PCA-ANN:Feature Selection Based Hybrid Intrusion Detection System in Software Defined Network

By

Sabila Nawshin (ID: 012203023)

Department of Computer Science and Engineering

United International University

Submitted in partial fulfilment of the requirements of the degree of

MSc. in Computer Science & Engineering

July 2023

Declaration

I, Sabila Nawshin declare that this thesis titled **PCA-ANN:Feature Selection Based Hybrid Intrusion Detection System in Software Defined Network** and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a MsCSE degree at United International University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at United International University or any other institution, this has been clearly stated. Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given.This thesis is entirely my own work. I have acknowledged all main sources of help and where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

(Sabila Nawshin, ID:012203023)

Certificate

I do hereby declare that the research works embodied in this thesis s entitled **PCA-ANN:Feature Selection Based Hybrid Intrusion Detection System in Software Defined Network** is the outcome of an original work carried out by Sabila Nawshin under my supervision. I further certify that the dissertation meets the requirements and the standard for the degree of MSc in Computer Science and Engineering.

Signed: _____

Date: _____

(Salekul Islam, PhD)

Professor, Department of Computer Science and Engineering

Department of Computer Science and Engineering,

United International University,

Dhaka-1209, Bangladesh.

Approval Certificate

This thesis titled **PCA-ANN:Feature Selection Based Hybrid Intrusion Detection System in Software Defined Network** submitted by Sabila Nawshin, Student ID: 012203023, has been accepted as satisfactory in fulfillment of the requirement for the degree of Master of Science in Computer Science and Engineering on July, 2023.

Head Examiner:

Signed: _____

Mohammad Shahriar Rahman, PhD
Professor, Department of Computer Science and Engineering (CSE)
United International University (UIU), Dhaka, Bangladesh

Examiner 1:

Signed: _____

Mohammad Mamun Elahi
Assistant Professor, Department of Computer Science and Engineering (CSE)
United International University (UIU), Dhaka, Bangladesh

Examiner 2:

Signed: _____

Dr. Md. Motaharul Islam
Professor & Director-MSCSE, Department of Computer Science and Engineering (CSE)
United International University (UIU), Dhaka, Bangladesh

Ex-Officio:

Signed: _____

Dr. Mohammad Nurul Huda
Professor & Head of the Department of Computer Science and Engineering
United International University Dhaka, Bangladesh

Abstract

The increasing complexity of modern networks and the rise of sophisticated cyber attacks has made the development of effective Intrusion Detection Systems (IDS) a critical need. Software Defined Networking (SDN) technology provides us with a programmable central controller, providing a central view of the whole network as opposed to the existing internet structure where each of the routers only has information about it's surrounding routers, which results in the systems and algorithms developed in it to operate in an distributed setting. The centralized view provided by SDN makes it an attractive platform for IDS deployment. The networks under SDN is, however, more vulnerable to malicious activities or attacks than the traditional network topology due to the same centralised nature. The recently published "inSDN" dataset was prepared specifically for intrusion detection in SDN. In this study, we have used this dataset to introduce a novel Intrusion Detection System (IDS) model that integrates Principal Component Analysis (PCA) - a feature selection methodology commonly used in traditional Machine Learning (ML) to extract the principal features from large datasets and reduce dimensionality - and Artificial Neural Networks (ANN) to classify network traffic based on the extracted features. The model achieved an accuracy of 99.95% for multi-class classification. The results show that the proposed model outperforms the current state-of-the-art techniques in a much simpler settings and reduces the need for complex models that require extensive computation in the "inSDN" attack dataset.

Acknowledgements

All praise is due to Allah, it is by his grace that I was able to finish this work.

This work would have not been possible without the input and support of many people over the last two years. I would like to express my gratitude to everyone who contributed to it.

I would like to thank my supervisor, Dr. Salekul Islam, Professor, United International University for providing his continuous support and guidance. His guidance helped me get back to work after repeated study gaps. His encouragement has been pivotal in the completion of this work. My gratitude also goes to Dr. Mohammad Shahriar Rahman, Mohammad Mamum Elahi and Dr. Md. Motaharul Islam, members of the examination committee, for the valuable feedback and suggestions they provided on this work.

Lastly, my heartfelt gratitude goes to my family, especially my parents for never forgetting about this work and pushing me forward. This work would not have been completed without their continuous support and prayers.

Contents

List of Figures **ix**

List of Tables **x**

1 Introduction **1**

1.1 Motivation 1

1.2 Problem Statement 2

1.3 Objective 3

1.4 Thesis Contributions 3

1.5 Organization of the Thesis 4

2 Background Study **5**

2.1 Software-Defined Networking (SDN) 5

2.2 Attack vulnerability in SDN 6

2.3 Intrusion Detection Systems (IDS) 9

2.3.1 ML/DL based IDS in SDN 10

3 Related Work **13**

3.1 Literature on Deep Learning Based Methodologies 16

3.1.1 Overview & Analysis 16

3.1.2 Detailed Review 19

3.2 Literature on Shallow ML Methodologies 21

3.2.1 Overview & Analysis 21

3.2.2 Detailed Review 23

3.3 Gap Analysis 25

4 Proposed Method **27**
4.1 Dataset 28
4.1.1 Dataset Exploration 29
4.2 Data Pre-Processing 32
4.3 Feature Selection 33
4.3.1 Why PCA? 33
4.3.2 Implementation 33
4.4 Classification Model 35

5 Experimental Results & Discussion **40**
5.1 Evaluation Metrics 40
5.2 Evaluation Environment 40
5.3 Binary Classification Model 41
5.4 Multi-class Classification Model 42
5.5 Discussion 45

6 Conclusion & Future Work **46**
6.1 Conclusion 46
6.2 Limitation 46
6.3 Future Work 47

Bibliography **48**

List of Figures

2.1 Separating Data plane and Control plane with Software Defined Networking . . . 6
2.2 Different attack points in SDN . . . 7
2.3 The different categories of IDS . . . 9
2.4 Overview of IDS Deployment in SDN . . . 10

4.1 Workflow of the proposed model . . . 27
4.2 Distribution of classes in the inSDN Dataset . . . 29
4.3 Observing the Dataset . . . 30
4.4 Data Pre-processing steps . . . 32
4.5 Steps of Principal Component Analysis . . . 34
4.6 Neural Network Model . . . 35
4.7 Steps in defining and training the neural network classifier model using python . . . 36

5.1 Loss graph for Binary classifier . . . 41
5.2 Confusion Matrix for Binary Classification Model . . . 41
5.3 Confusion Matrix for Multi-class Classification Model . . . 43
5.4 Loss graph for Multi-class classifier . . . 44

List of Tables

3.1 Brief overview of selected works 13
3.2 Analyzing the selected works 17
3.3 Analyzing the selected works 22

5.1 Experimental Results (Binary Classification Model) 42
5.2 Experimental Results (Multi-Class Classification Model) 44

Chapter 1

Introduction

1.1 Motivation

The Internet has become an indispensable part of our daily lives and is rapidly expanding, leading to an unsustainable burden on the existing network infrastructure. To address this challenge, Software Defined Networking (SDN) [29] provides a dynamic, programmatically efficient, and adaptable network configuration through its functional isolation and centralized programmability. Intelligent networking devices such as routers, firewalls, and load balancers operate on the control plane and communicate with neighboring devices to extract forwarding information and build a forwarding table. However, with the increasing load on the internet, managing thousands of devices in a network becomes increasingly difficult. SDN solves this issue by separating the data plane from the control plane and centralizing control to manage these devices more efficiently.

Intrusion Detection Systems (IDSs) have been developed to detect and notify network administrators of malicious attacks, as the internet is subjected to various malicious activities [22]. Traditional networks have seen notable works in detecting and protecting against malicious activities, and Software-Defined Networking (SDN) is no exception. Various machine learning (ML) techniques have been applied to IDSs in both traditional networks and SDN, including Decision Trees, Support Vector Machines (SVMs), and Random Forests. Decision Trees construct models of decisions and their consequences, while SVMs aim to classify data into classes using hyperplanes. Random Forests generate multiple decision trees and combine their results to improve

model accuracy. Combining these techniques can enhance the overall performance of an IDS system.

Deep Learning (DL) [20] is a subset of ML that utilizes Artificial Neural Networks (ANNs) to solve complex engineering problems. Researchers have explored DL for IDS systems due to its ability to automatically learn features and detect patterns in data. However, DL methods require large datasets and significant computational power to train models, which can be costly in the central controller. The inSDN dataset [17] has opened up new avenues for researchers working on IDS in SDN due to its impressive number of features and instances, and can be considered a big data. It provides a comprehensive SDN dataset for training and evaluating the performance of complex intrusion detection systems. With the availability of such datasets, the application of advanced ML techniques like DL can be applied in SDN for Intrusion detection systems to provide promising solutions to enhance network security.

Principal Component Analysis (PCA) [24] is a statistical method that reduces the dimensionality of large datasets while preserving most of the variance in the data. This technique helps to discover patterns and relationships in high-dimensional data by transforming it into a lower-dimensional space. It identifies the most important features, or principal components, and combines them to create a new, smaller set of features that retains most of the variance in the original dataset. This helps to improve model performance and reduce overfitting.

1.2 Problem Statement

With its dynamic and spontaneous features, Software Defined Network (SDN) can minimize the workload of individual routers, leading to overall faster performance. Implementing complex algorithms for Intrusion Detection also becomes much easier using the centralized view provided by the controllers and the programmable functionalities of SDN. The controller contains a considerable amount of computational power that the individual routers did not possess, making it possible to implement complex Deep Learning (DL) models for various activities. Therefore, many of the existing literature are leaning towards DL models with the larger datasets.

In this study, a novel IDS model that integrates Principal Component Analysis (PCA) and Artificial Neural Networks (ANN) is introduced to classify network traffic

based on the extracted features. The goal of this research is to develop an accurate and efficient IDS that can effectively detect intrusions in SDN network using the "inSDN" dataset while reducing the need for complex models and extensive computation.

1.3 Objective

The objective of this study is to apply PCA to reduce the dimensionality of the large "inSDN" [17] dataset and debate the need of more complex DL models in the dataset. PCA will generate a smaller set of features that retained most of the original dataset's variance and we will train Neural Network (NN) models using that processed dataset for binary-class classification and multi-class attack classification and evaluate our results based on the previous works in the literature.

1.4 Thesis Contributions

The main contribution of this study are,

- We have extensively explored a number of previous literature and identified a bias towards more complex models for larger datasets in recent literature.
- We have used the large "inSDN" dataset where most previous works have leaned towards complex DL models and applied a much simpler feature selection based model which has not been used before.
- We have proposed a novel IDS model that integrates PCA with NN model for Intrusion detection and reduces the burden on the NN model to allow it to provide itself more computational power to train itself efficiently while lowering complexity. No other work has been done in the literature utilizing this method in this dataset to the best of our knowledge.
- We have identified and extracted 31 predominant features from the dataset with 84 features. We have utilized different programming languages (R, Python) in the different stages of the study to analyze and extract the features from the dataset and to build and train the Neural Network Models.

- We have provided a comprehensive exploration of the dataset and identified the redundant attributes. The attribute selections in literature are all done in an implicit nature where the models extract the useful attributes in the training step. None of the previous work have explicitly identified them in this dataset before to the best of our knowledge.
- We have shown competitive analysis of results for both multi-class attack classification (BFA, BOTNET, DDoS, DoS, Probe, U2R, Web Attack and Normal traffic) and binary class attack classification (Malicious and Normal traffic).
- We have achieved state-of-the art results in the dataset with our proposed model and have discussed the possible reasons for the computationally heavier model's reduced performance.

1.5 Organization of the Thesis

The rest of the study is organised as follows:

Chapter 2 provides background knowledge related to this work

Chapter 3 explores the previous literature and related works.

Chapter 4 explains the proposed methodology.

Chapter 5 discusses the experimental results and evaluates the proposed model.

Chapter 6 concludes the study with discussion on future directions.

Chapter 2

Background Study

A comprehensive overview of the existing knowledge to serve as the foundation for the study is presented in this chapter. A number of promising works can be found for Intrusion detection in SDN, some of the most promising works that utilized various methodologies have been reviewed in the next chapter. Before going into the literature, the background knowledge needed to understand this work is discussed here.

2.1 Software-Defined Networking (SDN)

Software-Defined Networking (SDN) [29] represents an architectural paradigm in networking decouples the network control plane from the forwarding plane (Data Plane). Unlike traditional networking, where the control plane responsible for routing and forwarding decisions resides within network devices, SDN separates this functionality and relocates it to a centralized software-based controller, while retaining the forwarding plane within the network devices. As each network device is not required to be configured individually, this approach enables networks to be more flexible, scalable, and cost-effective, as well as easier to manage.

Within an SDN architecture, the network devices, commonly referred to as switches or forwarding elements, undertake the task of packet forwarding based on instructions received from the centralized controller. Serving as the network's intelligence hub, the controller offers a comprehensive view of the network and grants programmable control over network behavior. To establish communication and define forwarding rules and policies, the controller interacts with the switches through standardized protocols.

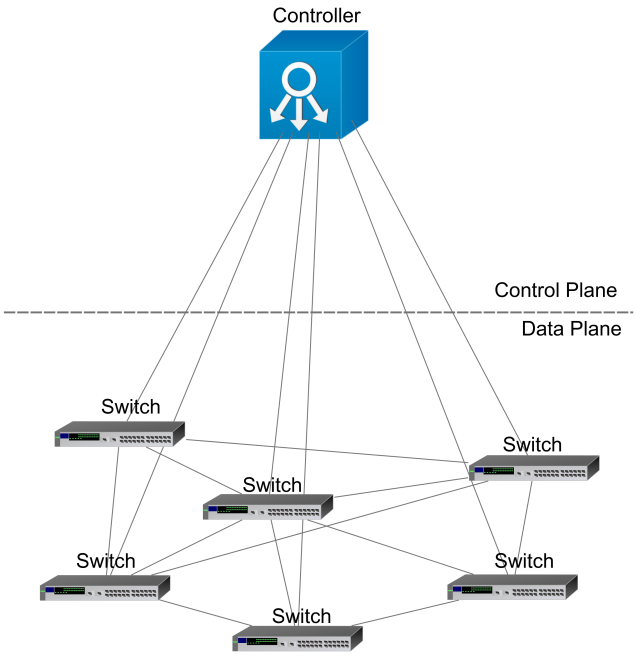

Figure 2.1: Separating Data plane and Control plane with Software Defined Networking

The centralization of the control plane within SDN empowers network administrators to dynamically manage and configure the network through software, eliminating the dependence on individual device configurations. This programmability provides avenues for efficient network management, swift deployment of new services, and streamlined integration of network functions and applications. It also facilitates the creation of connection-oriented technologies and routing frameworks. Acting as a network operating system, SDN functions as the controller that manages the data plane from the control plane, thereby providing network administrators with a flexible and scalable approach to network management. The programmability of the network enables the implementation of complex and computationally heavy algorithms, while also promoting dynamic network structures.

2.2 Attack vulnerability in SDN

The centralized architecture of Software-Defined Networking (SDN) makes it more susceptible to a wide range of attack types and threats compared to traditional internet

structures [5]. Unlike conventional networks, SDN exposes all layers to different types of attacks, some of which are specific to SDN. While attacks on the application layer and data plane variables are common in both traditional and SDN networks, SDN faces unique challenges such as attacks on the SDN controller and communication channels between control and data plane devices. It is crucial to acknowledge that compromising a single machine or application in an SDN environment can lead to catastrophic consequences, potentially compromising the entire network [14]. Kreutz et al. [19] identified several attack variables that could jeopardize the SDN architecture, while [4, 8, 10, 21, 26, 33] highlighted the security concerns in OpenFlow. Figure 2.2 outlines the different attack vectors and their targets in SDN.

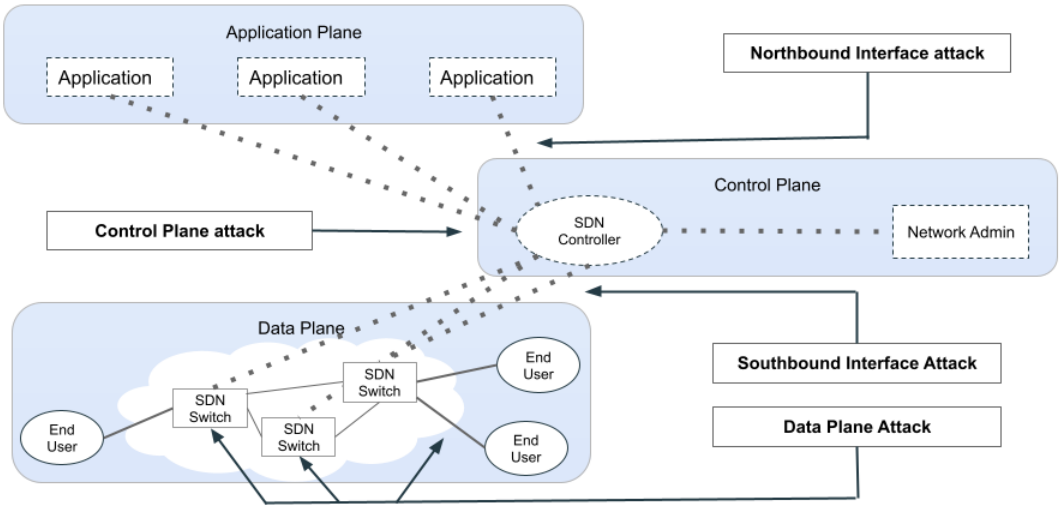

Figure 2.2: Different attack points in SDN

The different attacks in Figure 2.2 are briefly explained below:

- **Attacks on the data plane**: In the context of SDN, attackers can focus on the data plane network elements by exploiting vulnerabilities in the hosts within the network. This can enable the attacker to launch various attacks, such as flooding network components with attack-oriented traffic using connected switches, deploying fake switches in the SDN network to redirect traffic flow, and manipu-

lating network flow to overwhelm the controller. Such attacks can cause a significant impact on the network performance by consuming controller resources and modifying forwarding tables. In comparison, the traditional internet structure presents greater challenges for attackers to gain control over the entire network flow or compromise hardware switches to modify their forwarding tables.

- **Attacks Control plane**: SDN control plane is responsible for managing and controlling the network. An attacker can compromise the controller or the communication channel between the controller and the network devices to gain control of the network. This can lead to various attacks such as denial of service, man-in-the-middle attacks, and unauthorized access.

- **Attacks on Southbound interface**: This is one of the attacks on the communication links from the controller in SDN. The southbound interface connects the controller with the network devices. An attacker can exploit vulnerabilities in this interface to manipulate network devices or inject malicious traffic into the network. This can lead to attacks such as eavesdropping, packet dropping, and man-in-the-middle attacks.

- **Attacks on Northbound interface**: This is another one of the attacks on the communication links from the controller in SDN. The northbound interface is used by applications to interact with the SDN controller. An attacker can exploit vulnerabilities in this interface to manipulate the behavior of the controller or gain unauthorized access to sensitive data. This can lead to attacks such as data theft, denial of service, and unauthorized access.

The security concerns surrounding the OpenFlow protocol, widely used in SDN, underscore the importance of robust security measures in SDN deployments. To develop effective security strategies and countermeasures, it is imperative to gain a comprehensive understanding of the vulnerabilities specific to SDN. By identifying the attack vectors and their targets within SDN, researchers and practitioners can better assess the extent and nature of potential vulnerabilities. This knowledge enables the design and implementation of proactive security mechanisms to mitigate the risks associated with SDN deployments.

2.3 Intrusion Detection Systems (IDS)

Intrusion Detection Systems (IDS) are used to mitigate the risks posed by the vulnerabilities in network systems. An Intrusion Detection System is an essential tool for network security, designed to detect suspicious behaviors and anomalies in a network to prevent attacks. It regularly scans the network to identify abnormal behaviors and reports malicious activities to the network administration through a SIEM (Security Information and Event Management) system to reduce false alarms. While detecting suspicious network traffic is the main task of an IDS, recognizing normal traffic to dispose of false alarms is also important. IDS can be classified into five categories: Network Intrusion Detection System (NIDS), Host Intrusion Detection System (HIDS), Protocol-based Intrusion Detection System (PIDS), Application Protocol-based Intrusion Detection System (APIDS) and Hybrid Intrusion Detection System.

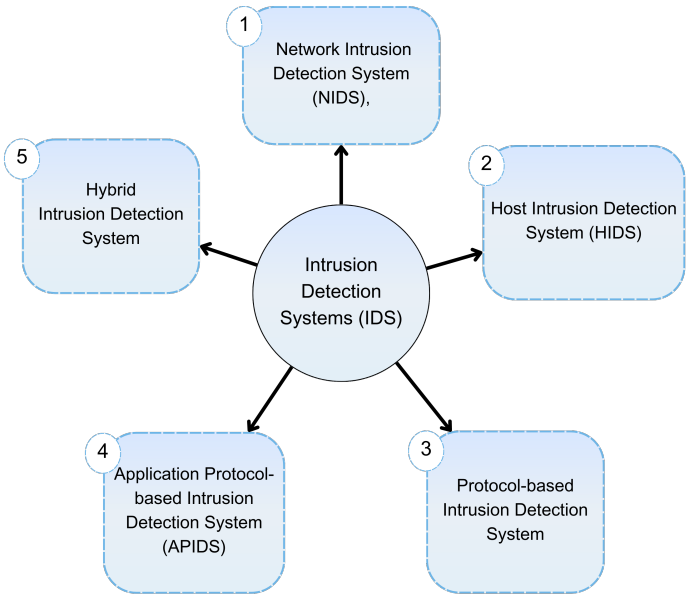

Figure 2.3: The different categories of IDS

NIDS, which monitors subnets of the network and compares the traffic flow with the known attacks to alert the administrators of breaches, can be implemented using different detection methods. The Signature-based Method detects attacks based on

specific patterns, while the Anomaly-based Method uses machine learning to create a trustworthy activity model and declares anything suspicious that does not match the model.

2.3.1 ML/DL based IDS in SDN

In the context of SDN, the deployment of IDS involves strategically placing IDS components, such as sensors, analyzers, or monitors, at strategic points within the network topology. These components can be located at switches, routers, or SDN controllers, depending on the specific requirements of the network environment. By distributing IDS components across the network, it becomes possible to capture and analyze network traffic in real-time, allowing for prompt detection and response to potential security threats [2].

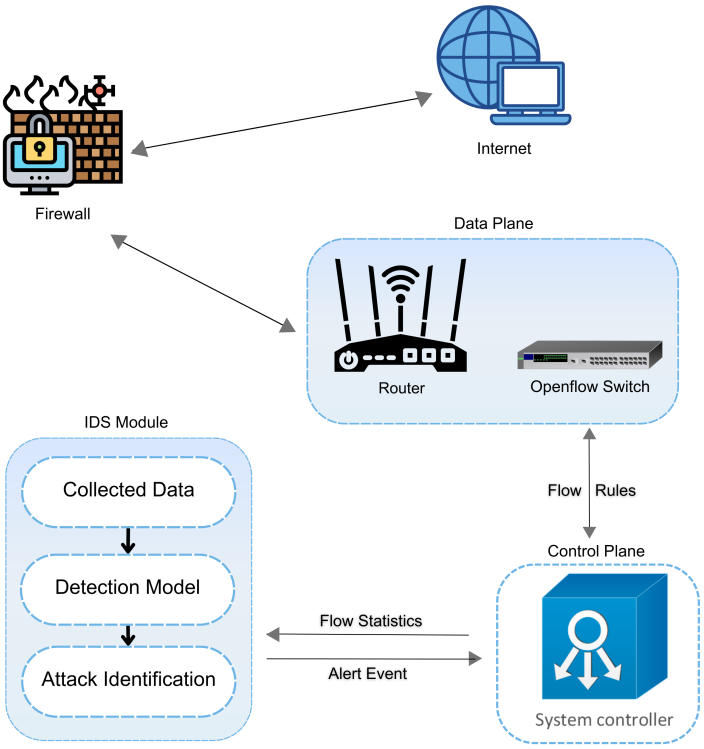

Figure 2.4: Overview of IDS Deployment in SDN

The deployment of Intrusion Detection Systems (IDS) within SDN [29] architecture

presents a promising approach to enhancing network security. SDN provides a flexible and programmable framework that enables the effective deployment and management of IDS components throughout the network infrastructure [27]. SDN's central controller acts as a single point of authority, enabling efficient coordination and communication between IDS components. This centralized control facilitates the sharing of information, such as traffic flow data, security policies, and threat intelligence, among IDS components. As a result, IDS components can collaboratively analyze network traffic and exchange insights, leading to improved accuracy and effectiveness in detecting and mitigating security incidents. The programmability of SDN allows for dynamic and adaptive security policies and rules. IDS components can leverage the programmable capabilities of SDN switches and controllers to define and enforce security policies in real-time. This enables IDS to respond quickly to changing network conditions and emerging threats. For example, when an IDS component detects suspicious activity or an attack, it can instruct SDN switches to dynamically reconfigure network flows, isolate compromised devices, or redirect traffic to specialized analysis systems for further investigation. Another benefit of deploying IDS in SDN is the ability to leverage network-wide visibility and contextual information.

SDN provides a global view of the network topology, traffic patterns, and device behaviors. IDS components can utilize this information to gain insights into the overall network state and identify potential security anomalies. By correlating network-wide data, IDS in SDN can detect sophisticated attacks that may span multiple network segments or exhibit unusual behavior patterns.

ML or DL algorithms form the core of IDS models deployed in SDN, utilizing historical network traffic data to learn patterns and behaviors, enabling the detection of both known and emerging attack vectors. This dynamic adaptation to evolving network conditions and sophisticated attack techniques empowers ML or DL based IDS models to provide proactive defense mechanisms. In SDN, ML or DL based IDS is deployed strategically within the SDN infrastructure to effectively monitor and analyze network traffic in real-time. By leveraging the programmability and centralized control of SDN, IDS models can be deployed at key points such as switches or controllers to inspect incoming and outgoing traffic. As can be observed in Figure 2.4, the IDS module can directly be in contact with the central controller which will send the diagnostic data from the router and switches to the IDS module. The IDS module in the figure

is drawn with Ml/DL models in consideration, such modules are usually made with three sections - Data Collection Module (That collects data from the controller for prediction), Ml/DL Model Module (That makes predictions), Alert Module (That alerts based on the predictions from the ML/DL model). A **ML based Network Intrusion Detection Systems (NIDS)** using the **Anomaly-based method** is developed in this work.

Chapter 3

Related Work

In this chapter, a synthesis of relevant literature and prior studies is presented to establish the context and significance of the research problem. A number of promising works can be found for Intrusion detection in SDN, some of the most promising works that utilized various Machine Learning and Deep Learning methodologies have been reviewed in this section. The works reviewed were mainly divided into two sections, Deep learning based works and shwallower Machine Learning Based works. Table 3.1 provides an brief overview of the reviewed papers.

Table 3.1: Brief overview of selected works

Title	**Author' Name with Year**	**Core Methodologies**
Deep Learning Based Intrusion Detection Systems		
Deep learning approach for Network Intrusion Detection in Software Defined Networking	Tang et al. 2016 [36]	DNN Model
A Deep Learning Based DDoS Detection System in Software Defined Networking (SDN)	Niyaz et al. 2017 [28]	Auto-Encoder based Multi-Vector model

Table 3.1 – Continued

Title	**Author' Name with Year**	**Core Methodologies**
Detection and defense of DDoS attack–based on deep learning in OpenFlow-based SDN	Chuanhuang et al. 2017 [12]	Recurrent Neural Network (RNN) with Long Short Term Memory (LSTM) and Convoluted Neural Network (CNN)
Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks	Tang et al. 2018 [35]	Gated recurrent units(GRU), Recurrent Neural Network (RNN)
Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in SDN: A social multimedia perspective	Garg et al. 2019 [18]	Restricted Boltzmann Machine (RBM) and Gradient Descent based Support Vector Machine (SVM)
A Botnet Detection Method on SDN using Deep Learning	Maeda et al. 2019 [25]	Deep Lerning Based Model
DeepIDS: Deep learning approach for intrusion detection in software defined networking	Marwan 2020 [37]	RNN-based model utilizing a novel regularization method (Named RNN-SDR)
A Hybrid CNN-LSTM Based Approach for Anomaly Detection Systems in SDNs	Abdallah et at. 2021 [1]	Convolutional Neural Network (CNN), Long Short-Term Memory Network (LSTM), L2 Regularization and dropout method

Table 3.1 – Continued

Title	**Author' Name with Year**	**Core Methodologies**
A Flow-Based Anomaly Detection Approach With Feature Selection Method Against DDoS Attacks in SDNs	El sayed et al. 2022 [31]	Long Short Term Memory (LSTM), Autoencoder, Information Gain (IG), Random Forest (RF)
A GAN-Based Intrusion Detection Model for 5G Enabled Future Metaverse	Ding et al. 2022 [15]	Generative Adversarial Network (GAN), Deep Auto-Encoders (DAE), and Random Forest (RF) algorithms
Intrusion Detection Systems based on other ML methodologies		
Machine learning based malicious payload identification in software-defined networking	Cheng et al. 2021 [9]	Packet-driven sampling method, linear prediction techniques
QoS-aware Traffic Classification Architecture Using Machine Learning and Deep Packet Inspection in SDNs	Yu et al. 2018 [38]	Tri-Training mechanism, support vector machines (SVM), k-nearest neighbors (KNN), and Naive Bayes (NB)
An extended SDN architecture for network function virtualization with a case study on intrusion prevention	Lin et al. 2015 [23]	Two-tier mechanism for traffic flow classification, utilizing data plane classification and DPI to extracting attack characteristics
Machine Learning-Based Detection of Ransomware Using SDN	Cusack et al. 2018 [13]	Programmable forwarding engines (PFE) to identify flow characteristics

Table 3.1 – Continued

Title	Author' Name with Year	Core Methodologies
Machine-Learning Based Threat-Aware System in Software Defined Networks	Song et al. 2017 [34]	Decision Tree (DF) and Random Forest (RF) algorithms
Usage of Machine Learning Algorithms for Flow Based Anomaly Detection System in Software Defined Networks	Fatih et al. 2021 [3]	K Nearest Neighbor, Decision Tree and Support Vector Machine
Support Vector Machine Meets Software Defined Networking in IDS Domain	Boero et al. 2017 [6]	Entropy-based Information Gain (IG) method.
MitM detection and defense mechanism CBNA-RF based on machine learning for large-scale SDN context	Sebbar et al. 2020 [32]	Random Forest (RF) based method

3.1 Literature on Deep Learning Based Methodologies

We have reviewed some of the recent prominent works on Intrusion Detection Systems in Software Defined Networking that utilizes Deep learning methodologies in this section. Various studies have explored ML and DL techniques for IDS in SDN.

3.1.1 Overview & Analysis

Table 3.2 provides detailed analysis of the reviewed papers. We have pointed out each works focus point, discussed their contribution to the sector and some of the challenges they may face. Tang et al. [36] achieved 75.5% accuracy using a DNN model on flow-based anomaly detection. Garg et al. [18] proposed a hybrid DL model combining RBM and SVM for detecting malicious activities. Niyaz et al. [28] utilized stacked

auto-encoders for DDoS detection with reduced False Alarm Rate (FAR). Abdallah et al. [1] achieved 96.32% accuracy by combining CNN and LSTM. Ding et al. [15] proposed a hybrid IDS model using GAN, DAE, and RF algorithms, achieving 99.8% and 99.6% accuracy in binary and multi-class classification, respectively. These studies demonstrate the potential of DL models in detecting various attacks in SDN.

Table 3.2: Analyzing the selected works

Author' Name with Year	**Focal Point**	**Contributions**
Tang et al. [36]	Developing a flow-based anomaly detection method	Their method achieved 75.5% accuracy in detecting flow traffic-based anomalies.
Garg et al. [18]	Developed a hybrid two-module DL model for IDS in SDN	Utilizes an enhanced Restricted Boltzmann Machine (RBM) and Gradient Descent based Support Vector Machine (SVM) to detect malicious activities. The model selects features in one module and detects anomalies in the other.
Niyaz et al. [28]	Developing a stacked auto-encoder-based multi-vector DDoS detection method that predicts whether received traffic is normal or attack-oriented.	Selected useful feature attributes from network headers to decrease the False Alarm Rate (FAR) and increase recognition efficiency.
Tang et al. [35]	Improving their previous model	Used a GRU-RNN facilitated Intrusion Detection System (IDS) and achieved an accuracy of 89%

Table 3.2 – Continued

Author' Name with Year	Focal Point	Contributions
Abdallah et at. [1]	Developing a hybrid Intrusion Detection System for SDN by combining Convolutional Neural Network and Long Short-Term Memory Network.	The model achieved an accuracy of 96.32% in detecting zero-day attacks on the InSDN dataset.
Maeda et al. [25]	Developed a DL-based botnet detection method that detects infected hosts and isolates them from the SDN network to prevent further spreading.	They collected data from botnet attacks on traditional internet structures and built a botnet attack detection model in SDN based on the data gathered.
El sayed et al. [31]	Focused on the security concerns of Software-Defined Networking (SDN) and the challenge of detecting Distributed Denial of Service (DDoS) attacks in SDN environments.	Proposed a Deep Learning technique based on Long Short Term Memory (LSTM) and Autoencoder to identify DDoS attacks in SDNs using feature selection methods like Information Gain (IG) and Random Forest (RF). The results demonstrated by their proposed model proves the efficiency of the DL approach in detecting DDoS attacks without significant degradation in the controller's performance.

Table 3.2 – Continued

Author' Name with Year	Focal Point	Contributions
Marwan [37]	Developing an RNN-based model utilizing a novel regularization method named RNN-SDR.	They decayed the weights to reduce complexity and increase efficiency.
Chuanhuang et al. [12]	Developed a complex model to detect DDoS attacks using RNN with LSTM and CNN.	Their model achieved satisfactory results in the ISCX dataset. These works demonstrate the potential of DL models in detecting various types of attacks in SDN.
Ding et al. [15]	Developing a hybrid intrusion detection model for Metaverse network security under 5G mobile networks.	Their model combined GAN, Deep Auto-Encoders (DAE), and RF algorithms based on IoT technology. The model achieved 99.8% and 99.6% accuracy in binary and multi-class classification experiments, respectively.

3.1.2 Detailed Review

Tang et al. [36] proposed a flow-based anomaly detection method using a simple DNN model with six basic features from the NSL-KDD dataset. Their method achieved 75.5% accuracy in detecting flow traffic-based anomalies.

Garg et al. [18] introduced a hybrid two-module DL model that utilizes an enhanced Restricted Boltzmann Machine (RBM) and Gradient Descent based Support Vector Machine (SVM) to detect malicious activities. The model selects features in one module and detects anomalies in the other.

Niyaz et al. [28] presented a stacked auto-encoder-based multi-vector DDoS detec-

tion method that predicts whether received traffic is normal or attack-oriented. They selected useful feature attributes from network headers to decrease the False Alarm Rate (FAR) and increase recognition efficiency.

Tang et al. [35] improved their previous work by using a GRU-RNN facilitated Intrusion Detection System (IDS) and achieved an accuracy of 89%.

Abdallah et at. [1] proposes a hybrid Intrusion Detection System for SDN by combining Convolutional Neural Network and Long Short-Term Memory Network. L2 Regularization and dropout method are used to overcome overfitting. The model achieves an accuracy of 96.32% in detecting zero-day attacks on the InSDN dataset. The results show that integrating CNN with LSTM improves intrusion detection performance and regularization techniques improve CNN's ability to detect new intrusions.

Maeda et al. [25] proposed a DL-based botnet detection method that detects infected hosts and isolates them from the SDN network to prevent further spreading. They collected data from botnet attacks on traditional internet structures and trained a model to detect botnet attacks in SDN.

El sayed et al. [31] focused on the security concerns of Software-Defined Networking (SDN) and the challenge of detecting Distributed Denial of Service (DDoS) attacks in SDN environments. They proposed a Deep Learning technique based on Long Short Term Memory (LSTM) and Autoencoder to identify DDoS attacks in SDNs using feature selection methods like Information Gain (IG) and Random Forest (RF). Their proposed method was evaluated on three different datasets, i.e., InSDN, CICIDS2017 and CICIDS2018, and the results demonstrated the efficiency of the DL approach in detecting DDoS attacks without significant degradation in the controller's performance.

Marwan [37] presented an RNN-based model utilizing a novel regularization method named RNN-SDR. They decayed the weights to reduce complexity and increase efficiency.

Chuanhuang et al. [12] proposed a comparatively more complex model to detect DDoS attacks using RNN with LSTM and CNN. Their model achieved satisfactory results in the ISCX dataset. These works demonstrate the potential of DL models in detecting various types of attacks in SDN.

Finally, Ding et al. [15] proposed a hybrid intrusion detection model for Metaverse network security under 5G mobile networks that achieved the best results on the inSDN dataset on literature. The model combines GAN, Deep Auto-Encoders (DAE), and RF

algorithms based on IoT technology. GAN uses gradient penalty Wasserstein distance to improve the dataset's imbalance. DAE optimizes parameters to improve efficiency, and RF constructs nodes using Gini coefficient and random sampling with replacement. The model achieves 99.8% and 99.6% accuracy in binary and multi-class classification experiments, respectively.

3.2 Literature on Shallow ML Methodologies

Some of the works using shallower machine learning methodologies are reviewed in this section. The reviewed works explores the use of machine learning (ML) in intrusion detection systems (IDS) for Software-Defined Networking (SDN). We discussed various ML-based IDS solutions in SDN, including deep packet inspection methods, traffic classification techniques, and risk awareness models. The studies investigate the effectiveness of ML algorithms such as decision trees, random forests, support vector machines, and k-nearest neighbors. These approaches address challenges like resource-performance issues, encrypted traffic analysis, and the detection of different types of attacks.

3.2.1 Overview & Analysis

The studies reviewed investigate the effectiveness of various ML algorithms. Cheng et al. [9] developed a deep packet inspection system using ML algorithms, while Yu et al. [38] employed the Tri-Training mechanism. Other works explored DPI mechanisms for attack detection, and additional studies focused on risk awareness, DDoS attack detection, malware intrusion detection, and ML-based approaches. These contributions enhance the effectiveness of IDS in SDN and address resource-performance concerns, encrypted traffic analysis, and diverse attack detection. Promising avenues include deep learning, modular architectures, and RF-based models for specific attack identification.

Table 3.3 provides detailed analysis of the reviewed papers. We have pointed out each works focus point, discussed their contribution to the sector and some of the challenges they may face.

Table 3.3: Analyzing the selected works

Author' Name with Year	**Focal Point**	**Contributions**
Cheng et al. 2021[9]	Incorporating deep packet inspection (DPI) and leveraging a range of machine learning (ML) algorithms.	Their approach optimized resource utilization while maintaining accurate detection capabilities at the packet-level granularity.
Yu et al. 2018 [38]	Developing an advanced approach for traffic classification	Offering valuable insights on using Tri-Training mechanism in utilizing DPI
Lin et al. 2015 [23]	Developing a two-tier mechanism aiming at enhancing traffic flow classification in traditional SDN architectures.	Their work proved that reliance on DPI as the sole means of extracting attack characteristics faced limitations due to the increasing use of encryption by malicious actors to evade payload analysis.
Cusack et al. 2018 [13]	Introducing a scheme leveraging programmable forwarding engines (PFE) to identify flow characteristics associated with ransomware.	Proposed scheme enabled the creation of flow records that offer per-packet information, facilitating the extraction of critical flow features for effective ransomware classification.
Song et al. 2017 [34]	Developed a risk awareness-based IDS model	Proposed model assisted in maintaining balanced flow among network routes.

Table 3.3 – Continued

Author' Name with Year	Focal Point	Contributions
Fatih et al. 2021 [3]	Investigating the effectiveness of ML algorithms.	Algorithms investigated for detecting DDoS attacks in SDN includes KNN, DT, and SVM
Boero et al. 2017 [6]	Utilizing an SVM-based IDS solution for detecting various malware intrusions	Their method selected core features based on the Entropy-based Information Gain (IG) method.
Sebbar et al. 2020 [32]	Developing an RF-based model to identify Man In The Middle (MITM) attacks	Their model selected nodes based on context and predefined policies.

3.2.2 Detailed Review

The popularity of ML techniques in network security has led to the increased application of these methods in intrusion detection systems (IDS) for Software-Defined Networking (SDN).

An innovative approach proposed by Cheng et al. [9] introduces a comprehensive solution for intrusion detection in Software-Defined Networking (SDN) by incorporating deep packet inspection (DPI) and leveraging a range of machine learning (ML) algorithms. This system encompasses the utilization of decision trees (DT), random forests (RF), multinomial Naive Bayes (NB), k-nearest neighbors (KNN), support vector machines (SVM), and RF algorithms. To ensure effective identification of malicious activities, the solution employs a primary recognition mechanism that operates at the flow level and implements IP filtering. A DPI engine is additionally integrated into the system to inspect both encrypted and unencrypted network traffic. The approach encompasses the extraction of different features for classification purposes. For unencrypted traffic, payload features based on tri-gram frequency and TF-IDF are extracted, while for encrypted traffic, TLS cipher suites are utilized as prominent features. It also

tackles the resource-performance challenge by incorporating a customizable packet-driven sampling method that relies on linear prediction techniques and optimizes resource utilization while maintaining accurate detection capabilities at the packet-level granularity.

Cabaj et al. [7], Jr and Mountrouidou [11], and Lin et al. [23] have extensively explored the application of DPI mechanisms in SDN environments for effective attack detection.

Lin et al. [23] proposed a two-tier mechanism aimed at enhancing traffic flow classification in traditional SDN architectures. Their approach involved performing initial traffic classification on the data plane, and if the initial categorization module failed to handle the traffic, the packets were forwarded to a DPI module for further analysis. However, the reliance on DPI as the sole means of extracting attack characteristics faced limitations due to the increasing use of encryption by malicious actors to evade payload analysis.

To address this challenge, Cusack et al. [13] introduced an innovative scheme that leverages programmable forwarding engines (PFE) to identify flow characteristics associated with ransomware. By harnessing the capabilities of PFEs, the proposed scheme enables the creation of flow records that offer per-packet information, facilitating the extraction of critical flow features for effective ransomware classification.

Song et al. [34] presented a risk awareness-based IDS model that employs data processing, data modeling using DT and RF algorithms, decision-making, and a response scheme to maintain balanced flow among network routes.

Yu et al. [38] also contributed to the field of intrusion detection by presenting an advanced approach for traffic classification using DPI. Their method involved the utilization of the Tri-Training mechanism, a technique that combines the three distinct classifiers: support vector machines (SVM), k-nearest neighbors (KNN), and Naive Bayes (NB). By leveraging the collective capabilities of these classifiers, they aimed to enhance the accuracy and effectiveness of traffic classification in SDN environments. The Tri-Training mechanism acts as a collaborative framework, harnessing the strengths of each classifier to achieve robust and reliable results. Through this synergistic approach, they introduced a novel solution for DPI-based traffic classification, providing valuable insights on ensemble methods.

Fatih et al. [3] investigated the effectiveness of ML algorithms, including KNN, DT, and SVM, for detecting DDoS attacks in SDN. Boero et al. [6] utilized an SVM-based IDS solution for detecting various malware intrusions, selecting core features based on the Entropy-based Information Gain (IG) method. Sebbar et al. [32] introduced an RF-based model to identify Man In The Middle (MITM) attacks by selecting nodes based on context and predefined policies.

3.3 Gap Analysis

Some of the previous literature has used different types of feature selection on the different available IDS datasets.

Tang et al.'s [36] proposed flow-based anomaly detection method used the Deep Neural Network (DNN) model to exract the useful features. They achieved 75.5% accuracy in detecting flow traffic-based anomalies. Niyaz et al.'s [28] stacked auto-encoder-based multi-vector DDoS detection method utilized feature selection to decrease the False Alarm Rate (FAR) and increase recognition efficiency. They selected useful feature attributes from network headers. On the "inSDN" dataset, El sayed et al.'s [31] proposed Deep Learning technique based on Long Short Term Memory (LSTM) and Autoencoder for DDoS attack detection in SDN utilized feature selection methods like Information Gain (IG) and Random Forest (RF) to identify relevant features for detection. Ding et al.'s [15] hybrid intrusion detection model for Metaverse network security under 5G mobile networks, which integrated GAN, Deep Auto-Encoders (DAE), and RF algorithms also extracted useful features in the hybrid model.

It can be observed from the literature review that most of the recent works done in the larger datasets are leaning towards Deep Learning methodologies. The complex models used are treated as black boxes where the useful information from the features are expected to be extracted by the DNN models without further thought into the complex computations undertaken. For example, Ding et al. [15] used Deep Auto Encoders (DAE) to optimize the parameters of the dataset in his hybrid model. It uses the same black box concept mentioned where it is expected of the model to optimize the parameters once fed into the model. Compared to DNN methods, which are typically deep and complex models that learn hierarchical representations of the data, PCA is computationally less intensive and has lower model complexity. PCA operates on the

statistical properties of the data and does not involve complex optimization procedures. We deduced that using a lightweight method like PCA before feeding the data to the Neural Network (NN) model should reduce the burden on the model and allow it to train itself efficiently while lowering complexity.

We will address this gap by explicitly extracting the principal features in the dataset using PCA before the data is fed to the NN model for training. While simple, this idea of integrating PCA with NN models has not been utilized in this dataset before to the best of our knowledge.

Chapter 4

Proposed Method

In this chapter, the systematic approach used to investigate and address the research objectives are presented. It provides a detailed description of the research design and the analysis techniques employed to obtain the results in the later section. By following a rigorous methodology, the study aims to ensure the accuracy, reliability, and reproducibility of the research outcomes. The methodology chapter serves as a roadmap for the readers, outlining the step-by-step process undertaken to conduct the research. The workflow of our proposed model is given in Figure 4.1

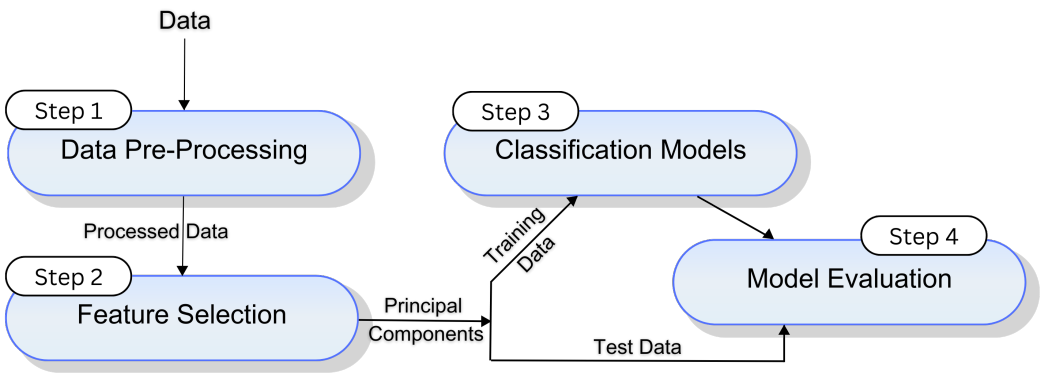

Figure 4.1: Workflow of the proposed model

The data from the dataset is processed using normalization and scaling at first. PCA has been applied on it afterwards to extract the principal features from the dataset. The processed dataset is split into training and testing segments afterwards. The training

data is used to train the classification models, and test segment is used to test the trained model. We evaluated the results using different evaluation metrics afterwards. The basic steps of the proposed model is as below -

Input: inSDN Dataset

Step 1: Data Pre-processing. Combine data into one matrix and check for missing values. Normalize and Scale Data.

Step 2: Feature Selection. Calculate Covariance matrix of the processed dataset and calculate it's Eigrnvectors and eigenvalues. Retain the columns with eigenvalues greater than threshold

Step 3: Model Definition & Training. Define classification model and training parameters. Train Model with training data and produce predictions on test data

Step 4: Evaluation. Generate the confusion matrix with the predicted labels and calculate Precision, Recall and accuracy.

4.1 Dataset

The quality of training datasets heavily affects the effectiveness of intrusion detection systems (IDSs)[22], but their deployment is challenging due to the lack of up-to-date, real-world datasets, which is mainly attributed to privacy and legal concerns. To evaluate a proposed deep learning model, we utilize the InSDN dataset [17], specifically designed to address the limitations of existing datasets in the context of software-defined networking (SDN) [29]. It contains various attack scenarios and classes, including DoS, DDoS, web attacks, password guessing, botnet, exploitation, and probe attacks, as well as normal traffic from popular application services. The attacks in the dataset originate from both internal and external networks to simulate real-world scenarios and comprise over 80 statistical features in CSV format. The dataset includes a total of

343,939 instances of both normal and attack traffic, with normal traffic accounting for 68,424 instances and attack traffic containing 275,515 instances.

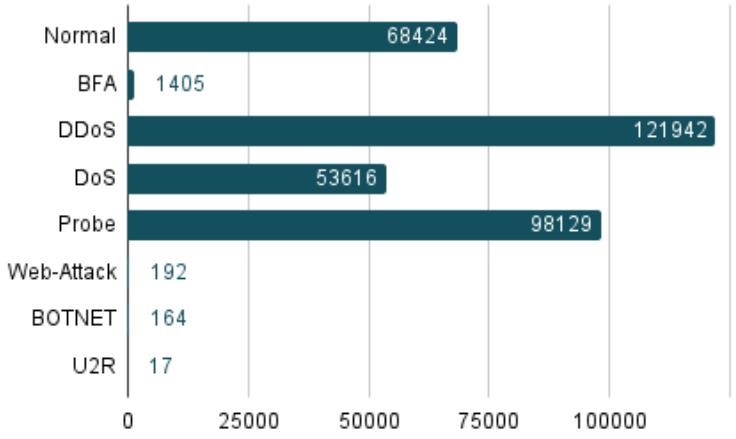

Figure 4.2: Distribution of classes in the inSDN Dataset

In our work, 67% of the dataset was used for training and validation. The remaining 33% was used for testing the dataset.

4.1.1 Dataset Exploration

The InSDN dataset provides both raw traffic traces captured by Tcdump tool and CSV files with flow features extracted by the CICFlowMeter [16] tool. The dataset is divided into three groups: the first contains flow features of normal traffic, the second contains flow features of attacks on their switch, and the third contains attack data from attacks on their Metasploitable-2 server, which includes popular application services such as HTTPS, HTTP, DNS, Email, FTP, and SSH.

The dataset is labeled with the "Label" column containing traffic type and class, where normal traffic is classified as *"Normal"* and attack traffic is classified into *"DoS", "DDoS", "Web Attacks", "R2L", "Malware", "Probe",* and *"U2R"* attack classes. Figure 4.2 illustrates the distribution of classes in the dataset. The dataset contains 68424 instances of normal traffic, 1405 instances of BFA traffic, 121942 instances of DDoS traffic, 53616 instances of DoS attacks, 98129 instances of probe attacks, 192 instances of web attacks, 164 instances of botnet traffic, and 17 instances of U2R traffic.

The dataset can be utilized for both supervised and semi-supervised learning models. According to Elsayed et al. [17], the features in the dataset can be categorized into network identifiers, packet-based attributes, bytes-based attributes, interarrival time attributes, flow timers attributes, flag attributes, flow descriptors attributes, and subflow descriptors attributes.

Reading Dataset

```
X = pd.read_csv("...../Processed_X.csv")
y = pd.read_csv("...../y.csv")

#dropping column with row numbers
X.drop(X.columns[[0]], axis=1, inplace=True)
y.drop(y.columns[[0]], axis=1, inplace=True)

#converting to tensor
X = torch.from_numpy(np.float32(X.values))
y = np.unique(y, return_inverse=True)
```

The code to read the data from the csv files is provided above. The *attackData* variable contains the combined data from the attacks on the normal server and their metasploitable-2 server. The attack and the normal traffic data are merged together in the *data* variable. It has 84 columns and 343889 rows.

	Flow ID	Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	Fwd Pkt Len Mean
0	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	124937.0	1071.0	9100	0	2839.477273
1	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	1071.0	439537.0	517	0	10.009346
2	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	66.0	758.0	66	0	13.200000
3	192.168.20.133-192.168.20.2-	192.168.20.2	53	192.168.20.133	35108	6	5/2/2020 13:58	6975	1	1	0.0	0.0	0	0	0.000000

Figure 4.3: Observing the Dataset

In Figure 4.3, the first few rows and columns of the dataset has been observed to

get an idea about the dataset. Elsayed et al. [17] erased the first 7 columns of the data that contained the network identifier attributes to reduce the risk of overfitting when they tested their dataset in various models to test it (*"Flow ID", "Src ID", "Src Port", "Dst IP", "Dst Port", "Protocol", "Timestamp"*).

The CSV files in the dataset are categorized in the three groups (Normal, attack on switch, attack on metasploitable - 2 server).The dataset exploration started by reading the three CSV files into a python file using the pandas library and combining them into a single dataFrame [30] object. After that, we identified the number of instances and the number of features by observing the shape of the dataFrame.

Measuring the Skewness in data

```
# Check the skew of all numerical features
numeric_feats = data.dtypes[data.dtypes != "object"].index

skewed_feats = data[numeric_feats].apply(lambda x:
      skew(x.dropna())).sort_values(ascending=False)

skewness = pd.DataFrame({'Skew' :skewed_feats})
```

Skewness in a data refers to the symmetry of a distribution. The code above shows the code snippet used to measure skewness in the *inSDN dataset*. A distribution is skewed if the tail on one side of the mode is fatter or longer than on the other. It is measured by the data's difference from the normal distribution. Skewed data can often lead to skewed residuals because "outliers" are strongly associated with skewness, and outliers tend to remain outliers in the residuals, making residuals skewed. As such, the skewness in the dataset was looked into next. Most of the features in the dataset is found to be very skewed, and may result in erroneous results if the model is not specified correctly.

Some of the names of the features and the skewness score is as provided - *Tot Fwd Pkts: 585.166828, Subflow Fwd Pkts: 585.166828, TotLen Fwd Pkts: 326.033065, Subflow Fwd Byts: 325.652028, Fwd Header Len: 174.563575, Subflow Bwd Byts: 163.858834, TotLen Bwd Pkts: 163.369948, Tot Bwd Pkts: 158.612005*

4.2 Data Pre-Processing

During the pre-processing stage, we combined all three groups (Normal traffic, Attacks data on the switch and Attack data from attacks on the Metasploitable-2 server) of data available in the dataset. We ran some basic checks over the dataset to observe instances of missing data and observed that most of the flag attributes columns were consistently set to 0 and contained no useful information. To eliminate irrelevant features and enhance the computational efficiency of subsequent steps, we removed 13 such empty features from the dataset, resulting in a total of 71 features.

Additionally, we dropped the first 7 columns containing network identifier attributes, as these attributes do not provide any significant contextual information for identifying attacks. After this step, we were left with a total of 64 columns.

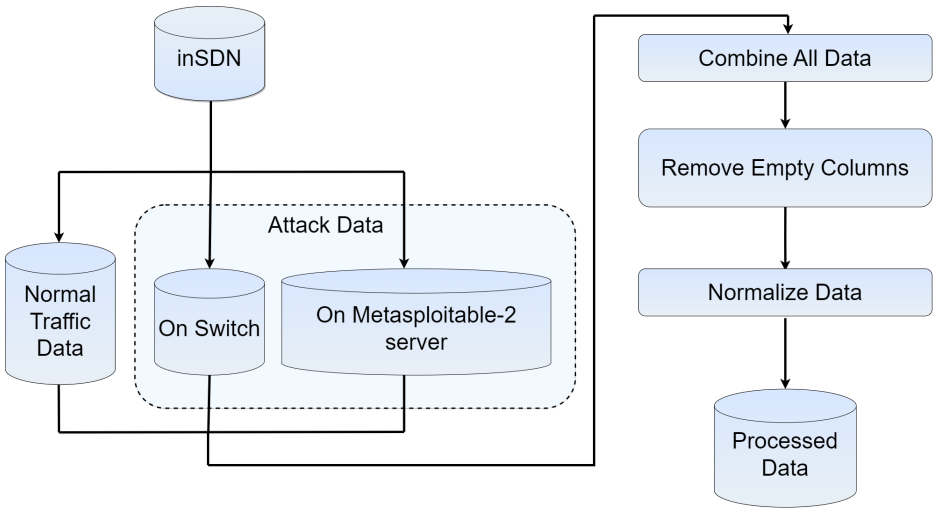

Figure 4.4: Data Pre-processing steps

To further improve the accuracy and effectiveness of the machine learning models, we normalized and scaled the data. We transformed the data to have a mean of zero and a variance of one, which ensured that all features contributed equally to the analysis and prevented bias towards features with higher values. This scaling method makes it easier to compare the importance of different features and provides a better representation of the dataset.

A graphical representation of the pre-processing steps can be found in Figure 4.4. This figure displays a flowchart of the steps taken in the pre-processing stage, starting

from merging the data groups to finalizing the processed dataset.

4.3 Feature Selection

4.3.1 Why PCA?

We decided to utilized PCA in the inSDN dataset to decrease the dimensionality of the dataset and preserve the original data's variance. The dataset contains 83 statistical features (and one column with ID), among which many may be redundant, irrelevant, or noisy, leading to reduced accuracy and overfitting in classification models. PCA is a feature selection method that transforms the original dataset into a new set of orthogonal variables called principal components. Each component captures a distinct amount of variation in the original data. By selecting only the most critical principal components, the dataset's dimensionality is reduced, and redundant features are eliminated, resulting in improved accuracy and efficiency of the machine learning model. In summary, PCA is employed in the inSDN dataset to decrease its dimensionality, eliminate redundant features, and improve the accuracy and efficiency of the machine learning model.

4.3.2 Implementation

We used R programming language to implement PCA. The Figure 4.5 explains the steps of Principal Component Analysis.

The first step in PCA involves computing the covariance matrix of the input data, given by:

$$\mathbf{\Sigma} = \frac{1}{n-1}(\mathbf{X} - \boldsymbol{\mu})^T(\mathbf{X} - \boldsymbol{\mu}) \tag{4.1}$$

where $\mathbf{X}$ is the $n \times d$ data matrix and $\boldsymbol{\mu}$ is the $d \times 1$ vector of column means. The eigenvectors and eigenvalues of the covariance matrix were then computed as follows:

$$\mathbf{\Sigma}\mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{4.2}$$

where $\mathbf{v}_i$ is the i-th eigenvector and λ_i is its corresponding eigenvalue. Eigenvectors with the eigenvalues greater than **0.1** were selected, which led to **31** features being

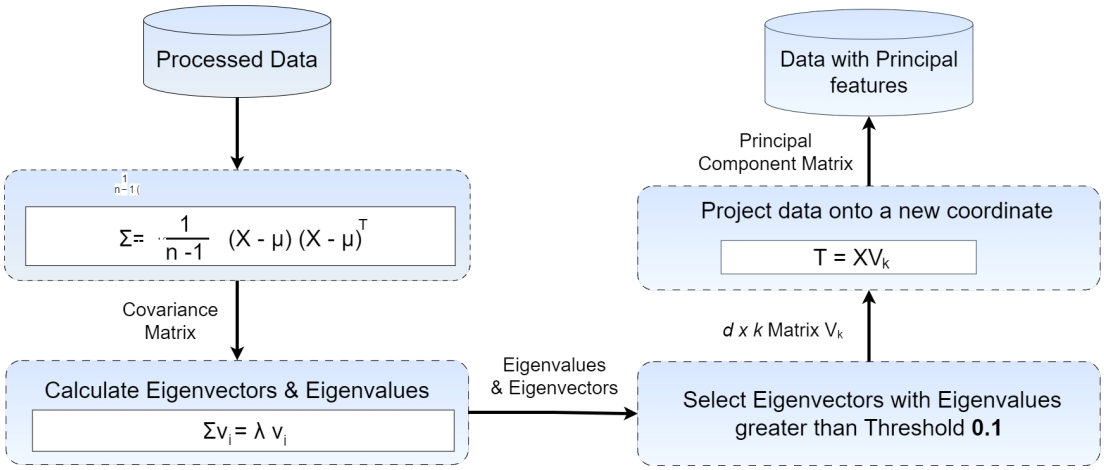

Figure 4.5: Steps of Principal Component Analysis

selected among the 64 present. The $d \times k$ matrix $\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$ where k is **31** is then formed to project the data onto a new coordinate system to obtain the principal components, given by:

$$\mathbf{T} = \mathbf{X}\mathbf{V}_k \tag{4.3}$$

where $\mathbf{T}$ is the $n \times k$ matrix of principal components. The resultant matrix $\mathbf{T}$ was saved in a .csv file, which we used on the later steps to train the DNN model.

The code for implementing PCA is as below. The Xs Matrix here is the normalized X matrix from the pre-processing steps. It is first multiplied by it's transpose and then divided by the number of instances to calculate the covariance matrix, as mentioned in the first step of PCA in Figure 4.5.

Implementing PCA

```
S = (t(Xs) %*% Xs)/nrow(Xs);

cov.svd <- svd(S)

# U and D Matrix
U = cov.svd$u
d = cov.svd$d
```

```
D = diag(d)

r = length(d[d>0.1])

Z = Xs %*% U[,1:r];
#Z is the target Matrix

write.csv(Z, file="Processed_X.csv")
write.csv(y, file="y.csv")
```

4.4 Classification Model

To test the effectiveness of the feature selection methodology and complete the hydrid model, we decided to start our experiments with a simple artificial neural network model. Our plan was to use increasingly more complex and deeper Neural Network Models. The previous works in literature has used various model including CNN based models utilizing LSTM and GAN.

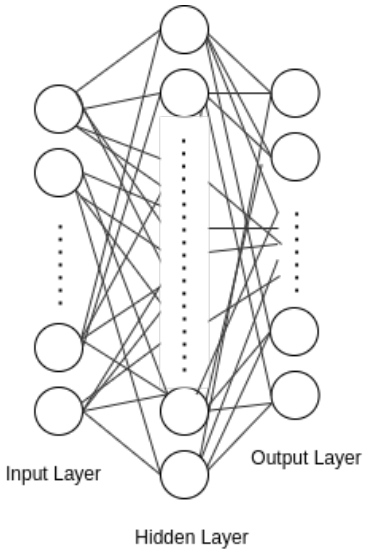

Figure 4.6: Neural Network Model

We started our experiments with a very simple Artificial Neural Network Model as shown in Figure 4.6. We deviated from the R programming language used in the previous pre-processing stage and used the python pytorch library to build the NN model.

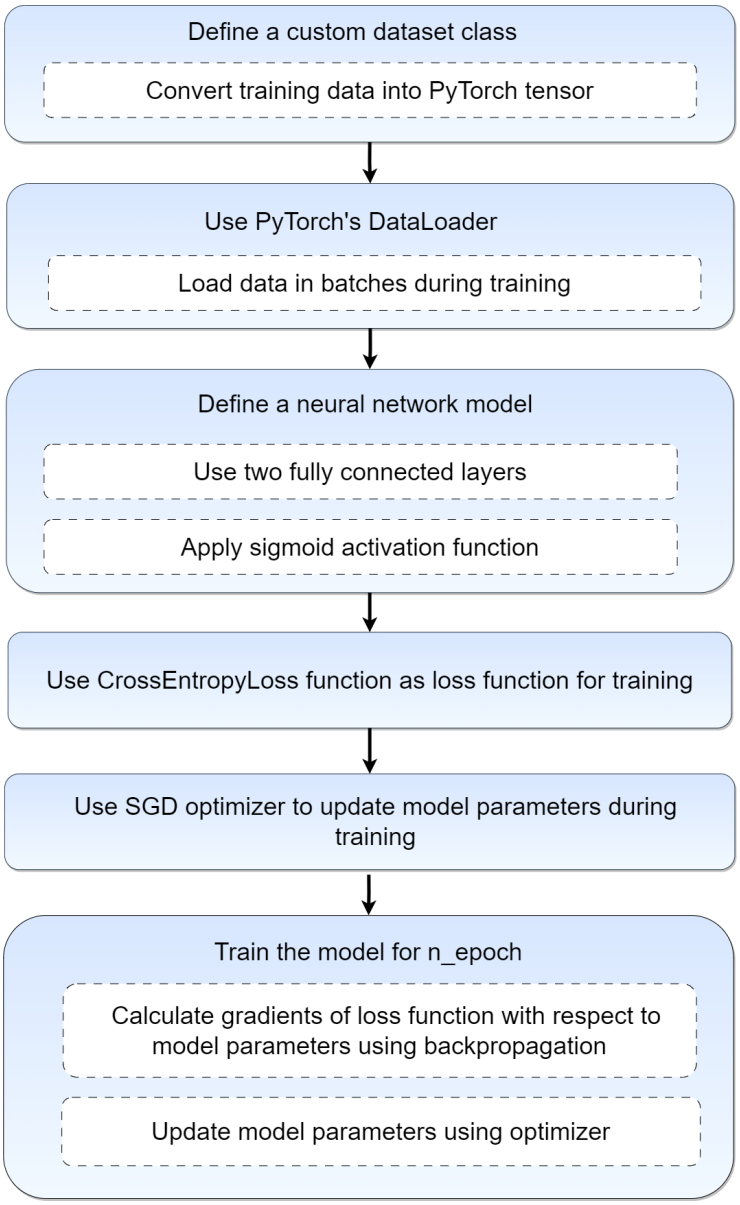

Figure 4.7: Steps in defining and training the neural network classifier model using python

While the R programming language is ideal for statistical analysis and understanding the dataset, the Python programming language provides much ease in implementing ANN models.

To combine the two different programming languages, the data from the pre-processing stage that was calculated using R was saved into a .csv file and it was read from the .csv file in the python implementation.

Figure 4.7 depicts the python implementation step by step.

The dataset is first converted to pyTorch tensor to be used in the model and then partitioned into training and validation sets using the *train_test_split* function from Scikit-learn. A PyTorch *DataLoader* is then created to efficiently load the big data in batches during training.

Following this, the neural network model is defined using the PyTorch *nn.Module* class. The model comprises two fully connected layers, and the output of the first layer is passed through a sigmoid activation function.

The *CrossEntropyLoss* function is defined to be employed as the loss function and the SGD optimizer is used to optimize the model parameters during trianing after that.

The training loop then runs for *n_epoch* number of epochs. For each batch, the gradients of the loss function with respect to the model parameters are computed using backpropagation, and the optimizer is used to update the model parameters.

In the binary classification problem, the output layer has 2 neurons and the multi-class classification model has 7 neurons in the output layer corresponding to the 6 attack classes and normal traffic. The input Layer always consists of 31 neurons that takes in 31 features selected from the feature selection step.

The binary classification model is only trained for 100 epochs and the multi-class classification model was trained for 500 epochs.

The code for building and training the model is as follows:

Building and training ANN Model

```
#Splitting the Dataset
x_train, x_val, y_train, y_val = train_test_split
            /(X, y, test_size=0.33, random_state=42)
```

```
#Defining Data loader
from torch.utils.data import Dataset, DataLoader
class Data(Dataset):
    def __init__(self):
        self.x=x_train
        self.y=torch.from_numpy(y_train)
        self.len=self.x.shape[0]
    def __getitem__(self,index):
        return self.x[index], self.y[index]
    def __len__(self):
        return self.len



data_set=Data()
trainloader=DataLoader(dataset=data_set,batch_size=64)

#Defining Model
class Net(nn.Module):
def __init__(self,D_in,H,D_out):
    super(Net,self).__init__()
    self.linear1=nn.Linear(D_in,H)
    self.linear2=nn.Linear(H,D_out)


def forward(self,x):
    x=torch.sigmoid(self.linear1(x))
    x=self.linear2(x)
    return x

#Defining parameters and functions
input_dim=31
hidden_dim = 32 # 10 for binary classification
output_dim=7   # 2 for binary classification

model=Net(input_dim,hidden_dim,output_dim)
criterion=nn.CrossEntropyLoss()
learning_rate=0.1
optimizer=torch.optim.SGD(model.parameters(), lr=learning_rate)

#Training Iterations parameters
n_epochs=100
loss_list=[]
```

```
#Function for plotting
def my_plot(epochs, loss):
    plt.plot(epochs, loss)

#n_epochs of training
for epoch in range(n_epochs):
    epoch_loss = []
    for x, y in trainloader:
        #clear gradient
        optimizer.zero_grad()
        #make a prediction
        z=model(x)
        loss=criterion(z,y)
        # calculate gradients of parameters
        loss.backward()
        epoch_loss.append(loss.item()*x.shape[0])

        # update parameters
        optimizer.step()

    loss_list.append(sum(epoch_loss)/len(epoch_loss))
    print('epoch {}, loss {}'.format(epoch,
                    sum(epoch_loss)/len(epoch_loss)))
```

Chapter 5

Experimental Results & Discussion

5.1 Evaluation Metrics

Accuracy, Precision and Recall are used as evaluation metrics.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{5.1}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.2}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.3}$$

Where TP (True Positive) represents the number of correctly identified normal traffic, TN (True Negative) represents the number of correctly identified attack traffic, FP (False Positive) represents the number of attack traffic identified as normal traffic, and FN (False Negative) represents the number of normal traffic identified as attack.

5.2 Evaluation Environment

All the experimental evaluation was done in google colab. While the NN model was trained with the GPU, the feature extraction using PCA was done in the CPU. The

evaluation was done with the GPU. A short summary of the specifications of the GPU, CPU and RAM is given below.

- GPU: Tesla K80 with 2496 CUDA cores. It has a 12 GB GDDR5 VRAM.
- CPU: Single core hyper threaded Xeon Processors (2.3 Ghz)
- RAM: 12.6 GB

5.3 Binary Classification Model

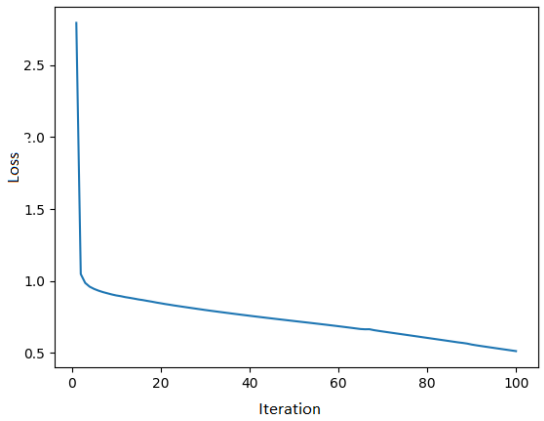

Figure 5.1: Loss graph for Binary classifier

The training loss for the Binary classification model is shown in Figure 5.1.

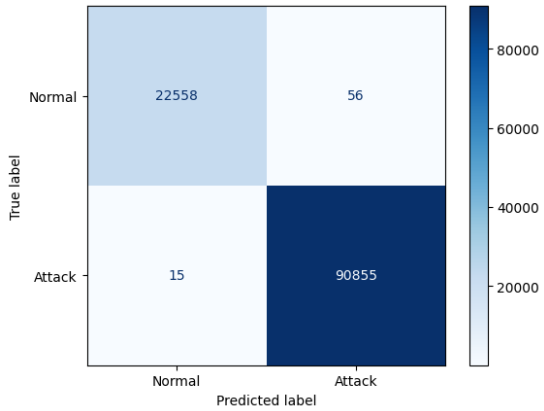

Figure 5.2: Confusion Matrix for Binary Classification Model

The confusion matrix for our feature selection based hybrid binary classification model is shown in Figure 5.2. Out of the 22614 Normal traffic instances, 22558 were correctly classified as such and out of the 90870 attack instances, only 15 were misclassified.

To compare our results with other models, we compared the results with theCNN model, LSTM model, CNN+LTSM model and the GAN Based Hybrid model that combined GAN, DAE, and RF algorithms developed by Ding et al. [15], who achieved the best results on the inSDN dataset. The results show that all models achieved high accuracy, recall, and precision values.

Table 5.1 presents the experimental results for the binary classification model.

Table 5.1: Experimental Results (Binary Classification Model)

Classification Model	Accuracy	Recall	Precision
CNN [15]	0.991	0.992	0.997
LSTM [15]	0.994	0.996	0.996
CNN+LSTM [15]	0.997	0.997	0.999
Hybrid Model [15]	0.998	0.999	0.998
Our model	**0.999**	0.999	0.997

It can be observed that our model outperformed the other models in terms of accuracy, achieving a value of 0.999. The recall and precision values of the our model were also high, with 0.999 and 0.997, respectively. On the other hand, the CNN+LSTM model achieved the highest precision score of 0.999. However, it is computationally much more complex than the other models. The Hybrid model performed well overall, with an impressive recall score of 0.999. Our model performs comparable to Ding et al.'s [15] Hybrid model that has the best achieved results in literature.

5.4 Multi-class Classification Model

The Multi-class classification model divides the attack traffic into 7 different attack types - BFA, BOTNET, DDoS, DoS, Probe, U2R and Web Attack.

Figure 5.3 shows the confusion matrix for the multi-class classification model.

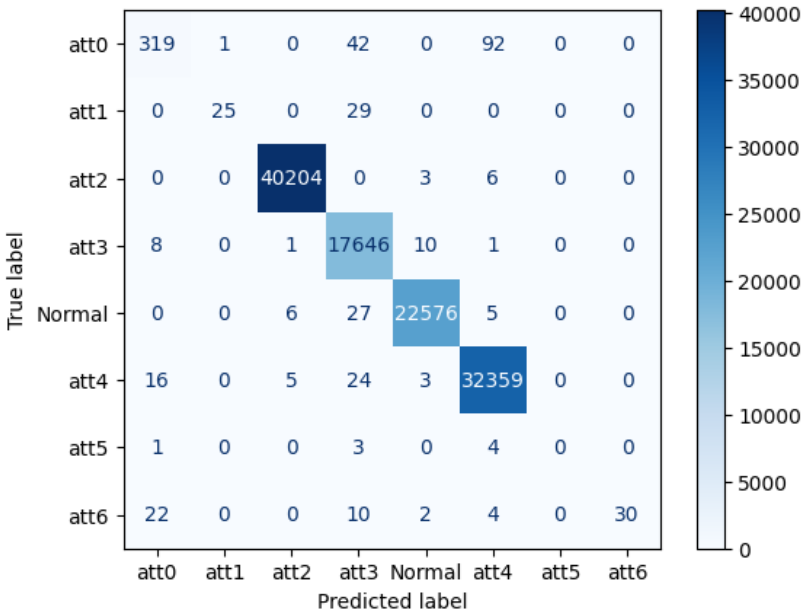

Figure 5.3: Confusion Matrix for Multi-class Classification Model

The values in the diagonal of the confusion matrix represents the number of traffic correctly classified by the multi-class classification model. The columns and row labeled as "Normal" signifies the normal traffic. The seven different types of attack traffic is labeled as $att0 - 6$, with $att0 = BFA$, $att1 = BOTNET$, $att2 = DDoS$, $att3 = DoS$, $att4 = Probe$, $att5 = U2R$ and $att6 = Web - Attack$. Out of the Out of the 22614 Normal traffic instances, 22576 instances were correctly classified as so. Only 38 instances of normal traffic were incorrectly classified at total. Out of the different attack traffics, only 18 at total were classified as normal traffic and all the rest were correctly classified as attack traffic. The amount of misclassification among the different attack traffic is very low as well, with maximum of 92 BFA attack traffic being classified as Probe Traffic. The reasoning for that can the smaller number of BFA traffic (454 instances) when compared to Probe traffic (32407 instances).

The training loss for the multiclass classification model is shown in Figure 5.4.

The results were compared with Ding et al.'s [15] CNN model, LSTM model, CNN+LTSM model and the GAN Based Hybrid model that combined GAN, DAE, and RF algorithms.

The table 5.2 presents the experimental results for the proposed classification model

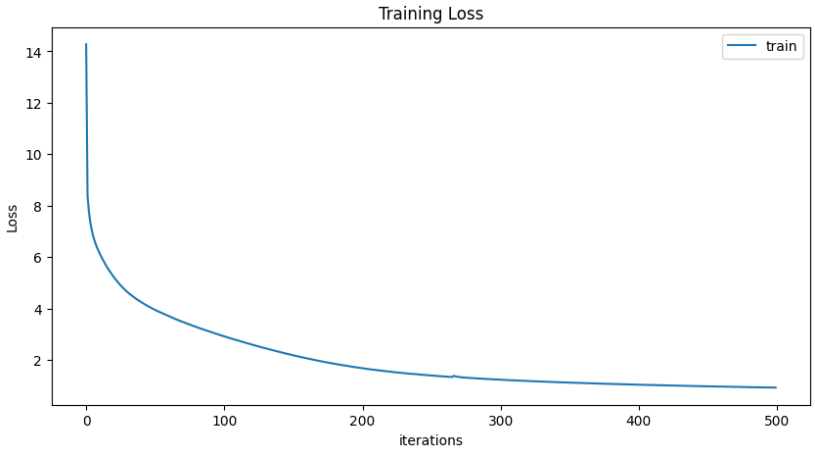

Figure 5.4: Loss graph for Multi-class classifier

Table 5.2: Experimental Results (Multi-Class Classification Model)

Classification Model	Accuracy	Recall	Precision
CNN [15]	0.948	0.948	0.951
LSTM [15]	0.946	0.946	0.948
CNN+LSTM [15]	0.924	0.924	0.922
Hybrid Model [15]	0.996	0.996	0.995
Our model	**0.9995**	**0.9992**	**0.9983**

and compares it with the CNN, LSTM, CNN+LSTM and Hybrid Model built by Ding et al. [15]. The proposed model achieved the highest accuracy of 0.9995, with Recall and Precision of 0.9992 and 0.9983, respectively.

In comparison to the other models, the proposed model achieved superior performance in all three evaluation metrics. The Hybrid Model achieved the second-highest accuracy of 0.996, with Recall of 0.996 and Precision of 0.995. CNN and LSTM achieved similar performances, with an accuracy of around 0.95 and Recall and Precision around 0.95 and 0.95, respectively. The CNN+LSTM model achieved the lowest accuracy of 0.924, with Recall and Precision of 0.924 and 0.922, respectively.

These results indicate that the proposed model outperformed the existing models, including the state-of-the-art Hybrid Model, in classifying the multi-class data.

5.5 Discussion

At the beginning of our experimental set up, the plan was to to start our experiments with a simple artificial neural network model and increase the model complexity and depth in later steps. Our plan included using a GAN model trained on the principal features to generate artificial data and combine it in the training phrase to train a more complex and deeper Convolutional Neural Network Model. The results gained from our first experiment surprised us, the simplest of our models out-performed the other models in literature. We suspect the reason to be the excessive number of redundant features in the dataset. Out of the 83 statistical features, only 31 made it out of the feature selection step even with a relatively low value of eigenvalue (only 0.1) selected in the third step of PCA. Previous Models in Literature leaned towards complex models to deal with the redundancy in the dataset. It can be seen from our analysis that while big, the inSDN dataset contains little information that requires deep learning models. We suspect that if the current models in literature are implemented after the feature selection (instead of the current ANN model), the results will prove to be close to perfect. The models may overfit easily.

Chapter 6

Conclusion & Future Work

6.1 Conclusion

In conclusion, this study presents a novel model for intrusion detection in software-defined networks (SDN) using a feature selection based methodology that incorporated PCA and ANN to reduce the dimensionality of input data. The proposed model is evaluated using the InSDN dataset, which comprises a diverse range of normal traffic and attack scenarios. Two simple neural network models are constructed for binary and multi-class classification using PyTorch.

We evaluated the performance of the models using different evaluation metrics such as accuracy, precision and recall. The results showed that the proposed model achieved high scores in all 3 metrics used, indicating its effectiveness in detecting various types of attacks while minimizing false positives. Our proposed approach can be further improved by incorporating more sophisticated neural network architectures and advanced feature engineering techniques, such as autoencoders or convolutional neural networks, to capture more complex relationships and patterns in the data. Additionally, incorporating real-time network traffic data and incorporating the use of multiple datasets from various sources could also improve the performance of the intrusion detection system.

6.2 Limitation

Our work is conducted on a different settings from the works it has been compared with. While our settings have been intentionally set to be lower in computational power and

lesser data has been used in the training set, the difference in the settings may indicate the results gained are not exact. We expect the results to be better in a settings similar to the literature it has been compared to.

6.3 Future Work

In the near future, we would like to implement and run the state-of-the-art existing models in the same settings and compare the results with and without feature selection. It would provide us further insights into the effects of feature selection before training IDS models for SDN. If the results are seen to be prevelant only on the inSDN dataset, it will prove our hypothesis that the dataset, despite being big - contains little information with a lot of redundant features that provides little to no contribution in the Indtrusion Detection System models.

We hope to pursue deeper knowledge in the related sectors to achieve success with our project and be able to transform our efforts into something greater to give back to the world and its people.

Bibliography

[1] M. Abdallah, N. An Le Khac, H. Jahromi, and A. Delia Jurcut, "A hybrid cnn-lstm based approach for anomaly detection systems in sdns," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, ser. ARES 21. Association for Computing Machinery, 2021.

[2] M. R. Ahmed, S. Islam, S. Shatabda, A. K. M. M. Islam, and M. T. Robin, "Intrusion detection system in software-defined networks using machine learning and deep learning techniques – a comprehensive survey," *TechRxiv*, 2021, preprint.

[3] M. F. Akbaş, C. Güngör, and E. Karaarslan, "Usage of machine learning algorithms for flow based anomaly detection system in software defined networks," in *Intelligent and Fuzzy Techniques: Smart and Innovative Solutions*. Cham: Springer International Publishing, 2021, pp. 1156–1163.

[4] J. Benabbou, K. Elbaamrani, , and N. Idboufker, "Security in openflow based sdn, opportunities and challenges," *Photonic Network Communications*, vol. 37, no. 1, pp. 1–23, 2019.

[5] K. Benzekki, A. E. Fergougui, and A. E. Elalaoui, "Software defined networking (sdn): a survey," *Security and communication networks*, vol. 9, no. 18, pp. 5803–5833, 2016.

[6] L. Boero, M. Marchese, and S. Zappatore, "Support vector machine meets software defined networking in ids domain," in *2017 29th International Teletraffic Congress (ITC 29)*, vol. 3, 2017, pp. 25–30.

[7] K. Cabaj, M. Gregorczyk, and W. Mazurczyk, "Software-defined networking-based crypto ransomware detection using http traffic characteristics," *Computers & Electrical Engineering*, vol. 66, pp. 353–368, 2018.

[8] H. Cheng, J. M. J. Liu, J. C. M. Wang, and J. Bian, "A compatible openflow platform for enabling security enhancement in sdn," *Security and Communication Networks*, vol. 2018, 2018.

[9] Q. Cheng, C. Wu, H. Zhou, D. Kong, D. Zhang, J. Xing, and W. Ruan, "Machine learning based malicious payload identification in software-defined networking," *Journal of Network and Computer Applications*, vol. 192, p. 103186, 2021.

[10] J. C. C. Chica, J. C. Imbachi, and J. F. Botero, "Security in sdn: A comprehensive survey," *Journal of Network and Computer Applications*, p. 102595, 2020.

[11] T. Chin, X. Mountrouidou, X. Li, and K. Xiong, "Selective packet inspection to detect dos flooding using software defined networking (sdn)," in *2015 IEEE 35th International Conference on Distributed Computing Systems Workshops*, 2015, pp. 95–99.

[12] L. Chuanhuang, W. Yan, Y. Xiaoyong, S. Zhengjun, and W. Weiming, "Detection and defense of ddos attack–based on deep learning in openflow-based sdn," *International Journal of Communication Systems*, vol. 31, no. 5, pp. 1–15, 2017.

[13] G. Cusack, O. Michel, and E. Keller, "Machine learning-based detection of ransomware using sdn," *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, 2018.

[14] A. Dawoud, S. Shahristani, and C. Raun, "Software-defined network security: Breaks and obstacles," *Networks of the Future: Architectures, Technologies, and Implementations*, p. 89, 2017.

[15] S. Ding, L. Kou, and T. Wu, "A gan-based intrusion detection model for 5g enabled future metaverse," *Mobile Networks and Applications*, vol. 27, pp. 2596–2610, 2022.

[16] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related," in *Proceedings of the 2nd*

international conference on information systems security and privacy (ICISSP), 2016, pp. 407–414.

[17] M. Elsayed, N.-A. Le-Khac, and A. Jurcut, "Insdn: A novel sdn intrusion dataset," *IEEE Access*, 09 2020.

[18] S. Garg, N. Kumar, J. J. P. C. Rodrigues, and J. J. P. C. Rodrigues, "Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in sdn: A social multimedia perspective," *IEEE Transactions on Multimedia*, vol. 21, no. 3, pp. 566–578, 2019.

[19] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *The second ACM SIGCOMM workshop on Hot topics in software defined networking*, 2013, pp. 55–60.

[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[21] S. Lee, J. Kim, S. Woo, C. Yoon, V. Y. S. Scott-Hayward, P. Porras, and S. Shin, "A comprehensive security assessment framework for software-defined networks," *Computers & Security*, vol. 91, p. 101720, 2020.

[22] H. J. Liao, C. H. R. Lin, Y. C. Lin, and K. Y. Tung, "Intrusion detection system: A comprehensive review," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 16–24, 2013.

[23] Y.-D. Lin, P.-C. Lin, C.-H. Yeh, Y.-C. Wang, and Y.-C. Lai, "An extended sdn architecture for network function virtualization with a case study on intrusion prevention," *IEEE Network*, vol. 29, no. 3, pp. 48–53, 2015.

[24] A. Maćkiewicz and W. Ratajczak, "Principal components analysis (pca)," *Computers & Geosciences*, vol. 19, no. 3, pp. 303–342, 1993.

[25] S. Maeda, A. Kanai, S. Tanimoto, T. Hatashima, and K. Ohkubo, "A botnet detection method on sdn using deep learning," *IEEE International Conference on Consumer Electronics (ICCE)*, pp. 1–6, 2019.

[26] V. Moorthy, R. Venkataraman, and T. R. Rao, "Security and privacy attacks during data communication in software defined mobile clouds," *Computer Communications*, vol. 153, pp. 515–526, 2020.

[27] S. Nawshin, S. Ahsin, I. Jahan, R. J. Moon, A. M. Haque, N. Muslim, and S. Islam, "Analyzing multicast routing algorithms on software defined network," in *2020 IEEE Region 10 Symposium (TENSYMP)*, 2020, pp. 1006–1009.

[28] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based ddos detection system in software-defined networking (sdn)," *EAI Endorsed Transactions on Security and Safety*, vol. 4, no. 12, 12 2017.

[29] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[30] Pandas, "pandas.dataframe," https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html, onine; accessed: 07-July-2023.

[31] M. S. E. Sayed, N.-A. Le-Khac, M. A. Azer, and A. D. Jurcut, "A flow-based anomaly detection approach with feature selection method against ddos attacks in sdns," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 4, pp. 1862–1880, 2022.

[32] A. Sebbar, K. ZKIK, and Y. e. a. Baddi, "Mitm detection and defense mechanism cbna-rf based on machine learning for large-scale sdn context," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, pp. 5875–5894, 2020.

[33] M. P. Singh and A. Bhandari, "New-flow based ddos attacks in sdn: Taxonomy, rationales, and research challenges," *Computer Communications*, 2020.

[34] C. Song, Y. Park, K. Golani, Y. Kim, K. Bhatt, and K. Goswami, "Machine-learning based threat-aware system in software defined networks," *26th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–9, 2017.

[35] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. G. and, "Deep recurrent neural network for intrusion detection in sdn-based networks," *4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pp. 462–469, 2018.

[36] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," *International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pp. 258–263, 2016.

[37] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, M. Ghogho, and F. E. Moussa, "Deepids: Deep learning approach for intrusion detection in software defined networking," *Electronics*, vol. 9, no. 9, pp. 1–18, 2020.

[38] C. Yu, J. Lan, J. Xie, and Y. Hu, "Qos-aware traffic classification architecture using machine learning and deep packet inspection in sdns," *Procedia Computer Science*, vol. 131, pp. 1209–1216, 2018, recent Advancement in Information and Communication Technology:.