for detecting network anomalies within SNMP-MIB dataset,” International Journal of Wireless and Mobile Computing, vol. 15, no. 1, p. 67, 2018, doi: 10.1504/ijwmc. 2018.10015860.

[92] A. Manna and M. Alkasassbeh, “Detecting network anomalies using machine learning and SNMP-MIB dataset with IP group,” 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS), Oct. 2019, doi: 10.1109/ictcs.2019.8923043.

[93] S. SMADI, “Application Layer Denial of Services Attack Detection Based on StackNet,” International Journal of Advanced Trends in Computer Science and Engineering, vol. 9, no. 3, pp. 3929–3936, Jun. 2020, doi: 10.30534/ijatcse/2020/215932020.

[94] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, “Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy,” 2019 International Carnahan Conference on Security Technology (ICCST), Oct. 2019, doi: 10.1109/ccst.2019.8888419.

[95] D. K. Gharkan and A. A. Abdulrahman, “Construct an efficient distributed denial of service attack detection system based on data mining techniques,” Indonesian Journal of Electrical Engineering and

Computer Science, vol. 29, no. 1, p. 591, Jan. 2022, doi: 10.11591/ijeecs.v29.i1.pp591-597.

[96] G. Al-Naymat, M. Al-kasassbeh, and E. Al-Hawari, “Exploiting SNMP-MIB Data to Detect Network Anomalies Using Machine Learning Techniques,” Intelligent Systems and Applications, pp. 991–1004, Nov. 2018, doi: 10.1007/978-3-030-01057-7_73.

[97] M. Alkasassbeh, “A Novel Hybrid Method for Network Anomaly Detection Based on Traffic Prediction and Change Point Detection,” Journal of Computer Science, vol. 14, no. 2, pp. 153–162, Feb. 2018, doi: 10.3844/jcssp.2018.153.162.

[98] Y. Wei, J. Jang-Jaccard, F. Sabrina, A. Singh, W. Xu, and S. Camtepe, “AE-MLP: A Hybrid Deep Learning Approach for DDoS Detection and Classification,” IEEE Access, vol. 9, pp. 146810–146821, 2021, doi: 10.1109/access.2021.3123791.

[99] M. V. O. de Assis, L. F. Carvalho, J. J. P. C. Rodrigues, J. Lloret, and M. L. Proença Jr, “Near real-time security system applied to SDN environments in IoT networks using convolutional neural network,” Computers & Electrical Engineering, vol. 86, p. 106738, Sep. 2020, doi: 10.1016/j.compeleceng.2020.106738.

[100] O. E. Tayfour and M. N. Marsono, “Collaborative detection and mitigation of DDoS in software-defined networks,” The Journal of Supercomputing, vol. 77, no. 11, pp. 13166–13190, Apr. 2021, doi: 10.1007/s11227-021-03782-9.

[101] Y. Wei, J. Jang-Jaccard, F. Sabrina, A. Singh, W. Xu, and S. Camtepe, “AE-MLP: A Hybrid Deep Learning Approach for DDoS Detection and Classification,” IEEE Access, vol. 9, pp. 146810–146821, 2021, doi: 10.1109/access.2021.3123791.

[102] M. Gohil and S. Kumar, “Evaluation of Classification algorithms for Distributed Denial of Service Attack Detection,” 2020 IEEE Third

International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), Dec. 2020, doi: 10.1109/aike48582.2020.00028.

الملخص

تجاوزت الشبكات المعرفة بالبرمجيات (SDN) مشاكل الشبكات التقليدية من خلال فصل مستوى التحكم عن مستوى البيانات. حيث سمح هذا الفصل بنقل ادارة الشبكة إلى وحدة تعرف باسم وحدة التحكم، والتي تكون مسؤولة عن ادارة الأجهزة ضمن مستوى البيانات بشكل فعال. ولكن، أصبحت معمارية SDN هدفًا جذابًا للعديد من الهجمات التي يمكن استغلالها من قبل المتسللين. على سبيل المثال، هجمات حجب الخدمة (DoS) أو هجمات حجب الخدمة الموزعة (DDoS) تعد واحدة من أكثر الهجمات تدميراً التي يمكنها مهاجمة وظائف شبكة SDN وجعل معظم خدمات الشبكة غير متاحة للعملاء المعتمدين.

وبالتالي، فإن الهدف من هذه الأطروحة هو بناء نماذج لكشف التسلل (الهجمات) بناءً على أساليب التعلم الآلي المحسنة لتأمين بيئة SDN ضد هجمات DoS/DDoS بالإضافة إلى أنواع أخرى من الاختراقات. حيث تم تعزيز كفاءة وفعالية حلول الكشف من خلال اقتراح أربعة نماذج محسنة: نموذج تقليل الخصائص الهجين (HFR)، والنموذج المحسن لخوارزمية CatBoost هو (OCBM)، ونموذج الشبكة العصبية العميقة المضبوطة بدقة هو (FTDNN)، ونموذج القائم على اساس دمج مصنفات مختلفة هو (DSBHED).

استخدم نموذج HFR جميع أساليب تقليل الخصائص (filter-based, wrapper-based and embedded-based) لتحديد أفضل مجموعة فرعية من الميزات، مما ادى إلى تحسين سرعة وأداء النماذج. في البداية، يتم تقييم الميزات باستخدام ثلاثة تقنيات (MI,WOA, and ETC). وبعد ذلك، تقوم تقنية التصويت بالأغلبية (MVT) بتصفية الميزات غير المهمة وغير المتسقة، مما يؤدي إلى إنشاء مجموعة بيانات تضم عدد اقل من الميزات. وبالاخير يتم حساب قيمة (Mean value) لتوحيد أوزان الميزات التي تم الحصول عليها من MVT، مما يؤدي إلى مجموعة فرعية تضم فقط الميزات الاكثر أهمية.

تم اقتراح النموذج القائم على تقنية ML، OCBM، والنموذج القائم على تقنية DNN ، FTDNN، لتحسين التنبؤ الثنائي لهجمات DoS/DDoS مع الحفاظ على معدل منخفض لقيمة (FPR) وتقليل وقت التدريب والاختبار. للتمييز بدقة بين (normal traffic) وهجمات DoS/DDoS، ذلك يتطلب تحسين المعلمات الفائقة لمصنف OCBM. ولتحقيق ذلك، تم استخدام تقنية تحسين (GSCV) للعثور على أفضل القيم لهذه المعلمات.

من ناحية أخرى، في حالة النموذج القائم على تقنية DNN ، تعد طريقة (RSCV) أداة قيمة لتحديد البنية المثالية لشبكة DNN ومعلماتها من خلال استكشاف قيم المعلمات بكفاءة. علاوة على ذلك، على عكس العديد من الدراسات الأخرى، تم إجراء العديد من الحالات التجريبية لتقييم أداء كل من نماذج OCBM وFTDNN ضد هجمات غير المعروفة.

بالإضافة إلى ذلك، للكشف بدقة عن الهجمات مختلفة مثل port scan وSYN، وUDP، وغيرها، تم اقتراح نموذج DSBHED. حيث يعتمد هذا النموذج على مجموعة من المصنفات المتعددة المستندة إلى التعلم الآلي للتنبؤ بالنتائج النهائية. تتضمن مكونات نموذج DSBHED على المُصنفات المحسنة التالية (ORF) و(KNN) ومصنف (OLightGBM). تم تحسين المعلمات الفائقة لـتلك المصنفات باستخدام تقنية GSCV لتحقيق الأداء الأمثل.

تم تقييم كفاءة وفعالية النماذج المقترحة من خلال مقاييس تقييم الاداء التالية (detection rate, accuracy, precision, F1-score, and FPR بالاضافة الى لك تم حساب وقت التدريب والاختبار لجميع النماذج المقترحة) وذلك باستخدام مجموعات بيانات InSDN، وCICDDoS-2019، وSNMP، وCICIDS-2017، حيث اشارت النتائج التجريبية إلى أن النماذج المقترحة حققت أداء أفضل في الكشف عن الهجمات المختلفة طبقا الى جميع مقاييس التقييم، مقارنة بالدراسات السابقة ومصنفات ML الاخرى والتي تم تجربتها على نفس مجموعات البيانات المذكورة انفا. وبالتالي، فإن النماذج المقترحة توفر ثقة كبيرة في تأمين شبكات SDN.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية تكنولوجيا المعلومات
قسم البرمجيات

نظام كشف التسلل المطور بالاعتماد على تقليل الخصائص الهجين و تقنيات التعلم الالي للشبكات المعرفة برمجيا

أطروحة مقدمة إلى

مجلس كلية تكنولوجيا المعلومات جامعة بابل عن استيفاء جزئي لمتطلبات درجة دكتوراه في تكنولوجيا المعلومات / البرمجيات

من قبل

حسنين علي عبيس عبود

بأشراف

أ.د.وسام سمير بهية

1445 هـ

2023 م

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية تكنولوجيا المعلومات
قسم البرمجيات

نظام كشف التسلل المطور بالاعتماد على تقليل الخصائص الهجين و تقنيات التعلم الالي للشبكات المعرفة برمجيا

أطروحة مقدمة إلى

مجلس كلية تكنولوجيا المعلومات جامعة بابل كجزء من متطلبات نيل درجة الدكتوراه فلسفة في تكنولوجيا المعلومات / برمجيات

من قبل

حسنين علي عبيس عبود

بأشراف

أ.د.وسام سمير بهية

1445 هـ

2023 م