Enhancing the Detection Rate and Mitigation of DDoS Attacks in Software Defined Network Controller Using Hybrid Deep Learning Model

Wengelawit Alemu Assefa

A Thesis Submitted to

The Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfilment of the Requirement for the Degree of Master's in

Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

June, 2024
Adama, Ethiopia

Enhancing the Detection Rate and Mitigation of DDoS Attacks in Software Defined Network Controller Using Hybrid Deep Learning Model

Wengelawit Alemu Assefa

Advisor: Ketema Adere Gemeda (PhD)

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfilment of the Requirement for the Degree of Master's in

Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

June, 2024

Adama, Ethiopia

Declaration

I, hereby declare that this Master Thesis entitled **"Enhancing the Detection Rate and Mitigation of DDoS Attacks in Software Defined Network Controller using Hybrid Deep Learning"** is my own work and has not been submitted for the award of any academic degree, diploma, or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation.

Wengelawit Alemu Assefa_ _____ _____

Name of Student Signature Date

Advisor Recommendation

I, the advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled "**Enhancing the Detection Rate and Mitigation of DDoS Attacks in Software Defined Network Controller Using Hybrid Deep Learning"** prepared under my guidance by **Wengelawit Alemu Assefa** submitted in partial fulfillment of the requirements for the degree of Masters of Science in Computer Science and Engineering (CSE). Therefore, I recommend the submission of the revised version of the thesis to the department following the applicable procedures.

Ketema Adere Gemeda (PhD) _____ _____

Major Advisor Signature Date

Approval page of M.Sc. Thesis

I, the advisor of the thesis entitled **"Enhancing the Detection Rate and Mitigation DDoS Attacks in Software Defined Network Controller Using Hybrid Deep Learning"** developed by Wengelawit Alemu Assefa, hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis

Dr. Ketema Adere_ _____ _____

Major Advisor Signature Date

We, the undersigned, members of the Board of Examiners of the thesis by **Wengelawit Alemu Assefa** have read and evaluated the thesis entitled "**Enhancing the Detection Rate and Mitigation of DDoS Attacks in Software Defined Network Controller using Hybrid Deep Learning**" and examined the candidate during the open defense. This is, therefore, to certify that the thesis is accepted for the partial fulfillment of the requirement of the degree of Master of Science in Computer Science and Engineering (CSE).

_____ _____ _____

Chairperson Signature Date

_____ _____ _____

Internal Examiner Signature Date

_____ _____ _____

External Examiner Signature Date

Final approval and acceptance of the thesis proposal is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

_____ _____ _____

Department Head Signature Date

_____ _____ _____

School Dean Signature Date

_____ _____ _____

Office of Postgraduate Studies Dean Signature Date

ACKNOWLEDGEMENTS

First, I would like to express my deepest gratitude to Almighty God for granting me the opportunity to start and complete my endeavors and for providing me the strength to persevere in all my pursuits.

I also extended my heartfelt thanks to my advisor, Ketema Adere (Ph.D.), who guided me from the beginning to the end of this journey. His positive ideas and comments on my work encouraged me greatly, and I will never forget the support he gave to all of us.

Lastly, I would like to thank my family members and friends for their unwavering support and encouragement during the ups and downs of my time at the University. I expressed my gratitude to all ASTU computer science and engineering staff members for their advice, motivation, and guidance in leading me into the academic research world.

CONTENTS

LIST OF TABLE

LIST OF FIGURES

LIST OF EQUATIONS

LIST OF ACRONYMS

API	Application program interfaces
CNN	Convolution neural network
DT	Decision tree
DL	Deep Learning
DDOS	Distributed denial of service
GA	Genetic algorithm
GRU	Gated recurrent unit
KPCA	kernel principal component analysis
KNN	K-Nearest Neighbor
LSTM	Long short-term memory
LDDoS	Low Distributed denial of service
MLP	Multilayer perceptron
ML	Machine Learning
NB	Naïve Biyase
ONOS	Open network operating system
POX	Partial Oxidation
RF	Random forest
SDN	Software Defined Network
SVM	Support vector machine
TCAM	Ternary Content Addressable Memory

Abstract

Software-defined network (SDN) is a cutting-edge method of managing and controlling networks, using software-based controllers or application programming interfaces to interface with the underlying hardware that divides the data plane from the control plane. SDN resolved the issues with the conventional network architecture like lack of flexibility*, scalability, and centralized control, leading to inefficient resource allocation, complex management, and limited adaptability to changing network requirements. However, there are several security risks that this new paradigm architecture is susceptible to, particularly DDoS attacks. This attack involves continuously sending packets to the computer networks or the controller which makes the controller out of service. Users won't be able to access the system or network resources because of these attacks. Many researchers were using ML and nowadays, DL to detect DDoS attacks on SDN, where they lack mitigation of the detection they have done and mostly classification of the attack types and feature selection. In this paper, we classified the traffic into normal and different attack types found in both datasets used which are InSDN and CICDDoS2019. These data are publicly available and generated from the real scenario of the SDN environment. We classified the traffic by using various DL (deep learning) like LSTM (long short-term memory), RNN (recurrent neural network), MLP (multi-layer perception), and hybrid of the DL models with CNN (convolutional neural network), with a separate train test of 80/20. After choosing our dataset we pre-processed it by cleaning and transforming it. Next, we used the chi-square feature selection algorithm to select the highest relevant features from both our datasets. From our experimental results, we concluded that the hybrid CNN-MLP model demonstrated the highest performance metrics, achieving an accuracy of 99.89% using the CICDDoS2019 dataset and 99.91% using the InSDN dataset. Our model outperformed the baseline LSTM model by 0.46%, reaching the highest accuracy in both datasets. Furthermore, we successfully mitigated attacks classified by the hybrid model. Our network architecture, implemented using Mininet, comprised 3 POX controllers, 8 switches, and 30 hosts, with 2 hosts designated as attackers by applying flow rules to drop malicious packets based on identified attacker IP addresses. Additionally, to ensure network stability, if a bottleneck occurs on the main controller, the other controllers seamlessly will take over.*

KEYWORDS: *SDN, DDoS, InSDN, Hybrid Deep Learning, Feature selection, POX controller, Flow rules, Mitigation, Mininet*

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

Software-defined Networking (SDN) has become a new approach that acted like a central control center, unlike the existing traditional internet architecture, providing flexible management by making the network programmable from a logical centralized control plane (Gadze et al., 2021). SDN's ability to separate the data and controller planes gave developers more flexibility and made application maintenance easier in these kinds of situations (Wang et al., 2014). There were three main parts of SDN: the application layer, the controller layer, and the infrastructure layer (Mladenov, 2019). The controller layer held configurations based on rules that are made and applied in the data plane; the data plane comprises network devices like routers and switches that are used to forward data packets. The application layer is the building applications, such as load-balancing traffic and IDs, that are built (Munir et al., 2022). The control plane is composed of POX, ONOS, Floodlight, RYU, and open Daylight controllers (Gebremeskel et al., 2023).

SDN offers significant capabilities, but it also presents security vulnerabilities. This susceptibility stems partly from its reliance on a central controller, which became a prime target for attackers. DDoS attacks were common due to SDN's centralized architecture, especially at the controller layer. These attacks had a network-wide impact (Nadeem et al., 2022). Traditionally, ML was used to detect such attacks. However, DL has emerged as the preferred method due to its effectiveness in detecting DDoS attacks (Mittal et al., 2023).

DDoS attacks typically involve a large number of hacked, compromised computers generating an enormous amount of traffic directed toward the target machine or SDN controller. As a result, the targeted system became overloaded with resources. Trying to handle the surge of network traffic, ultimately preventing it from doing its intended tasks (Mansoor et al., 2023). DDoS assaults affect enterprises in a big and diverse way. By overloading network resources, rendering websites, programs, or entire networks slow or unreachable, they can seriously disrupt services. Extended downtime results from this disturbance, which interferes with regular

operations and user access. Such disruptions have significant financial repercussions, including lost productivity, lost sales opportunities, and possible compensation claims from impacted clients. Furthermore, recurrent or severe DDoS attacks can cause catastrophic reputational harm, destroying customer confidence and trust and possibly losing business (Mittal et al., 2023).

In addition to service disruption and financial losses, DDoS assaults impose significant operational expenses on businesses. Mitigating the effects of these attacks frequently entails paying for more bandwidth, security services, and IT personnel to oversee and resolve situations. Furthermore, DDoS attacks can work as a decoy for other malicious actions, such as data breaches or malware implantation, which compromise important information. Failure to protect against DDoS assaults and ensure continuous service availability can result in noncompliance with industry rules, incurring legal penalties and fines. These broad implications highlight the vital necessity for strong security measures to identify and mitigate DDoS attacks efficiently.

A major challenge in detecting and mitigating DDoS attacks on SDN networks was the lack of suitable training data (Mittal et al., 2023). Many existing datasets used in (Nadeem et al., 2022) were collected from small, non-SDN networks, leading to models that perform poorly in real-world SDN deployments due to differences in traffic scale and behavior. Additionally, (Murtuza & Asawa, 2024) utilized synthetic data gathered from the SDN network environment. However, this data was often small-scale and wasn't mostly found publicly, hindering validation and replication of research findings. The limited availability and outdated nature of existing data posed significant obstacles for researchers working on SDN-based DDoS detection (Pandithurai et al., 2024).

Machine learning was the dominant approach for DDoS attack detection in previous works. However, deep learning methods are now gained favor due to their ability to perform both feature extraction and data classification, even from incomplete data (Ahuja et al., 2021). During the training phase, DL methods performed sophisticated mathematical operations over multiple hidden layers with various parameters (Mansoor et al., 2023). During the comparison of machine learning algorithms with deep learning approaches, extensive employment of matrix operations is undertaken. GPU is effective in matrix operations, the availability of the devices

makes computation efficient and rapid (Taheri et al., 2023). In the case of mitigation, we have seen that many researchers' papers don't include mitigation ((Ahuja et al., 2021), (Sahoo et al., 2020), (Gebremeskel et al., 2023), (Sun et al., 2022)) where the existing research mainly focused on detection and in few cases the classification only. The problem is that descriptions of the attack types happening on the controller are not provided, as most of the models are trained for binary classification of the attacks and mitigation isn't done. Also, studies that aim to mitigate DDoS attacks still require improvement in their accuracy.

1.2 Motivation of the Study

While SDN offers numerous advantages over the traditional network architecture, nowadays security and reliability of SDN networks are becoming more vulnerable to attacks, where attackers generate excessive traffic flows that overwhelm the links between switches in the data plane. This creates bottlenecks, effectively preventing legitimate users from accessing network services and causing data loss (Taheri et al., 2023).

Numerous cyberattack efforts, including website targeting, malware deployment, infrastructure scans, DDoS attacks, and network hacking, have been reported to the Information Network Security Agency (INSA). In particular, DDoS attacks account for a significant percentage of these threats—1,493 occurrences have been reported. These cyberattacks are primarily intended to harm infrastructure and interfere with the provision of services.

A significant gap in the current research is the lack of feature selection, leading to increased response and training time and making it computationally expensive, which decreases model accuracy. Existing research often lacks focus on categorical classification and effective mitigation strategies, with many studies addressing only binary attack types.

The inspiration for undertaking this study arises from the need to improve the classification and mitigation of DDoS attacks targeting the SDN controllers we utilize, with the simultaneous goal of minimizing their impact when such attacks occur.

1.3 Statement of the Problem

DDoS attacks occurring in SDN networks are the highest threat to SDN which makes it to be more researched and solved by many researchers so far. So, when attackers send many packets

to the controller it causes a bottleneck or failure of the controller, where the controller is not in control of the other planes. So, it causes other planes not to function. Much research has been done to solve these issues. Like (Novaes et al., 2021), machine learning was used to detect the attacks' binary classification. (Nugraha& Murthy, 2020) They proposed a hybrid DL CNN-LSTM model to detect slow DDoS attacks evaluated on a custom dataset in an SDN environment.

However, out of 31 reviewed papers, 25 of them have long training times because most of the works don't include feature extraction which leads the time to be long to train the model (Badotra et al., 2020), (Ahuja et al., 2021),(Deepa et al., 2019), (Novaes et al., 2021). This model mainly focused on detection rather than classification or mitigation. While detection of the attack is useful, classification of attacks happening to the controller is more needed to specifically know the attacks happening at the controller, which is limited to only knowing if it's been attacked or not (binary classification). Further, mitigation is not applied for those detected attacks which leads to further vulnerability to the SDN environments and not proper functioning of the controller, they are only concerned with the detection part.

The data set used for these papers is simulated or gathered from non-SDN networks which are small in size which makes it difficult to check in the real world on the SDN network whether it detects or not. Even if they synthesized data (Ahuja et al., 2021), (Santos et al., 2020), (MyintOo et al., 2019), (AbdulRaheem et al., 2023), (Badotra et al., 2022) or gathered custom data from the traffic for this purpose, still their small in size and data can't be found publicly to check the validity of their work and to check in more enhanced method and even for comparison of other data sets.

In (Li et al., 2022) mitigation technique, the limitation is that they blocked specific flow or compromised the host, not the traffic malicious, which leads to low performance of their work and most research works are done on the first layer ((A. Singh et al., 2023), (AbdulRaheem et al., 2023), (Li et al., 2022), (Nadeem et al., 2022)). The baseline paper (Taheri et al., 2023) stated that InSDN data sets were available, had a small dataset size, and had been gathered from an SDN network environment with no redundant data. We can apply this data set for our

research by using many methods along with the CICDDoS2019 dataset, easily done to detect DDoS attacks on SDN controllers using this data.

To overcome the mentioned problems above, we have proposed to use the InSDN and CICDDoS2019 data set along with feature selection and hybrid deep learning approach to detect and categorically classify those attacks and finally mitigate the attacks based on the packet flow IP address. We considered the mitigation work to drop the detected and classified attacks to a specific router after the second layer classification is done. Even further to improve the accuracy performance, to reduce running time, and to select the most relevant features.

1.4 Research Questions

For the mentioned issues earlier, here are the questions to be answered by the study.

1. How to identify and preprocess the features of the InSDN and CICDDoS2019 datasets to improve the accuracy of detecting and classifying DDoS attacks on SDN?
2. How can a hybrid deep learning model be developed to enhance the detection and classification of DDoS attacks on SDN?
3. How can we accurately evaluate the performance of the selected hybrid models to ensure their effectiveness?
4. How to mitigate the DDoS attack after detection is done?

1.5 Objective

1.5.1. General Objective

This research aims to develop a model for enhancing the detection and mitigation of DDoS attacks using feature selection and hybrid deep learning for the data sets.

1.5.2. Specific Objective

- To identify and select key features from InSDN and CICDDoS2019 data sets that contribute to detecting the DDoS attacks accurately.
- To identify and develop different hybrid deep learning models for enhancing the detection and classification of DDoS attacks on SDN for data sets.

- To propose an architecture for enhancing the detection and classification of DDoS attacks on SDN.
- To train, test, and optimize hybrid deep learning model performance by adjusting hyper-parameters and model architecture
- To establish appropriate metrics to evaluate and analyze the effectiveness of the hybrid deep learning models with the two datasets and consider the highest performance.
- To develop mitigation strategies for the attacks based on the detection result.

1.6 Significance of Study

This study examines critical components of data center and network security in SDN settings. The project intends to improve the availability, security, and dependability of SDN-based data centers by exploiting SDN-specific data sets, as well as to enable proactive detection of potential cyber assaults to boost network security and reduce interruptions to important services. Furthermore, the study focuses on improving the accuracy of detecting and mitigating DDoS assaults in SDN systems by employing sophisticated feature selection approaches such as chi-square selection to improve detection accuracy while reducing computing costs. In addition, the study assesses performance indicators, and optimizes procedures to satisfy the severe needs of modern network security environments, while streamlining attack mitigation operations to limit the impact of cyber-attacks and maintain network integrity and availability.

1.7 Scope and Limitation

1.7.1. Scope of the Study

The scope of this research includes the use of the pox controller to detect whether the attack has happened to it or not. The study deals with the InSDN and CICDDoS2019 SDN datasets. The reason for choosing this data set for this study was because it was gathered from an SDN network while other data sets are from conventional networks. We have used hybrid deep learning to detect and classify the DDoS attacks on the controller. Considered six and four attack types among the datasets. Mitigation is done on the second layer after classification is done.

1.7.2. Limitations of the Study

The Limitation of this study was using the dataset of the attack happening on the controller and considering six types only happening by external volumetric attacks. We considered a POX controller on the Mininet to mitigate the attacks.

1.8 Organization of the Paper

The rest of the paper is organized in the following way: Chapter Two**:** we have discussed more about the SDN and its vulnerability, especially about the DDoS attack widely, and reviewed the different types of attacks. We also included related works to relate it to our work and show the weakness of other works in many ways like their contribution, limitation, and further understanding of their work. Chapter Three: in this part, we discussed the method and technique we used to apply to our proposed solution. It also discussed the preprocessing steps in detail including the evaluation metrics. Chapter four: discussed the architecture of the proposed solution of our research and environment setup. Tasks were done for implementing our proposed solution including the codes for preprocessing, feature selection, training, and evaluating it. In chapter five, we discussed our results and experiments to conclude our proposed model solution. Finally, we constructed a conclusion, contribution, and future works in chapter six.

CHAPTER TWO

Literature Review and Related Work

This chapter briefly shows and explains an overview of SDN, types of SDN, various types of attacks in SDN, including DDOS attacks, techniques for identifying and classifying DDOS attacks, various statistical and classification algorithms, and related work on the security of the controllers.

2.1. Literature Review

2.1.1. Definition of SDN

SDN has some features that make it unique these are its virtual network configuration, dynamic network policy enforcement, expanded network management, and further well-designed centralized console. Additionally, compared to traditional methods, the overall operational cost is far cheaper (Bhuiyan et al., 2023).

Intending to replace traditional networks, it separates the control logic from network hardware (switches and routers). Administrators must work harder to maintain regular network functionality and overall network security because of the centralized control plane (Bawany et al., 2017). Compromised network objects include sensitive data about users and their network organization. Later, the details might be exploited for illicit purposes, like taking down the network (Bhuiyan et al., 2023).

2.1.2. Traditional Network vs. SDN

A traditional network is a conceptual model that characterizes the communication function of telecommunication or a computing system without regard to its underlying internal structure and technology, as SDN is a network architecture that radically changes the design and management of modern network topologies, particularly in today's hyper-scale data centers with enormously vast networks (Mansoor et al., 2023). All networking devices are centralized controlled, managed, and configured by a single controller (Mansoor et al., 2023).

As shown below in Table 2-1 and Figure 2-1, we have provided the main difference between the traditional network and SDN.

Table 2-1: Traditional Network vs. SDN

Parameter	**Traditional network**	**SDN network**
Based on	Protocol	Centralized controller
Network configuration	Many steps needed	Easy and process is automatic
Management	Each device needs to be updated individually	The programmability of the separated plane is easy using centralized control
Formation	Control, management, and data planes are all bundle up together on one device.	Planes are separated with a centralized controller.
Performance	Limited information	Cross-layer information
Calculating forward rule	Network equipment switch or router in the server	Software running the server
Global view	Not available	Available
Maintenance cost	High	Less

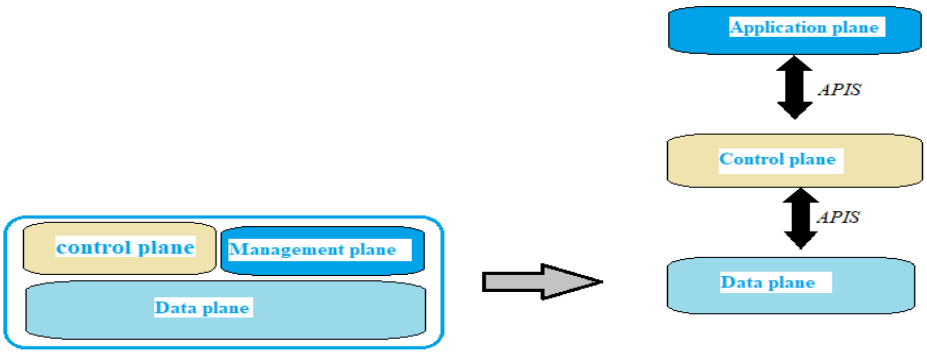


Figure 2-1: Traditional Network vs SDN

2.1.3. SDN Architecture

Compared to the conventional decomposition of a network into protocol layers, a Software-defined network simplifies network management by decoupling architecture into three layers and bringing innovation in the network communication through the controller (Amin et al., 2018). SDN decouples the network control from the data forwarding plane as Fig. 2-1. By separating the control layer and data planes, network control may be programmed, and underlying infrastructure can be abstracted for network services and applications. (Gebremeskel et al., 2023).

Most of the intelligence in the SDN is found in the control plane, which is managed and cared for by a software networking operating system. The forwarding plane continued to perform network analysis and management tasks (Neira et al., 2023). SDN is thought to be a technology that can effectively manage the entire network, simplifying and managing intricate network architectures (Joëlle& Park, 2018). The controller is the bridge that connects the upper application known as the northbound interface (Said Elsayed et al., 2020), and the underlying switching equipment known as the southbound interface, is a crucial component of SDN.

The three tiers of SDN of the network (Rawat& Reddy, 2017) are:-

- **The infrastructure layer:** It comprises a range of packet-forwarding devices, including switches, routers, virtual switches, and wireless access points. They are Open-Flow switches in SDN. These devices are solely focused on packet forwarding and have flow tables to hold the rules for packet forwarding that are established by the control plane (Munir et al., 2022).
- **The control layer:** It involves the Network Operating System (NOS), also known as the SDN controller, which is the brains of the entire network, including Flood-Light (Vedhapriyavadhana et al., 2018), Ryu, NOX, and POX (Nowsin et al., 2019). It decides the packet forwarding and puts those judgments into practice for network devices(J. Singh & Behal, 2020).
- **The application layer:** it's the SDN's top layer, which is in charge of transferring business and security-related software applications. To configure, monitor, and control the network, the Application Plane communicates with the Control Plane and the Infrastructure Plane (Bhuiyan et al., 2023). Applications that make use of the

programmability and flexibility of the SDN infrastructure are deployed and managed by network administrators and developers on the Application Plane.

It makes it possible to create cutting-edge network services (Rawat& Reddy, 2017) SDN uses two APIs for communication: northbound and southbound. Northbound serves as a translator between applications and the control layer, allowing them to request and receive network information. Southbound connects the control layer to actual network devices and conveys data traffic handling instructions between the data and control plane. OpenFlow, a common southbound protocol, allows the control layer to program these devices. (Bholebawa et al., 2016). It operates by routing and processing packets according to flow entries in the flow table. Each entry includes counters for monitoring statistics, match fields to identify packets based on header data, and a sequence of operations. The Open Flow switch parses packet headers, compares them to the flow table, and executes instructions for matching entries. If no match is found, the switch follows instructions to either discard the packet or send it to the controller (Mansoor et al., 2023).

As we can see in Figure. 2-2 (J. Singh & Behal, 2020), the controller is separated from the data and applications plane, making it dynamic network management and flexible traffic handling. So, for the controller to communicate with the data plane through a southbound interface these protocols (J. Singh & Behal, 2020) make possible effective control over the data plane.

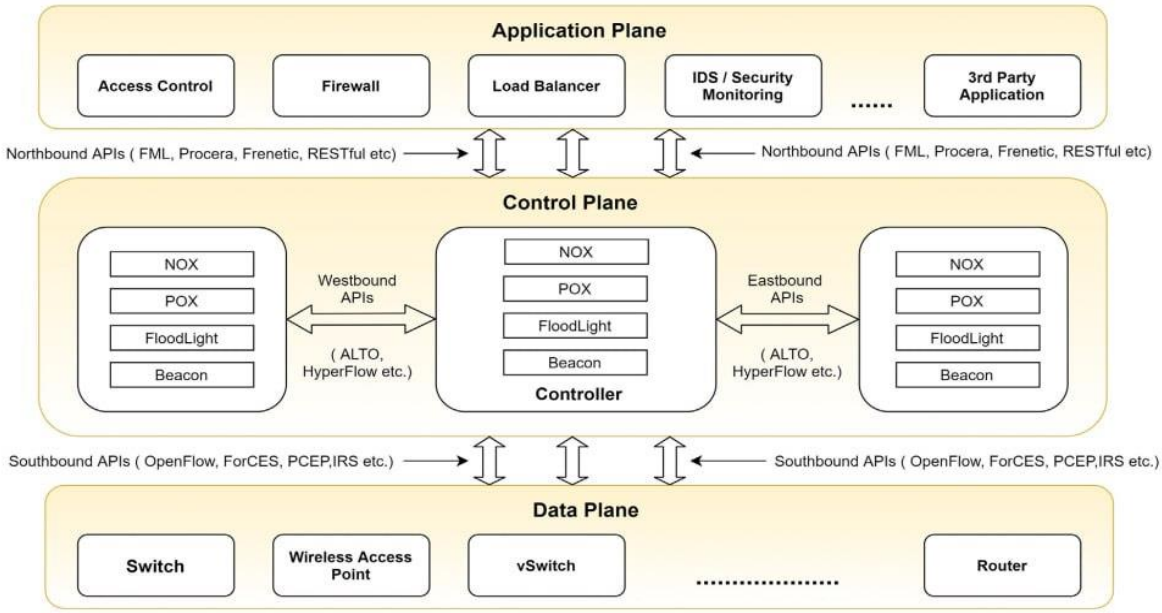


Figure 2-2: SDN Architecture

Numerous protocols are available, including Protocol-Oblivious Forwarding (POF), ForCES, OpFlex, and OpenFlow; however, as Open-Flow has emerged as the de facto protocol, numerous groups are striving to standardize it. And to communicate with the application plane through a northbound interface (Joëlle& Park, 2018); The NOS gives the application developers access to these APIs. It conceals the network's core details and aids in network programming. Commonly used APIs include, for instance, FML, Procera, NetKAT, and Frenetic.

2.1.4. SDN Benefits

Decoupling the control plane from the data plane in SDN offers numerous advantages.

- Network activities are regulated by a controller software program rather than the underlying hardware. This separation simplifies network administration significantly. It facilitates the establishment of large-scale attacks and defensive experiments (Aldaoud et al., 2023).
- The controller, with a network-wide view, can enforce consistent security policies and analyze traffic patterns for potential threats (Turner et al., 2023).
- SDN's programmability allows intelligence gathering from intrusion detection and prevention systems (Taheri et al., 2023).
- Centralized administration provides a global perspective, while the flexibility of the SDN controller enables rapid deployment of new application services and infrastructure to align with evolving business goals (Oluwasogo Adekunle, 2015).

2.1.5. DDoS Attacks

The rate of arriving inbound packets to the network is great in a DDoS assault, and the combination of legal and faked packets would collectively bind the network resources, causing resource exhaustion (Banitalebi Dehkordi et al., 2021). If this process continues, the server could be unreachable for any new lawful packets that arrive, and the packet could be dropped, rendering the network unstable. DDoS attacks are classified into three types: volumetric attacks, protocol exploitation attacks, and application layer attacks (Sahoo et al., 2020). DDoS attack is a volumetric attack because multiple computers target a particular application and a protocol attack in which the attackers send a flood of fake requests to the victim host using the IP address (Joëlle& Park, 2018).

2.1.5.1. DDoS Attack Types on SDN

SDN can mitigate DDoS assaults by isolating the data planes from the control plane. However, the security vulnerabilities around SDN remain unsolved (J. Singh & Behal, 2020). Because of its architecture, SDN may be a target for DDoS assaults. As previously said, SDN is divided vertically into three functional layers (Adekoya et al., 2020). DDoS assaults can compromise these tiers. These flaws are summarized below in Figure 2-3.

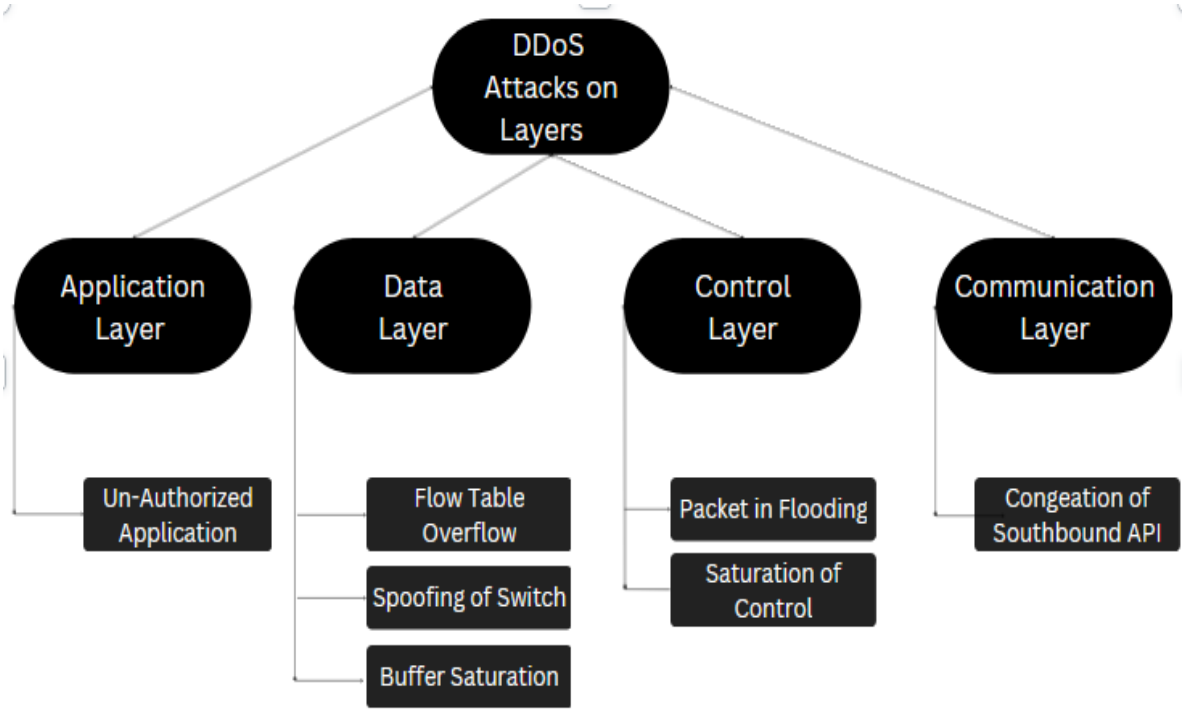


Figure 2-3: DDoS Attacks on Different Layers on SDN

- **Buffer Saturation:** When a new packet comes in SDN, the switch matches it with a flow rule. If no matches are found, the switch requests that the controller construct a new rule that sends part of the packet while buffering the rest. The complete packet is transmitted to the controller if the buffer is full. An attacker can take advantage of this by flooding the switch with false flows, depleting the buffer and resulting in a buffer saturation attack that prevents the switch from buffering real packets(J. Singh & Behal, 2020).
- **Flow Table Overflow:** TCAM is used by Open Flow switches in SDN to hold flow tables. If no match is detected when a new packet is received, the switch seeks a new rule from a controller. Attackers take advantage of this by flooding the switch with

bogus packets and filling the flow table with bogus entries. Legitimate entries are dropped, depriving authorized users of access to resources. Due to their high cost and power consumption, switches with limited TCAM memory are vulnerable to DDoS assaults(J. Singh & Behal, 2020).

- **Congestion of Control-Data Plane Link:** When a packet comes without a matching entry, the switch requests the flow rule from the controller, transmitting a portion of the packet via a packet_in request and buffering the remainder. The switch delivers the entire packet to the controller if the buffer is full. When attackers flood switches with fake packets, this procedure can generate channel bottlenecks, producing a slowdown for real requests to the controller(J. Singh & Behal, 2020).
- **Controller Saturation**: When an attacker flood switches with bogus packets and the switches pass all requests to the controller, the controller gets overburdened with unclear requests. This overload depletes the controller's throughput and processing power, causing the overall SDN architecture to degrade. To overwhelm and disturb the controllers, attackers often generate enormous aberrant packets (J. Singh & Behal, 2020).
- **Packet_in flooding**: When a vswitch gets a mismatched flow, it sends a packet_in message to the centralized controller via the southbound interface, requesting a forwarding rule. An attacker floods the vswitch with packets while impersonating IP addresses, forcing the vswitch to deliver bulk packet_in messages to the controller. This flood overwhelms the controller, making it unreachable to real users because it is busy with only bogus flow requests (J. Singh & Behal, 2020).
- **Spoofing of Switch:** In this case the malicious actor can spoof a switch's IP address in this attack, by delivering control messages to the controller with the modified address. Imagine a scammer tricks your network into thinking they're a real device (like a switch) by copying its ID (IP address). This fake device then sends phony control messages to the network's main control center, pretending to be the real switch. All the while, the real switch is still trying to talk to the control center. Fooled by the scammer, the control center accidentally cuts off the real switch and starts listening to the fake one instead. This disrupts communication and slows down your network. These fake messages are a clear sign of an attack, and you'll need to take steps to stop it (J. Singh & Behal, 2020).

- **Unauthorized Applications**: in this part, many of the apps in the application plane get access to network resources to deliver services to the controller and network. Some apps can access network resources by using the instances of other applications. However, there is no authentication or authorization of apps to check their legitimacy; as a result, certain malicious applications get access through instances of other applications and affect network behavior, becoming the cause of network performance degradation(J. Singh & Behal, 2020).

2.1.5.2. DDoS Attacks on the Control Layer

For a centralized SDN, the SPF is a challenging task. Despite the availability of a multi-controller SDN topology, SPF remains a major threat to DDoS attacks. When the network is overwhelmed, packets are created regularly. If the controller receives a runoff request, its resources could depleted after a given length of time (Mousavi & St-Hilaire, 2015) causing a malfunction of the controller. So, the buffer of the target switch becomes overflowing with packet headers, the entire flow is transmitted to the controller (Kansal& Dave, 2017). The new flows could consume the bandwidth required to control the link. In this situation, the regular flow request generates congestion. Message packet manipulation may result in an unanticipated side-channel attack. So, we studied the DDoS attacks happening on the controller side these are and showed them in Figure 2-4:

Volumetric Attacks: Imagine a highway jammed with so many fake cars (data packets) that real traffic (legitimate users) grinds to a halt. This attack floods the network with massive amounts of data, overwhelming its capacity and making it unavailable.

Protocol Exploitation Attacks: Think of a network as a group of people communicating with a specific language (protocol). Attackers exploit weaknesses in this language by sending nonsensical messages (malformed packets) that confuse network devices. These devices malfunction, wasting resources and disrupting communication.

Application Layer Attacks: Imagine a restaurant bombarded with fake online orders. Application layer attacks target specific weaknesses in applications or services, flooding them

with seemingly legitimate requests that overwhelm and crash them. This disrupts access to the real functionality for legitimate users.

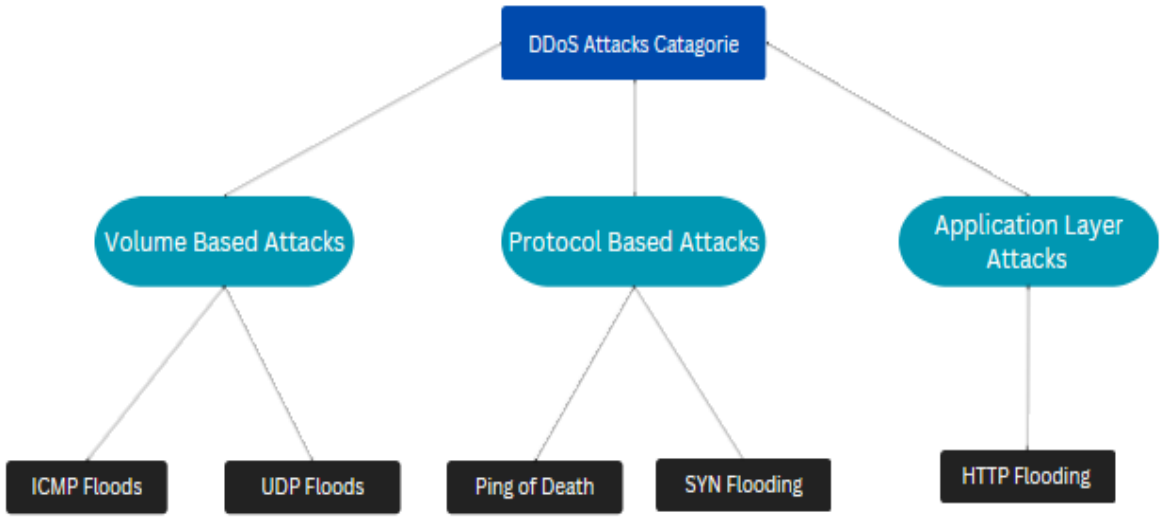


Figure 2-4: DDoS Categories

2.2. DDoS Attack Detection Technique in SDN

DDoS assaults can be detected using traditional ML and DL-based classifiers in SDN-based systems. These classifiers, however, differ in terms of architecture, internal parameters, and characteristics (Munir et al., 2022). One of the central concepts of ML and DL is to create a classifier model from an existing dataset. One model may perform admirably on one dataset but poorly on another. As a result, it is critical to create a classifier model that performs well on a specific dataset (Said Elsayed et al., 2020).

2.2.1. ML Technique

By automatically developing the model based on the training dataset, machine learning techniques are used to combat infiltration and DDoS attacks on SDN controllers or switches.

SVM, decision trees, random forests, and other machine-learning algorithms were frequently used in the previous network to detect anomalies (Khempetch & Wuttidittachotti, 2021),(Nadeem et al., 2022). These methods can also be advantageous for SDN-based anomaly detection systems. They have used different datasets, but most research has not yet got high accuracy in their work.

(Alashhab et al., 2022; Sekar et al., 2023) suggest using a mixed machine learning approach based on SVM and a decision tree to identify suspicious traffic provided by the detecting trigger mechanism. However as the network is subjected to more intense traffic, the controller's workload increases, and the effectiveness of DDoS detection declines. As well the classification of almost all are binary and not well classified (Banitalebi Dehkordi et al., 2021; Hassan et al., 2024; Myint Oo et al., 2019; Santos et al., 2020).

(Kanber et al., 2022; Santos et al., 2020; Sekar et al., 2023) have feature selection but need domain expertise, models are not suited for capturing complex patterns, and non-linear relationships. While considering scalability it lacks improvement in it.

2.2.2. DL Technique

Deep Learning is a subset of machine learning that utilizes neural networks with multiple layers to automatically, learn representations from the data. In the idea of detecting DDoS attacks, DL involves leveraging these neural networks to analyze and classify network traffic patterns to identify malicious activities indicative of a DDoS attack. DL automatically learns the most relevant features from raw network traffic data. This allows for the identification of complex patterns without human interventions. We have selected the deep learning model more because we can get higher detection accuracy due to its ability to learn from large realistic and complex datasets. Also reduces false positives and ensures legitimate traffic is not mistakenly flagged.
LSTM and RNN are groups of gates that can keep data in memory for a protracted amount of time (Habib & Khursheed, 2024).

1. **RNN:** - These deep learning models hold information about previous inputs in the sequence in a hidden state, which allows them to evaluate sequential data, including text or time series (Mansoor et al., 2023). Every input word in the RNN has the same weight, and its parameters are shared by all sections. In classic neural networks, input and output are independent of one another (Elsayed, Le-Khac, Dev, et al., 2020; Gadze et al., 2021; Gebremeskel et al., 2023), and there is no notion of using previous output as input for the current step. But in applications like forecasting and prediction, like guessing a sentence's next word or projecting future sales, historical data is required, necessitating the recall of earlier input (Seba et al., 2024). Therefore, in contrast to a multilayer

perception, an RNN gets information from two sources. There is one input from the past and one from the present. The number of steps that an RNN can run for is limited due to decreasing gradient difficulties (Bhuiyan et al., 2023). They are used when the gap between the past and present data prediction info needed is small (Ahuja, Singal, & Mukhopadhyay, 2021)

2. **LSTM: -** is a special type of RNN that can be used to learn from and memorize long-range term dependencies and used for long-range prediction(Ahuja, Singal, & Mukhopadhyay, 2021). The default setting is to retain past knowledge for a considerable amount of time. Information is tracked over time according to LSTMs. Time-series prediction benefits from their recollection of previous inputs whereas RNN has a disadvantage because it works in short distances while LSTM can work in a large gap of the input and predict place because it saves short long memories (Ahuja, Singal, & Mukhopadhyay, 2021). In addition, to making time-series predictions, LSTMs are frequently employed in voice recognition, music production, and pharmaceutical research. Because of their chain-like structure and four interconnected layers, they have separate communication functions. The LSTM can store a large amount of historical data and is designed to avoid long-term reliance problems (Seba et al., 2024). Examining the flow data over an extended time helps improve detection accuracy (Gebremeskel et al., 2023).
3. **GRU: -** This is a special kind of Recurrent Neural Network (RNN) that, like LSTMs, tackles the vanishing gradient problem. But here's the twist: GRUs have a simpler design with fewer "gates" which is compared to LSTMs (Gebremeskel et al., 2023). This makes them train faster and perform better, especially when dealing with smaller datasets. Despite being simpler, GRUs are still good at capturing relationships between data points over time (temporal dependencies).
4. **MLP: -** is a feedforward neural network, which consists of multiple layers of perceptron that give activation functions. Fully coupled input and output layers are found in MLPs. The multi-layer perceptron has multiple hidden layers, but they have the same number of input and output layers which has the curse of dimensionality. Additionally, the MLP (Nugraha& Murthy, 2020) portion of the intended model classifies the attacks into

distinct DDoS attack types using the reduced feature produced by chi-square as inputs to minimize bias and performance overhead.

2.2.3. Hybrid DL Technique

In this technique, we combined two different DLs for better performance of our thesis. We have mainly considered CNN to be a hybrid with the other different deep learning models.

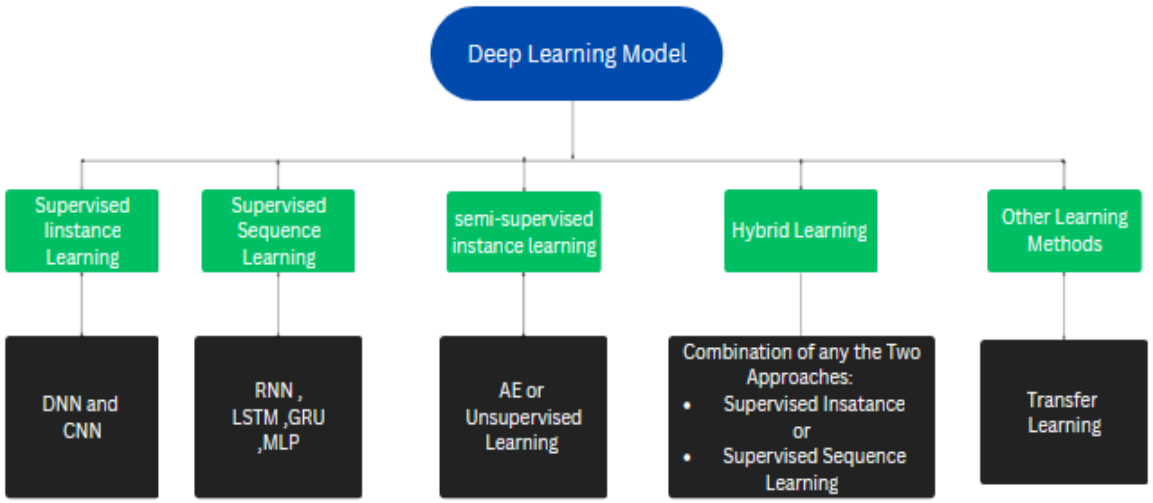


Figure 2-5 Deep Learning Types

2.3. Dataset Description

2.3.1. InSDN Dataset

Proposing a virtualized network testbed to generate a new SDN dataset, namely InSDN (Elsayed, Le-Khac, & Jurcut, 2020b). The performance of IDS approaches is determined by the quality of the training datasets. One of the most significant obstacles in deploying detection systems is the scarcity of current real-world datasets. The fundamental reason for the scarcity of public datasets in the intrusion detection space is privacy and legal concerns(Said Elsayed et al., 2020). In this paper, we use the InSDN dataset to test our proposed hybrid deep learning model. To address the inadequacies of current datasets in the context of SDN networks. The nature of attacks in SDN differs from those that usually affect conventional networks.

- A range of attack scenarios and attack types, including DoS, DDoS, Web assaults, password-guessing, bot-net, exploitation, and probe attacks, are included in the InSDN dataset.
- The attacks in the dataset are derived from internal and external networks in order to simulate real-world attack scenarios (Said Elsayed et al., 2020).
- More than 80 feature elements are included in the CSV file format, such as Protocol, Duration, Byte Count, Packet Count, and so on. (Elsayed, Le-Khac, & Jurcut, 2020b) report that the total number of dataset instances for both normal and attack traffic is 343,939, of which 68,424 are instances of normal data and 275,515 are examples of attack traffic.
- Each dataset has its unique set of security system needs, and benchmark datasets should be tailored to the specific context. This indicates that the attack vectors should be deployed following the peculiarities of the new architecture where some focused on certain attack types (Elsayed, Le-Khac, & Jurcut, 2020a).
- Furthermore, separating the control plane and data plane introduced several new SDN-specific risks. As a result, picking the incorrect features had a considerable impact on the performance of the most well-known classifiers (Elsayed, Le-Khac, & Jurcut, 2020b). These assaults are difficult to detect since the intruder is legitimately connected to the victim's server. As a result, employing such a dataset for model evaluation can be a useful indicator of how well the model matches the real-world environment.
- It has no redundant records, which precludes the learner model from favoring the most frequent records (Said Elsayed et al., 2020).

2.3.2. Other Data Sets

This section examines the publicly available datasets generated by conventional networks. Datasets are commonly used in traditional networks for intrusion detection, and they have been used to evaluate ML algorithms created for anomaly detection approaches in SDN networks (Elsayed, Le-Khac, & Jurcut, 2020b). Data not gathered from the SDN environment had many problems and wasn’t make it suitable for the research done for SDN works. Some of the data sets are listed below and summarized in Table 2-2:

- **The KDD'99** dataset, which is based on DARPA packet traces, is commonly used for evaluating intrusion detection systems. Notably, record redundancy was a problem, with duplicated records reaching 78% in the training set and over 75% in the testing file. This duplication impedes detection procedures, resulting in a bias toward frequent recordings such as DoS attacks and a reduction in accuracy for low-frequency attack categories such as R2L and U2R.
- **The NSL-KDD** dataset, a modified version of the KDD'99 dataset, the dataset used is a modified KDD'99 version, addressing issues like duplicate records, with training and testing subsets. The testing set has a more diverse attack distribution, including 17 attacks not in the training set. However, both KDD'99 and NSL-KDD were outdated and didn't reflect current network traffic patterns. The use of an obsolete TCP version in the DARPA dataset further limits its relevance (T. E. Ali et al., 2023). Both datasets have attributes irrelevant to Software-Defined Networking. Prior SDN research faced challenges with low detection rates and false alarms using NSL-KDD. Most SDN studies using KDD'99 and NSL-KDD primarily identify DoS attacks due to difficulties in accessing content features crucial for detecting U2R and R2L attacks in packet data.
- **ISCX2012** In a simulated network environment, the ISCX2012 dataset employs two profiles, Alpha for attack traffic and Beta for normal traffic. It collects 20 packet features and focuses on DoS and brute force assaults. However, the breadth of DoS attacks is limited due to a lack of coverage for vulnerabilities across many OSI layers. Notably, the statistic only includes HTTP traffic, failing to represent actual internet traffic, which is dominated by HTTPS. The dataset, like KDD'99 and NSL-KDD, lacks sufficient characteristics extracted from the Open Flow protocol for complete machine learning evaluation.
- **CICIDS 2017** The addition of an HTTPS Beta profile reflects the shifting online traffic scenario. Due to their inherent complexity(Aslam et al., 2023), profile scripts were used to generate normal traffic behavior. Notably, Panigrahi et al. (2018) note concerns such as 288,602 missing class labels, 203 missing information instances, and an overly large size with much redundant data.
- **CSE-CIC-IDS2018**, employs the concept of profiles to systematically generate the dataset, with B-profiles for normal traffic and M-profiles for attack scenarios, covering

similar attack scenarios as CICIDS 2017. However, it shares the same inherent problems as CICIDS 2017, including issues related to missing class labels, missing information instances, and an overly large dataset with redundant records. Additionally, similar to CICIDS 2017, CSE-CIC-IDS2018 employs synthetic traffic, introducing challenges associated with the use of artificially generated data seemingly irrelevant for Intrusion Detection System training (Songma et al., 2023).

Table 2-2: Dataset comparisons

Dataset	Realistic traffic	Label	Attacks Traffic	Type of network	No. of attribute	Format	Network Environment
KDD'99 (1998)	NO	Yes	DoS, Probe, R2L, U2R	Small network	41	Other	Conventional
NSL-KDD (2009)	NO	Yes	DoS, Probe, R2L, U2R	Small network	41	Other	Conventional
ISCX2012	YES	Yes	HTTP DoS, SSH brute force, infiltration	Small network	-	Packet, flow	Conventional
CICIDS 2017	YES	Yes	Botnet, DoS, Goldeneye, DDoS, SSH, SQL INJECTION, infiltration	Small network	83	Packet, flow	Conventional
CSE-CIC-IDS2018	YES	Yes	Botnet, DoS, Goldeneye, DDoS, SSH, SQL INJECTION, infiltration	Small network	83	Packet, flow	AWS platform

InSDN 2020	YES	Yes	Botnet, DoS, DDoS, Web attacks, password brute-forcing, probe, exploitation	Small network	83	Packet , flow	SDN
CICDDoS2019	YES	YES	MSSQL, SSDP, DNS, LDAP, NetBIOS, SNMP, NTP, TFTP, SYN FLOOD, UDP FLOOD, UDP-LAG	Large network	88	Packet , flow	SDN

2.3.3. CICDDoS2019 dataset

This dataset is aimed to be used for detecting and classifying 12 attacks of DDoS attacks having around 1234567 gathered from the SDN network. As (Mittal et al., 2023) mentioned one of the main reasons for this data development is because of little processing overhead. Data sets are used to carry out using TCP/UDP-based protocol.

As shown in Figure 2-6, illustrates the type of attacks mentioned in CICDDoS2019 these are:

Reflection-based DDoS: In these kinds of attacks, the attacker uses elements belonging to a third party to conceal their identify. Attackers in these scenarios alter transmissions by forwarding them to reflector servers, where the source IP address is configured to correspond with the IP address of the target (Kshirsagar & Kumar, 2022). The intention is to overload the victim with so many answer packets that it causes service outages or network congestion. These assaults may focus on holes in protocols at the application layer. The attack is carried out using transport layer protocols.

Exploitation-based DDoS: these attacks directly target weaknesses in the victim's infrastructure. Hackers identify vulnerabilities, in protocols, software, or settings. Exploit them to inundate the target resources with commands or traffic. Common tactics include network ports, exhausting server resources, or exploiting weaknesses in the application layer.

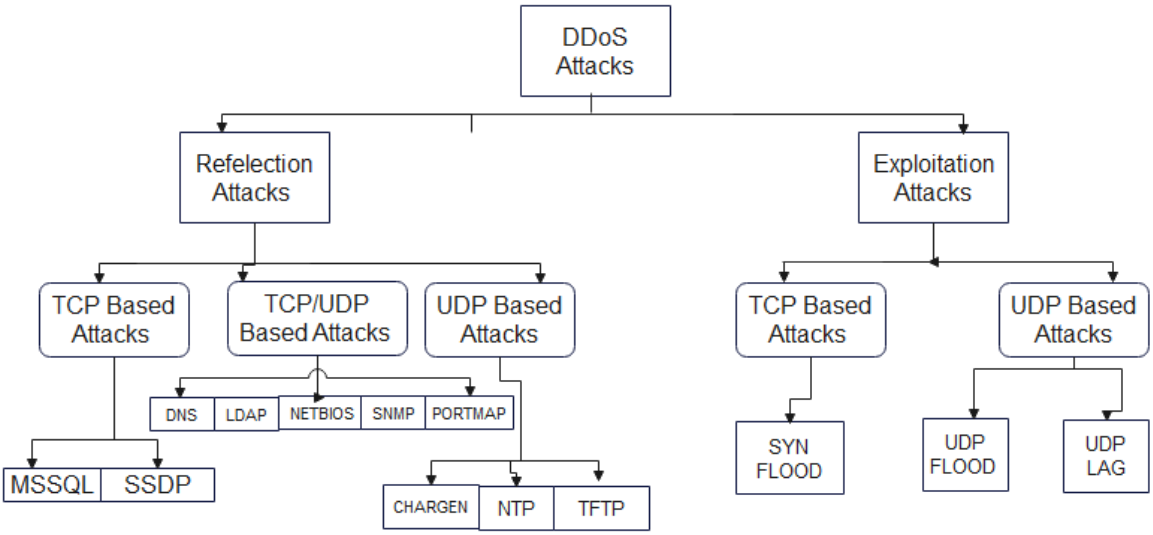


Figure 2-6: CICDDoS2019 Dataset Description

2.4. Related work

The DDoS attack has been the greatest attack happening on SDN many researchers have done much research on the mitigation and detection and proposed an ensemble technique by adopting different ML algorithms mainly KNN, SNM, SOM, and NB. Only detecting anomalous behavior of the data traffic in the SDN using dataset CAID2016, they combined the algorithms and found that SVM-SOM produced a higher accuracy of 98.12% than the others.(Santos et al., 2020) Stated that the data they used was created by a scapy tool to be in the SDN environment of their work, they used four MLs (SVM, MLP, DT, and RF) which only detected DDoS attacks. DT was best regardless of having the lowest time to process (Myint Oo et al., 2019) used advanced-SVM to determine the traffic as normal or attack traffic where data are generated by a scapy tool in an SDN environment, having an accuracy of 97%.

(Ahuja, Singal, & Mukhopadhyay, 2021) aimed to classify the traffic into normal and malicious classes based on features given in the dataset by using various DL techniques. They got an accuracy of 99.75% by applying SAE-MLP. (Sahoo et al., 2020) used KPCA and GA for feature extraction and selection and SVM for the detection and got an accuracy of 98.7% by reducing the dimension of the features into a new layer. (Nadeem et al., 2022) the proposed selection of optimal features and machine learning (SVM, KNN, NB, RF, and DT) classifier derived to detect SDN attacks. They obtained an accuracy of 99.7% by the RECURSIVE feature elimination method using the NSL-KDD dataset.

(Zhang et al, 2021) the authors proposed a solution using asynchronous messages and open flow entries in their work they used stacked sparse auto-encoder and also information entropy for feature extraction. They proposed a hybrid of SSAE-SVM and entropy, for the detection and mitigation of the DDoS attacks on SDN. (AbdulRaheem et al., 2023) this study proposes a snort and Zeek enabled with an ML-based model to classify the benign traffic from DDoS attack traffic. This study's main contribution is the discovery of new features for DDoS attack detection, which made it difficult to distinguish authorized traffic from attack traffic when spread across many points of origin. By using custom-generated data sets, they got an accuracy of 97.2%.

(Sun et al., 2022) Researchers (Sun et al., 2022) proposed a new approach using a combination of two deep learning techniques (CNN and GRU) to identify LDoS attacks. This method extracts key details like the number of packets and bytes of data from network traffic information (flow rules). It then calculates averages of these values to feed into the combined model. To improve the accuracy of this method, the researchers also enhanced an existing algorithm (Sailfish) to automatically fine-tune the settings of the CNN and GRU models during training. By these optimized models, the system effectively captured both short-term trends (using CNN) and long-term patterns (using GRU) within the network traffic data, leading to more accurate LDoS attack detection (Habib & Khursheed, 2024).

(Nugraha& Murthy, 2020) they proposed a hybrid DL CNN-LSTM model to detect slow DDoS attacks evaluated on a custom dataset in the SDN environment. All considerable performance metrics are above 99% in their detection work on the DDoS attacks SDN controller.

(Said Elsayed et al., 2020) proposed deep learning to detect DDoS attacks on SDN controllers using the InSDN dataset and achieved an accuracy of 93.4%. We used this paper to support our research work.

The suggested evaluation metric measured methods to assess the benchmark dataset are false positive rate and detection accuracy (Mansoor et al, 2023). The results show that the methods successfully detect DDoS assaults with average values of 94.186%, 92.146%, 8.114%, and 94.276% for detection accuracy, average precision, average FPR, and average F1-measure, respectively.

2.4.1. Summary of the related work on this study

Table 2-3, shows the above-mentioned related works done on DDoS detection and mitigation in SDN controllers.

Table 2-3: Related work

Authors	Controller	SDN environment	Dataset	Method	Accuracy	Gaps or limitations
(Sahoo et al., 2020)	POX	No	NSL-KDD	DT, KPCA, and GA	98.12% for binary classification	• needss improvement in accuracy • Only detection done • NO Feature selection
(Nugraha & Murthy, 2020)	Floodlight -	Yes	Custom data by Hping 3	CNN-LSTM	99%	• Not reliable • Small data size • Mitigation and classification is not done
.(Ahuja, Singal, & Mukhopadhyay, 2021)	POX	Yes	Custom made	SAE-MLP	99.75%	• data set not available • Binary classification • Mitigation is not done

(Nadeem et al., 2022)	POX	NO	NSL-KDD	SVM, KNN, NB	99.7%	• Doesn't use SDN environment dataset • Mitigation is done on first-level detection. • Features eliminated normally
(Sun et al., 2022)	RYU	Yes	CICDOS 2017	CNN-GRU	99.1%	• Dataset not publicly available • Doesn't include mitigation
(Li et al., 2022)	Floodlight	No	Generated by scapy	DL	98.98%	• Doesn't use SDN environment dataset • Block compromised host rather than malicious traffic
(A. Singh et al., 2023)	ONOS	Yes	Generated from Wireshark	SVM-RF	99.1%	• Mitigation done on first level detection • No Feature selection
(AbdulRaheem et	RYU	YES	Custom data	Snort, Zeek	97.2%	• Data is not available

al., 2023)				AND ML		• Improvement performance accuracy • Mitigation done on first level detection
(Mansoor et al, 2023)	RYU	YES	Custom data	DL	98.7%	• Data is not available • Research done by 1 DL only • Mitigation NOT done
(Gebremeskel et al., 2023)	Pox	Yes	CICDDoS2019	Entropy and LSTM	99.42%	• Mitigation is not done

2.5. Baseline paper

Different DL methods have been used in detecting and classifying DDoS attacks with feature selection or not. While coming to our baseline paper (Gebremeskel et al., 2023),(Said Elsayed et al., 2020) used dataset CICDDoS2019 and InSDN data respectively. They both compared different DDoS attack types and classified them into categorical and binary using various DL methods. The problem is that mitigation is not done on both research and classification of attacks is not done on InSDN datasets along with no feature selection which leads to long training time. The accuracy results of their research were 99.42% and 99.51% in CICDDoS2019 and InSDN datasets respectively.

CHAPTER THREE

RESEARCH METHODOLOGY

3. Introduction

This section discusses the methods used to enhance the detection and mitigation of DDoS attacks on SDN controllers using a feature selection and hybrid deep learning system. The research methodology included data collection, analysis, preprocessing, and the hardware and software tools used for training and testing our model.

Overall steps taken to do our research work:

- **Domain understanding**

  It's the first step we did for our research; we identified which part of the domains in the network area we wanted to work on by differentiating those areas and understanding them well. We picked one and went to the next steps.

- **Literature review**

  In this part, we reviewed many papers related to our work domain. The papers were articles, journals, and even conference papers.

- **Research gap identification**

  From the papers we reviewed; we found gaps and identified problems to be solved.

- **Designing the proposed solution**

  In the fourth step of our research process, we developed a proposed solution to address the gaps identified in our literature review. This design was informed by chi-square feature selection, which involved selecting more relevant features to minimize training and testing time. A hybrid deep learning approach was employed for the detection and classification mechanism, utilizing datasets from SDN environments and the publicly available CICDDoS2019 dataset. These datasets were generated by the SDN environment, enhancing the accuracy of our research on detecting and mitigating DDoS attacks on SDN controllers. Figure 3-1 below illustrates the general workflow of the proposed solution.

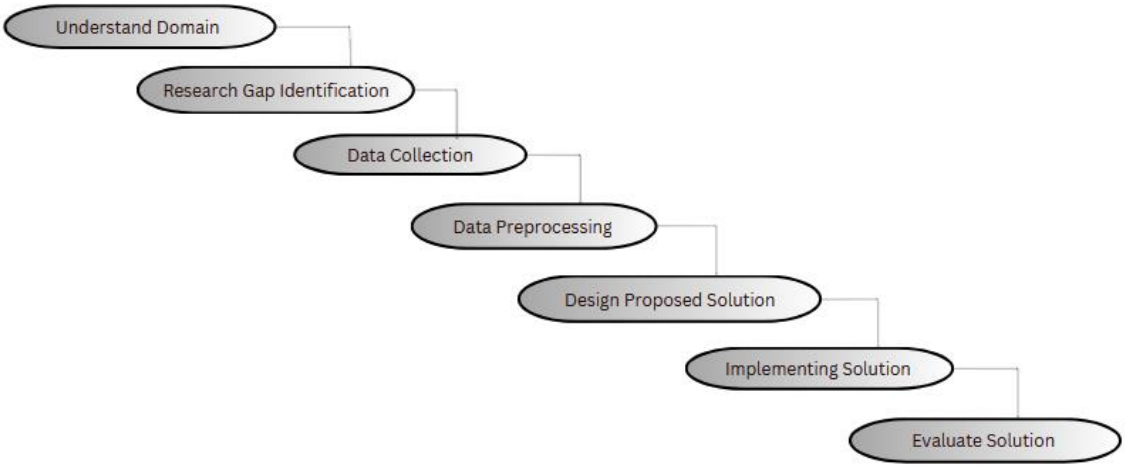


Figure 3-1: General Methodology of DDoS Attack Detection and Mitigation

- **Implementing a proposed solution**

  In this scheme, the training and testing of the model were conducted using collaborative tools such as Google Colab, while the simulation was performed using Mininet. Other software tools utilized for implementing the proposed solution included Python for coding, Scapy for packet manipulation, Mendeley Desktop for managing references, and Microsoft Office package for documentation. Additionally, hardware tools were employed as necessary to support the implementation process.

- **Analyzing and evaluating results**

  In this part, we showed the result of the proposed solution and evaluated the performance, accuracy, and other parameters.

3.1. Benchmark Dataset

The benchmark datasets utilized in this paper were curated by ((Elsayed, Le-Khac, & Jurcut, 2020b) and (Banitalebi Dehkordi et al., 2021)). The authors aimed to address the challenges arising from discrepancies among datasets, stemming from researchers individually annotating datasets to establish ground truth. Due to variations in annotation methods employed by researchers, resulting in diverse ground truths, and the common practice of not sharing annotated datasets, inconsistencies arise even when working with the same raw data. To address this issue, the researchers introduced a standardized way, encompassing mechanisms and algorithms, to annotate benchmarks and introduce different types of DDoS attacks into

pristine datasets using general tic tools. Even though we focused on the CICDDoS2019, we also incorporated an additional dataset, namely the InSDN data, to facilitate further comparison of different datasets based on our proposed solution.

For this research work, we used to exploit data sets and realistic data sets meaning, the data set was generated from an SDN environment while some others were generated from conventional networks (Taheri et al., 2023b). So, this data set helps to configure the differences and similarities of their effect on our proposed solution.

When an attacker overwhelms network resources like processing capacity, and bandwidth with a flood of false traffic from a single or multiple sources, where the SDN environment has a centralized control plane and data plane programmable this structure of the network can be manifested in many ways: centralized vulnerabilities, flow table exhaustion, traffic flooding, botnet attacks from this we could be doing classification and detection of volumetric and low-rate DDoS attacks. The attacker's main aim was to attack the control plane(M. N. Ali et al., 2023), which contains the network intelligence and routing decisions and is centralized in a controller. So, we focused on the attacks happening on the controller.

3.1.1. Dataset Source and Description

Datasets play a crucial role in creating suitable training and testing datasets for the intended solution. We used two data sets to train our models for comparison of different datasets in small and large-scale data and feature structures.

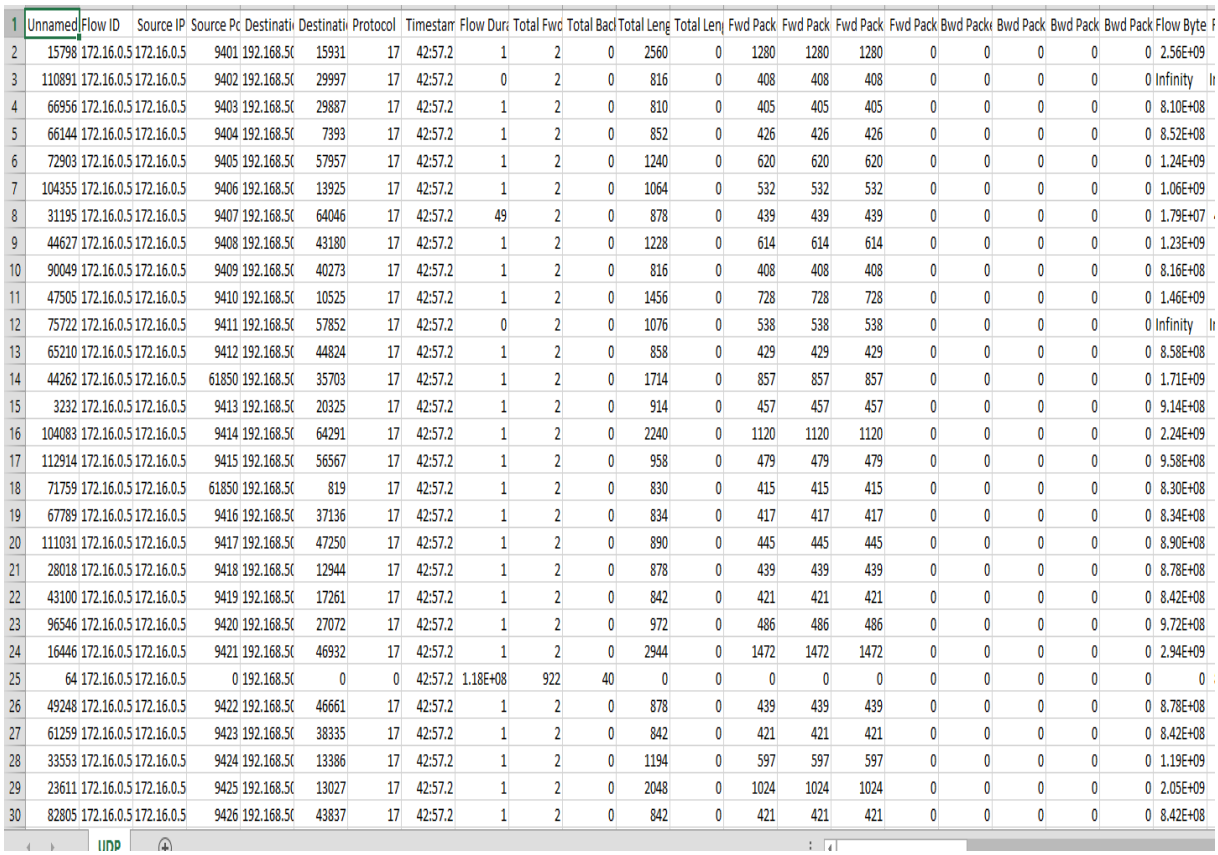
A. **InSDN dataset:** deploying detection systems faces a notable challenge due to the limited availability of up-to-date real-world datasets (Ahuja, Singal, Mukhopadhyay, et al., 2021). Privacy and legal concerns are the primary reasons for the scarcity of public datasets in the field of intrusion detection (Said Elsayed et al., 2020). In this study, we employed the InSDN dataset to evaluate our proposed deep learning model, aiming to mitigate the deficiencies present in existing datasets, particularly in the context of SDN. The data set consists of more than 80 features and has a total of 343,939 raw data. Among those data, 68,424 are normal traffic having more than 8 categories in their data. We used six of the top external attacks happening on the controller. In Figure 3-2 we have shown some of the raw data found in the dataset. (Availablehttp://iotseclab.ucd.ie/datasets/SDN)

1	Flow ID	Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp	Flow Dura	Tot Fwd P	Tot Bwd P	TotLen Fw	TotLen Bw	Fwd Pkt L	Fwd Pkt L	Fwd Pkt L	Fwd Pkt L	Bwd Pkt L	Bwd Pkt L	Bwd Pkt L	Bwd Pkt L	Flow Byts	Flow Pkts
2	172.17.0.2	200.175.2.	48856	172.17.0.2	80	6	12/1/2020 1:14	4045	2	5	476	1803	476	0	238	336.5828	1803	0	360.6	806.3261	563411.6	1730.532
3	172.17.0.2	200.175.2.	48856	172.17.0.2	80	6	12/1/2020 1:14	284	1	1	0	0	0	0	0	0	0	0	0	0	0	7042.254
4	172.17.0.2	200.175.2.	48858	172.17.0.2	80	6	12/1/2020 1:14	4302	2	5	478	1803	478	0	239	337.997	1803	0	360.6	806.3261	530218.5	1627.15
5	172.17.0.2	200.175.2.	48858	172.17.0.2	80	6	12/1/2020 1:14	83	1	1	0	0	0	0	0	0	0	0	0	0	0	24096.39
6	172.17.0.2	200.175.2.	48860	172.17.0.2	80	6	12/1/2020 1:14	3760	2	5	477	1803	477	0	238.5	337.2899	1803	0	360.6	806.3261	606383	1861.702
7	172.17.0.2	200.175.2.	48860	172.17.0.2	80	6	12/1/2020 1:14	87	1	1	0	0	0	0	0	0	0	0	0	0	0	22988.51
8	172.17.0.2	200.175.2.	48862	172.17.0.2	80	6	12/1/2020 1:14	9459	2	5	480	1803	480	0	240	339.4113	1803	0	360.6	806.3261	241357.4	740.0359
9	172.17.0.2	200.175.2.	48862	172.17.0.2	80	6	12/1/2020 1:14	154	1	1	0	0	0	0	0	0	0	0	0	0	0	12987.01
10	172.17.0.2	200.175.2.	48864	172.17.0.2	80	6	12/1/2020 1:14	4582	2	5	480	1803	480	0	240	339.4113	1803	0	360.6	806.3261	498254	1527.717
11	172.17.0.2	200.175.2.	48864	172.17.0.2	80	6	12/1/2020 1:14	761	1	1	0	0	0	0	0	0	0	0	0	0	0	2628.121
12	172.17.0.2	200.175.2.	48868	172.17.0.2	80	6	12/1/2020 1:14	6512	2	5	475	1803	475	0	237.5	335.8757	1803	0	360.6	806.3261	349815.7	1074.939
13	172.17.0.2	200.175.2.	48868	172.17.0.2	80	6	12/1/2020 1:14	78	1	1	0	0	0	0	0	0	0	0	0	0	0	25641.03
14	172.17.0.2	200.175.2.	48870	172.17.0.2	80	6	12/1/2020 1:14	4096	2	5	478	1803	478	0	239	337.997	1803	0	360.6	806.3261	556884.8	1708.984
15	172.17.0.2	200.175.2.	48870	172.17.0.2	80	6	12/1/2020 1:14	83	1	1	0	0	0	0	0	0	0	0	0	0	0	24096.39
16	172.17.0.2	200.175.2.	48872	172.17.0.2	80	6	12/1/2020 1:14	3613	2	5	486	1803	486	0	243	343.6539	1803	0	360.6	806.3261	633545.5	1937.448
17	172.17.0.2	200.175.2.	48872	172.17.0.2	80	6	12/1/2020 1:14	101	1	1	0	0	0	0	0	0	0	0	0	0	0	19801.98
18	172.17.0.2	200.175.2.	48874	172.17.0.2	80	6	12/1/2020 1:14	3649	2	5	478	1803	478	0	239	337.997	1803	0	360.6	806.3261	625102.8	1918.334
19	172.17.0.2	200.175.2.	48874	172.17.0.2	80	6	12/1/2020 1:14	160	1	1	0	0	0	0	0	0	0	0	0	0	0	12500
20	172.17.0.2	200.175.2.	48876	172.17.0.2	80	6	12/1/2020 1:14	4939	2	5	478	1803	478	0	239	337.997	1803	0	360.6	806.3261	461834.4	1417.291
21	172.17.0.2	200.175.2.	48876	172.17.0.2	80	6	12/1/2020 1:14	87	1	1	0	0	0	0	0	0	0	0	0	0	0	22988.51
22	172.17.0.2	200.175.2.	48878	172.17.0.2	80	6	12/1/2020 1:14	5230	2	5	479	1803	479	0	239.5	338.7041	1803	0	360.6	806.3261	436328.9	1338.432
23	172.17.0.2	200.175.2.	48878	172.17.0.2	80	6	12/1/2020 1:14	242	1	1	0	0	0	0	0	0	0	0	0	0	0	8264.463
24	172.17.0.2	200.175.2.	48882	172.17.0.2	80	6	12/1/2020 1:14	4618	2	5	480	1803	480	0	240	339.4113	1803	0	360.6	806.3261	494369.9	1515.808
25	172.17.0.2	200.175.2.	48882	172.17.0.2	80	6	12/1/2020 1:14	84	1	1	0	0	0	0	0	0	0	0	0	0	0	23809.52
26	172.17.0.2	200.175.2.	48884	172.17.0.2	80	6	12/1/2020 1:14	9832	2	5	479	1803	479	0	239.5	338.7041	1803	0	360.6	806.3261	232099.3	711.9609
27	172.17.0.2	200.175.2.	48884	172.17.0.2	80	6	12/1/2020 1:14	603	1	1	0	0	0	0	0	0	0	0	0	0	0	3316.75
28	172.17.0.2	200.175.2.	48886	172.17.0.2	80	6	12/1/2020 1:14	10011	2	5	478	1803	478	0	239	337.997	1803	0	360.6	806.3261	227849.4	699.2308
29	172.17.0.2	200.175.2.	48886	172.17.0.2	80	6	12/1/2020 1:14	273	1	1	0	0	0	0	0	0	0	0	0	0	0	7326.007

Figure 3-2: Raw Data Set of InSDN

B. **The CIC-DDoS2019 dataset:** specifically focuses on DDoS attacks, which represent a form of network security threat intending to overwhelm target networks with malicious traffic. Despite the proliferation of various machine learning techniques designed for the detection of DDoS attacks, a key challenge remains in minimizing processing overhead (Gebremeskel et al., 2023). This concern is emphasized by (Banitalebi Dehkordi et al., 2021) in their work from 2020. The data has over 88 features with 12 categories from these attack types, we considered four of them which were from the exploitation-based SYN, DrDoS, DrLDAP, and BEING. We used 1,245,402 from the total row data. This is due to the memory capacity to train and test this all data in our computers (Thakur et al., 2022). We attached the link and showed some raw data found in the data set in Figure 3-3.

(Availablehttp://205.174.165.80/CICDataset/CICDDoS2019/Dataset/PCAPs/01-12/).



	Unnamed	Flow ID	Source IP	Source Pc	Destinati	Destinati	Protocol	Timestam	Flow Dur	Total Fwd	Total Bacl	Total Leng	Total Len	Fwd Pack	Fwd Pack	Fwd Pack	Fwd Pack	Bwd Packe	Bwd Pack	Bwd Pack	Bwd Pack	Flow Byte
2	15798	172.16.0.5	172.16.0.5	9401	192.168.5(	15931	17	42:57.2	1	2	0	2560	0	1280	1280	1280	0	0	0	0	0	2.56E+09
3	110891	172.16.0.5	172.16.0.5	9402	192.168.5(	29997	17	42:57.2	0	2	0	816	0	408	408	408	0	0	0	0	0	Infinity
4	66956	172.16.0.5	172.16.0.5	9403	192.168.5(	29887	17	42:57.2	1	2	0	810	0	405	405	405	0	0	0	0	0	8.10E+08
5	66144	172.16.0.5	172.16.0.5	9404	192.168.5(	7393	17	42:57.2	1	2	0	852	0	426	426	426	0	0	0	0	0	8.52E+08
6	72903	172.16.0.5	172.16.0.5	9405	192.168.5(	57957	17	42:57.2	1	2	0	1240	0	620	620	620	0	0	0	0	0	1.24E+09
7	104355	172.16.0.5	172.16.0.5	9406	192.168.5(	13925	17	42:57.2	1	2	0	1064	0	532	532	532	0	0	0	0	0	1.06E+09
8	31195	172.16.0.5	172.16.0.5	9407	192.168.5(	64046	17	42:57.2	49	2	0	878	0	439	439	439	0	0	0	0	0	1.79E+07
9	44627	172.16.0.5	172.16.0.5	9408	192.168.5(	43180	17	42:57.2	1	2	0	1228	0	614	614	614	0	0	0	0	0	1.23E+09
10	90049	172.16.0.5	172.16.0.5	9409	192.168.5(	40273	17	42:57.2	1	2	0	816	0	408	408	408	0	0	0	0	0	8.16E+08
11	47505	172.16.0.5	172.16.0.5	9410	192.168.5(	10525	17	42:57.2	1	2	0	1456	0	728	728	728	0	0	0	0	0	1.46E+09
12	75722	172.16.0.5	172.16.0.5	9411	192.168.5(	57852	17	42:57.2	0	2	0	1076	0	538	538	538	0	0	0	0	0	Infinity
13	65210	172.16.0.5	172.16.0.5	9412	192.168.5(	44824	17	42:57.2	1	2	0	858	0	429	429	429	0	0	0	0	0	8.58E+08
14	44262	172.16.0.5	172.16.0.5	61850	192.168.5(	35703	17	42:57.2	1	2	0	1714	0	857	857	857	0	0	0	0	0	1.71E+09
15	3232	172.16.0.5	172.16.0.5	9413	192.168.5(	20325	17	42:57.2	1	2	0	914	0	457	457	457	0	0	0	0	0	9.14E+08
16	104083	172.16.0.5	172.16.0.5	9414	192.168.5(	64291	17	42:57.2	1	2	0	2240	0	1120	1120	1120	0	0	0	0	0	2.24E+09
17	112914	172.16.0.5	172.16.0.5	9415	192.168.5(	56567	17	42:57.2	1	2	0	958	0	479	479	479	0	0	0	0	0	9.58E+08
18	71759	172.16.0.5	172.16.0.5	61850	192.168.5(	819	17	42:57.2	1	2	0	830	0	415	415	415	0	0	0	0	0	8.30E+08
19	67789	172.16.0.5	172.16.0.5	9416	192.168.5(	37136	17	42:57.2	1	2	0	834	0	417	417	417	0	0	0	0	0	8.34E+08
20	111031	172.16.0.5	172.16.0.5	9417	192.168.5(	47250	17	42:57.2	1	2	0	890	0	445	445	445	0	0	0	0	0	8.90E+08
21	28018	172.16.0.5	172.16.0.5	9418	192.168.5(	12944	17	42:57.2	1	2	0	878	0	439	439	439	0	0	0	0	0	8.78E+08
22	43100	172.16.0.5	172.16.0.5	9419	192.168.5(	17261	17	42:57.2	1	2	0	842	0	421	421	421	0	0	0	0	0	8.42E+08
23	96546	172.16.0.5	172.16.0.5	9420	192.168.5(	27072	17	42:57.2	1	2	0	972	0	486	486	486	0	0	0	0	0	9.72E+08
24	16446	172.16.0.5	172.16.0.5	9421	192.168.5(	46932	17	42:57.2	1	2	0	2944	0	1472	1472	1472	0	0	0	0	0	2.94E+09
25	64	172.16.0.5	172.16.0.5	0	192.168.5(	0	0	42:57.2	1.18E+08	922	40	0	0	0	0	0	0	0	0	0	0	0
26	49248	172.16.0.5	172.16.0.5	9422	192.168.5(	46661	17	42:57.2	1	2	0	878	0	439	439	439	0	0	0	0	0	8.78E+08
27	61259	172.16.0.5	172.16.0.5	9423	192.168.5(	38335	17	42:57.2	1	2	0	842	0	421	421	421	0	0	0	0	0	8.42E+08
28	33553	172.16.0.5	172.16.0.5	9424	192.168.5(	13386	17	42:57.2	1	2	0	1194	0	597	597	597	0	0	0	0	0	1.19E+09
29	23611	172.16.0.5	172.16.0.5	9425	192.168.5(	13027	17	42:57.2	1	2	0	2048	0	1024	1024	1024	0	0	0	0	0	2.05E+09
30	82805	172.16.0.5	172.16.0.5	9426	192.168.5(	43837	17	42:57.2	1	2	0	842	0	421	421	421	0	0	0	0	0	8.42E+08

Figure 3-3: Sample CICDDoS2019 of Raw Datasets

3.2. Data Preprocessing Techniques

Data preparation, often termed as data preprocessing, involves cleaning, transforming, and structuring raw data into a format suitable for analysis. This phase is ceitical in any data analysis or machine/deep learning algorithm, as the quality and suitability of the data directly influence the accuracy and effectiveness of the analysis. In neural network modeling for intricate data analysis, data preparation holds particular importance, playing a pivotal role in determining the success of various complex data analysis tasks. We prepared the data directly to make it suitable for the training model. Utilizing CICFlowMeter, we converted over 88 features from the datasets into a .csv format. Before training the model, we implemented several preprocessing procedures on the data.

Common data preparation steps were done, which were removing duplication or irrelevant data, handling null values, integrating data, converting data, normalization of data, and lastly encoding categorical variables. We classified our data set into six types of attacks from the

InSDN dataset and four from CICDDoS2019. All of this was done to improve the performance of the DL model used for our research work.

The techniques we used on both datasets:

- **Cleaning Data:** one of the relevant challenges in datasets is dealing with missing values, often denoted by Nan value in the rows. To address this issue, various methods can be done such as removing the values or replacing values with designated values like mean or median. We processed by removing the Nan values.
- **Transforming Data:** is the process of converting the data into preferred type or format.
- **Data Integration:** To create a comprehensive dataset entails merging information from several sources into a single data frame. This is required in our situation since we collected our data from InSDN and CICDDoS2019, to train the models. The different data from several types of attacks must be combined.
- **Data Normalization**: entails setting a dataset's numerical feature scale to a uniform range. When analyzing or training machine learning models, the main objective is to bring disparate characteristics or variables to a common scale such that no feature dominates the others. By doing this, it is ensured that variables with bigger magnitudes are not favored in the model. When the features are varied in their scales or units of measurement, normalization is helpful since it can enhance the convergence and performance of some machine learning methods. Decimal scaling, Z-score normalization, and Min-Max scaling are examples of common normalization methods. For instance, Z-score normalization (standardization) is a scaling technique that transforms data to have a mean of 0 and a standard deviation of 1, whereas min-max scaling converts the values of a feature to a range between 0 and 1. These methods improve the consistency of the data and enable more effective model training and analysis. Decimal scaling divides each value by a power of ten and shifts the decimal point of each feature's values. The objective was to bring all the values into a predetermined range, usually between 0 and 1. To get a consistent scale, this method makes sure that the original distribution and relationships between values are preserved.

- **Data Labeling:** The label feature is transformed into categorical classification, with 0 and 1 assigned for effective categorization. In our case, we have 4 and 6 types in both data sets so we labeled it for the InSDN as: DoS is labeled as 1000, DDoS is labeled as 0100, BFA is labeled as 0010, and normal as 0001. And for the CICDDoS2019 BEGING is labeled as 1000, Dr_DoS_LDAP is labeled as 0100, Dr_DoS_NTP is labeled as 0010, and SYN is labeled as 0001.

3.3. Feature Selection

Feature selection is a critical process that involves choosing a subset of pertinent features from a larger set, as not all attributes are equally relevant for a specific predictive model. This practice helps remove irrelevant or redundant features, simplifying the model and enhancing interpretability. The selection method depends on the dataset's nature (Khaire & Dhanalakshmi, 2022), and utilized algorithms. Feature selection aids in reducing overfitting, enhancing accuracy, and interpretability, and lowering computational costs. Techniques include:

1. **Filter Methods:**
    - Correlation-based Feature Selection (CFS): is a technique that ranks features based on their correlation with the target variable as well as their inter-correlation. It utilizes an entropy-based metric to assess the relevance of the features (Nasir et al., 2020).
    - Chi-squared Test: it measures the dependence between features and the target variable using observed and expected frequencies. It selects features with the highest chi-squared statistics, indicating a strong relationship with the target variable (HUSSEİN & ÖZYURT, 2021).
    - Variance Threshold: removes features with low variance, as it assumes that low-variance features are less informative for the model.
2. **Wrapper Methods:**
    - Recursive Feature Elimination (Jeon & Oh, 2020): is an iterative feature selection technique that progressively removes the least important features until a stopping criterion is met, such as reaching a desired number of features or achieving the highest performance. RFE uses a machine learning algorithm to evaluate the importance of each feature and determine which ones to eliminate

in each iteration. The main advantage of RFE is its ability to identify the most important features by iteratively removing the least relevant ones, potentially capturing complex feature interactions. However, this comes at the cost of increased computational complexity, especially for large-scale problems. The effectiveness of RFE also depends on the choice of the underlying machine learning model and its ability to accurately assess feature importance.

3. **Embedded Methods:**
    - Lasso Regularization (Das et al., 2020): adds a penalty term to the model's objective function, encouraging sparsity and shrinking coefficients of irrelevant features to zero and selecting features with non-zero coefficients.
    - Decision Tree-based Methods (Umar et al., 2020): Uses decision trees to recursively split the feature space, selecting features that provide the most information gain or Gini impurity reduction during training.

In general, the main aim of feature selection is that all the attributes are not equally important for detecting DDoS attacks on SDN controllers and a larger number of data sets could reduce the performance and effectiveness of the model while training. We selected the chi-square method for the two datasets to get the most relevant features from the data set and to determine the highest features to detect and classify attacks on the controller. We got the 25 most relevant features listed in our datasets from the S-score and P-value indicating the importance relative to the target output including all features except low important ones.

The Chi-Square Feature Selection method is well-suited for DDoS attack detection with a large feature set (80 in this case) because it can effectively evaluate the importance of each feature, allowing us to identify the most discriminative and relevant variables for accurately detecting DDoS attacks. This dimensionality reduction helps mitigate the curse of dimensionality, improving the model's generalization performance and efficiency, while also providing a good starting point for exploring feature interactions using more advanced techniques. Moreover, the computational efficiency of the Chi-Square method makes it advantageous in time-sensitive DDoS detection scenarios, and the selected features can be seamlessly integrated with other mitigation strategies, such as the backup controller-based approach, to enhance the overall DDoS defense system and considering categorical attacks.

No.	Attribute Name	No.	Attribute Name
1	Protocol	25	Fwd-IAT-Min
2	Flow-duration	26	Bwd-IAT-Tot
3	Tot-Fwd-Pkts	27	Bwd-IAT-Mean
4	Tot-Bwd-Pkts	28	Bwd-IAT-Std
5	TotLen-Fwd-Pkts	29	Bwd-IAT-Max
6	TotLen-Bwd-Pkts	30	Bwd-IAT-Min
7	Fwd-Pkt-Len-Max	31	Fwd-Header-Len
8	Fwd-Pkt-Len-Min	32	Bwd-Header-Len
9	Fwd-Pkt-Len-Mean	33	Fwd-Pkts/s
10	Fwd-Pkt-Len-Std	34	Bwd-Pkts/s
11	Bwd-Pkt-Len-Max	35	Pkt-Len-Min
12	Bwd-Pkt-Len-Min	36	Pkt-Len-Max
13	Bwd-Pkt-Len-Mean	37	Pkt-Len-Mean
14	Bwd-Pkt-Len-Std	38	Pkt-Len-Std
15	Flow-Byts/s	39	Pkt-Len-Var
16	Flow-Pkts/s	40	Pkt-Size-Avg
17	Flow-IAT-Mean	41	Active-Mean
18	Flow-IAT-Std	41	Active-Std
19	Flow-IAT-Max	43	Active-Max
20	Flow-IAT-Min	44	Active-Min
21	Fwd-IAT-Tot	45	Idle-Mean
22	Fwd-IAT-Mean	46	Idle-Std
23	Fwd-IAT-Std	47	Idle-Max
24	Fwd-IAT-Max	48	Idle-Min

Figure 3-4: Some Features of Datasets

3.4. Model Selection

The objective of model selection was to find the most suitable model for effectively analyzing complex patterns in DDoS attacks on SDN controller network packet flows, specifically for detecting and classifying DDoS attacks in SDN. The first factor considered in model selection is performance optimization and scalability, ensuring the ability to generalize new data and handle large volumes of data with an optimized running time. These criteria helped us evaluate and choose potential models for our task. The second factor is capturing sequential patterns in the row data. Third is the flexibility and generalization of features in the datasets we have. The fourth one is the sequential flow of packets in the SDN controller to capture temporal dependencies in the flow patterns. The fifth is considering computational efficiency meaning the ability to work on real-time datasets. We summarized the comparisons of the models in Table 3-1.

The 1D CNN: is a powerful tool for detecting patterns in sequential data, such as network traffic. It can automatically extract relevant features from the input data, such as packet size,

frequency, and direction, and capture both short-term and long-term patterns. This is especially useful for detecting volumetric and low-rate DDoS attacks, which can be difficult to detect using traditional methods (K. Singh et al., 2021).

LSTM: networks are likely to be highly effective for identifying anomalous behavior in DDoS network flows due to their ability to capture intricate temporal patterns. Its networks are capable of capturing long-term and short-term dependencies in sequential data (Yeom et al., 2022). They can store information over extended periods, making them well-suited for detecting anomalies.

MLP: MLPs can be useful for certain classification tasks, including identifying anomalous behavior in network flow activities (Najar & Manohar Naik, 2022). However, non-linear relationships can be modeled, they can handle tabular data. It has lower computation complexity and is suited for classification target labels where these features make it more preferred for our work.

RNNs can be effective for capturing simple temporal patterns, but they may not be as effective at capturing complex long-term dependencies (Sumathi et al., 2022), which are crucial for detecting sophisticated DoS attacks.

Hybrid Model: Overall, the development of a hybrid deep learning model is a promising approach to enhance the detection and classification of DDoS attacks in SDN environments, where the network traffic data exhibits complex and evolving patterns.

The key advantages of using a hybrid deep learning model in this research are:

- **Improved Performance:** The hybrid model can leverage the strengths of multiple deep learning techniques to achieve higher accuracy, robustness, and generalization for DDoS attack detection and classification.
- **Handling Complex Patterns:** Hybrid models can capture the intricate and varying patterns in network traffic data more effectively than single deep learning models, leading to enhanced DDoS attack detection capabilities.

- **Leveraging Complementary Strengths:** By combining deep learning architectures like CNNs and RNNs, the hybrid model can leverage their unique strengths in learning spatial, temporal, and contextual features, creating a more comprehensive solution.
- **Improved Generalization:** The integration of multiple deep learning techniques allows the hybrid model to learn more robust and generalizable representations of DDoS attack patterns, improving its performance on unseen or new types of attacks.
- **Addressing Limitations of Individual Models:** The hybrid approach can help mitigate the limitations of individual deep learning models by combining them in a way that compensates for their weaknesses, leading to a more reliable and effective DDoS detection system.

Combining 1D CNN with LSTM: In this context, the 1D CNN component is used to extract spatial features from the input data along the temporal dimension. The CNN layer applies filters to the input sequence to capture patterns and relationships within the data (K. Singh et al., 2021). This can help in identifying relevant features and reducing the dimensionality of the input data. The LSTM layer then processes these features to capture long dependencies and temporal patterns, enhancing the model's ability to detect sophisticated DDoS attack patterns over time.

Combining 1D CNN with MLP: Combining a 1D CNN with an MLP can be beneficial for tasks such as sequence classification, sentiment analysis, and time series forecasting (Roopak et al., 2020), where capturing both local and global patterns in the input data is important for accurate predictions. Where MLP integrates these features to perform robust classification, resulting in improved detection accuracy which makes it adaptable to diverse attack types and enables effective classification of known and unknown DDoS attack signatures.

Combining 1D CNN with RNN: By combining the feature extraction capabilities of the 1D CNN with the sequence modeling capabilities of the RNN, the hybrid model can effectively capture both spatial and temporal patterns in the input data. This allows the model to learn complex patterns and dependencies present in sequential data.

Table 3-1: Comparison of DL Models

Categories	LSTM	RNN	MLP	CNN
Input and output structure	Sizes of input and output that result might differ	Sizes of input and output that result might differ	Input size is fixed to features and output size is determined by output of the neurons which is number of classes in classifications	Both input and output sizes are fixed
Model architecture	Contains a memory cell input gate, output gate, and forget gate can capture long-term dependencies	Simple recurrent layers struggle with long-term dependencies due to vanishing gradient problems	Fully connected layers	Convolutional layers and pooling layers.
Data preprocessing	May require significant memory	Require less memory	Depends on network size	May Require significant memory.
Suitable usage	When long-term dependencies are important	When short-term dependencies are adequate	The general purpose more for classification	Spatial relationships in data

3.5. Training and Testing Models

After preprocessing data, selecting features, splitting of the data is done and a DL model supporting the recommended input format is built using the Keras framework. Shapes from the input layer's, regularization, recurrent, activation, and dense layers are used to build this model. The model is then created and goes through compilation. The model defines the loss function,

optimizers, and metrics throughout this compilation stage. The loss function is defined and bound at this point. The evaluation measures are also defined in advance of the next round of model training.

3.6. Evaluation Methods

The proposed method's performance is evaluated using the following:

A. **Confusion matrix** is used in ML or DL to evaluate the classification model's performance. It presents an overview of the model's predictions and makes a comparison between them and the actual findings by dividing predictions into four categories: TP, TN, FP, and FN shown in Table 3-2. The matrix offers a thorough understanding of the model's performance.

Here are the components of a confusion matrix:

- TP: Instances where the model correctly predicts the positive class.
- TN: Instances where the model correctly predicts the negative class.
- FP: Instances where the model incorrectly predicts the positive class
- FN: Instances where the model incorrectly predicts the negative class

Table 3-2: Confusion Matrix

	Actual positive	Actual negative
Predictive positive	TP	FP
Predictive negative	FN	TN

B. **Accuracy:** Is determined by calculating the percentage of queries for which the model correctly predicted the label, as well as the total number of data samples that were successfully classified.

$$accuracy = \frac{(TP + TN)}{TP + TN + FP + FN}$$

Equation 3-1: Accuracy Formula

C. **Precision:** Divide the total positives by the total positives to determine how accurate the correct forecasts were.

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

Equation 3-2: Precision Formula

D. **Recall:** Recall is a measure of how comprehensive a classifier is. A low recall indicates a high number of False Negatives. The recall is a metric that indicates how well our model finds True Positives.

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

Equation 3-3: Recall Formula

E. **F1-measure:** The F1 score is derived from the combination of precision and recall concerning a certain positive class.

$$f1 - measure = 2 * \frac{precision * recal}{precission + recall}$$

Equation 3-4: F1-Measure Formula

3.7. Design and Development Tools

3.7.1. Design Tools

In this thesis, design tools played a crucial role in shaping and illustrating the proposed models. Two specific tools, EdrawMax and Canvas, were employed to craft diagrams, flowcharts, and the overall architecture of the proposed solution.

EdrawMax: is a versatile diagramming tool that allows users to create a wide variety of diagrams, such as flowcharts, mind maps, organizational charts, and network diagrams. The software is particularly well-suited for professionals and businesses that require detailed and technical diagrams. EdrawMax, the diagramming software, offers an extensive library of templates and symbols, enabling users to customize their diagrams to meet their specific requirements. The software's interface can be adapted to suit the user's needs, making it a flexible and user-friendly solution for diagramming tasks.

Canvas: is a graphic design software that focuses on creating visually compelling designs for social media, marketing, and various other applications. The software provides users with a vast collection of templates, images, fonts, and design elements, enabling them to generate

professional-looking designs with ease, even without extensive design expertise. Canvas is particularly well-suited for individuals, small businesses, and marketers who aim to create engaging designs quickly and efficiently.

3.7.2.Development Tools

3.7.2.1.Software Tools Used

- **Mininet:** To create a virtual network of hosts, switches, and controllers, utilize the network emulator Mininet 2.2.0. Open Flow is provided by mininet switches for highly flexible custom forwarding and SDN, and Linux network software is used by mininet hosts.
- **POX Controller:** POX is an open-source OpenFlow SDN controller with a Python foundation. Its purpose is to facilitate the rapid development and prototyping of novel network applications. With the POX controller, basic Open Flow devices can be transformed into hubs, switches, load balancers, and firewalls. It is well-liked for academic and scientific uses. It provides a versatile environment for testing out different network setups and methods. POX continues to be a leading option for SDN development and learning, backed by a vibrant community.
- **Python:** The reactive scripting language Python 3.9 is meant to be easy to learn. In this research, we applied deep learning techniques to create the model and configure the Pox controller.
- **Google Colab:** Colab is a cloud-based, free Jupyter notebook environment. The primary advantage is that no setup is required. Notebooks can be created, uploaded, and shared. Google Drive notebooks can be imported and exported. It is possible to import and publish GitHub notebooks.
- **Scapy:** Scapy is a very powerful tool that may be used for packet synthesis, scanning, sniffing, attacking, and forging.
- **Keras:** Keras is a compact, easy-to-learn high-level Python deep learning toolkit that can be used on top of TensorFlow. It is simpler to build and more user-friendly. It can also be used with a GPU or CPU.
- **Mendeley Desktop:** is an easy-to-use reference management program that functions as a social network to help scholars locate pertinent literature.

- **Microsoft Office Packages:** These are used to write the research paper in the appropriate style.

3.7.2.2. Hardware Tool

For our research to be efficient, also consider scalability in datasets and model complexity and adaptability, we need to use the following hardware tools:

- Hard disk: used for storing our datasets, programs, and documents. The dataset we upload for our work is around 6.89 GB only for the dataset.
- GPU: is used for speeding up the training time and handle parallel processing tasks efficiently. We used Google Colab to access the GPU.
- RAM: is used with GPU to speed up the training time. We installed RAM of 8GB.

CHAPTER FOUR

PROPOSED SOLUTION ARCHITECTURE

4. Chapter Overview

We presented a solution to the current problem in this chapter. The study's main objective is to enhance the detection and mitigation of DDoS attacks on SDN controllers using hybrid deep learning models such as RNN, LSTM, and MLP with 1D CNN. Metrics including accuracy, precision, recall, F1 score, confusion matrix, and Roc Curve were used to assess the model's performance offering a solution to the problem that had been found an essential part of every research project. This suggested remedy was based on an improved algorithm that gives priority to the best method for identifying and mitigating DDoS attacks on SDN controllers.

4.1. Proposed Model Architecture

The suggested model for accurately and effectively categorizing the attacks is a hybrid deep learning. Three-level detection is utilized for network traffic to guarantee improved accuracy and decreased processing complexity simultaneously. The first part was to detect attacks and second was to classify the types of attacks and lastly to mitigate those attacks.

One of the gaps we concluded from the literature review was that most of the work done on the detection part of DDoS attacks was on a small scale of the data set. Our proposed system enhanced the detection and mitigated the attacks after using large-scale and small data with efficient timing and resources with feature selection of the chi-square method.

Most of the works done so far haven’t used feature selection on their data which causes less detection performance and high training time, however, we considered using chi feature selection where 25 of the most relevant features were used to classify the attacks.

Mitigation was not mostly done in many of the papers, while in our work we also considered mitigating the attacks after being classified by our hybrid model. We made an assumption to carry out our work by considering 3 POX controllers, 8 switches, and 30 hosts simulated on the Mininet with the help of Miniedit and installed all the things needed and considered having two of the hosts to be the attackers causing the DDoS on the controller and mitigate the attack from

the normal traffic by dropping the attacks traffics. In case of a bottleneck or failure of the main controller, we have backup controllers to solve this.

The proposed architecture hybrid model works in three phases including the mitigation part. So first, we did the detection of the DDoS attacks by considering the two datasets for better results using feature selection along with the CNN model. The second phase was, that the attacks had been classified further into six and four types using another model.

While there are many different kinds of deep learning, we selected three models MLP, LSTM, and RNN that are frequently utilized in cyber More than 80 feature elements are included in the CSV file format such as ‘Protocol’, ‘Duration’, and ‘Byte’ security applications because of their functionality to detect and classify attacks even in handling and categorizing enormous amounts of data. They were used, for instance, to identify hostile nodes based on their behavior, detect anomalies or intrusions in network traffic, and carry out other security-related tasks.

The suggested model was composed of 6 primary phases. The first was the datasets should be loaded into our environment, the second was to preprocess the data suitably. In the third, we applied chi-square feature selection to select relevant features used for our work, and the fourth part, splitting of our data to training, and testing. Fifth was evaluating data which classifies well enough and finally, mitigation was done. The preprocessed dataset is divided into two parts: one is needed for model training to learn the pattern of the DDoS data, and the other is tested for model evaluation to assess the model's performance on unseen labeled data. Data preprocessing techniques were carried out before training the proposed model, and pertinent features were accessed.

A hybrid model was used in both data for classification purposes where 1D CNN combined with LSTM, MLP, or RNN to get the best performance regarding scalability, running time, and performance for better accuracy or precision. We used other metrics to measure our hybrid model like the ROC curve and confusion matrices. We showed the architecture model of our proposed model below in Figure 4-1.

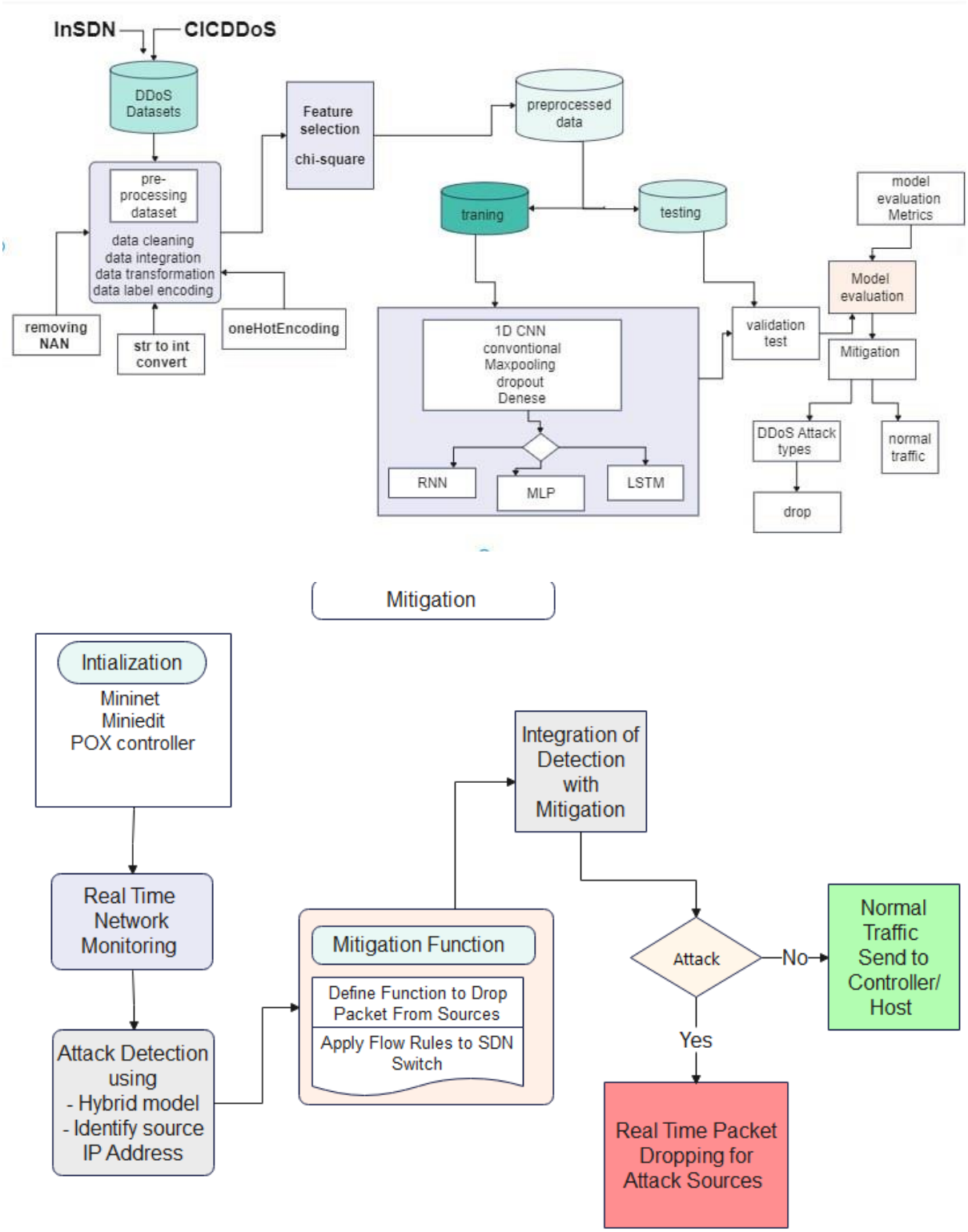


Figure 4-1: Proposed Model Architecture for Detection and Mitigation of DDoS Attacks on SDN Controller

4.2. Proposed Hybrid Deep Learning CNN-MLP Algorithm

To enhance the performance of a machine learning model, hybrid deep learning models combine two or more deep learning models or architectures. By combining the models, we used the advantages of both models, where these models have the potential to outperform those that rely solely on one deep learning technique or machine learning. We wrote a common algorithm used in building our hybrid model below:

```
Begin
    # Create a sequential model
    model = Sequential()
    # Add a 1D convolutional layer
     model.add(Conv1D(filters=32, kernel_size=3, activation='relu',
    input_shape=(sequence_length, num_features)))
    model.add(MaxPooling1D(pool_size=2))
    # Flatten the output from the CNN
     model.add(Flatten())
    # Add MLP layers
    model.add(Dense(units=128, activation='relu'))
    model.add(Dense(units=64, activation='relu'))
    # Output layer for categorical classification
     model.add(Dense(num_classes, activation='softmax'))
    # Compile the combined model
     model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
    # Train the model
    model.fit(X_train, y_train, epochs=20, batch_size=64, validation_data=(X_val, y_val),
callbacks=[EarlyStopping(monitor='val_loss', patience=5)], verbose=1)
    # Predict on test data
    y_pred = model.predict(X_test)
```

```
    detected_labels = {flow_id: class_labels[np.argmax(pred)] for flow_id, pred in
zip(test_flow_ids, y_pred)}
    return detected_labels
End
```

4.3. Hyper-parameter Optimization

Hyper-parameter optimization is essential in deep learning to find the best parameter values for optimal performance. Techniques like grid search, random search, Bayesian optimization, and gradient-based optimization were used to search for the most effective hyper-parameters. Cross-validation, regularization, and early stopping were commonly employed to ensure the model generalizes well. We used the following in our model.

- Random Search: Random search randomly samples hyper-parameter values from predefined ranges and evaluates the model performance for each random combination. Compared to grid search, random search is more efficient and can often find better hyper-parameter values in a shorter amount of time. So, we have used this search method for our research.
- Early Stopping: Early stopping is a technique used during model training to stop the training process once the model's performance on a validation set starts to deteriorate. It helps prevent overfitting and allows for finding an optimal number of training iterations before performance starts to degrade. We used 5 patience for our hybrid model.
- Dropout is an effective technique, used for reducing overfitting, it involves randomly selecting cells with layers and changing those cell output values to 0. We have used different dropouts for our model like 0.2, 0.05, and 0.5.

4.3.1. Activation Function

In neural networks, activation functions are mathematical procedures that add non-linearity to the model. They aid in the discovery of intricate patterns in data. Sigmoid, Tanh, ReLU, Leaky ReLU, PReLU, and ELU are examples of common types.

ReLU and its variations, such as Leaky ReLU, are frequently chosen for deep neural network training because of their efficiency and simplicity. They lessen problems that other activation functions have, such as the dying ReLU problem and the vanishing gradient problem.

Coming to our work **ReLU** would be a good option for activation functions in our CNN-MLP on DDoS attacks on SDN controllers, where robustness and computing efficiency are important considerations. To properly detect and mitigate DDoS attacks, they provide quick training and efficient learning of intricate patterns in network traffic data.

4.3.2. Loss Function

The loss function is the difference between the actual ground truth values in the training data and the predicted values of a machine learning model, which is also referred to as a cost function or objective function. Minimizing it is the aim of DL model training since it enhances the model's capacity to produce precise predictions on unobserved data.

Common types of loss functions vary depending on the task being performed, such as regression, classification, or other specialized tasks. Some widely used loss functions include Mean squared error, binary cross-entropy loss, categorical cross-entropy loss, hinge loss, and others.

As we discussed earlier, we mentioned that we have to classify the attacks so, we did our CNN-MLP model by categorical cross-entropy loss function to classify the different attack types.

4.3.3. Optimizer

In machine learning, an optimizer is a crucial algorithm used to minimize the loss function during model training by adjusting the model parameters. There are various optimizers available, each with unique characteristics and performance.

Among commonly used optimizers, Adam stands out as a strong candidate for detecting DDoS attacks. Adam combines the benefits of momentum and adaptive learning rates, making it well-suited for a wide range of detecting and classifying DDoS attacks on the SDN controller. It adapts learning rates for individual parameters based on their gradients' first and second moments, allowing for efficient training even with large-scale and complex datasets.

Additionally, Adam tends to converge quickly and performs well in practice across various applications, making it a reliable choice for DDoS attack detection on SDN controllers.

4.4.Proposed CNN-MLP Hybrid Model

For our dataset we used two data sets InSDN and CICDDoS2019 in our experiments. Where the InSDN includes 7 categories of attacks including normal traffic while CICDDoS2019 includes 12. Both have .csv files which we used for our work and the COCDDoS2019 datasets have huge sizes of more than 20 GB. So, we took 4 and 6 types of attacks respectively from their categories because the data were huge and included exploit attacks for our works. We considered 76 and 81 features and 300,000 rows from each attack before feature selection which gives us around 1,100,000 data from CICDDoS2019 and 207,146 from InSDN where a total of 6.89GB were used. After applying chi-square, we conducted our model training by 25 features. Overall, by using this dataset, we were able to concentrate on the difficulties associated with identifying volumetric DDoS assaults in SDN setups, which has given us important information for further research in this field.

We preprocessed the dataset for better performance and reduced overfitting and make it more suitable for training our deep learning. DL models like LSTM, MLP, and RNN are used in hybrid form with 1D CNN network in our proposed model for detecting and classifying DDoS attacks. With all the interaction of the layers and hyper-parameters, the 1D CNN with MLP has shown a robust framework of performance needed. For our work, we first need to define our parameters:

Batch size: defines how much data the model processes before changing the weight in our case we considered 300,800 and 1000.

Number of epochs: how many times the model iterates through the full training dataset during training time, in our case, we used 27, 40, and 100.

The activation function of a neuron adds non-linearity to its output, which helps neural networks recognize intricate patterns. We used relu in all our models.
Loss Function: Indicates how well the model is performing and provides guidance during training by quantifying the difference between the expected and actual values. We have used categorical-cross entropy for our models because we have four types of attacks to classify.

Optimizer: An algorithm that uses the data, loss function, and backpropagation gradient to update the network weights iteratively, to reduce loss and we used Adam optimizer. **Metrics:** Used to assess how well the model performs during testing and training, giving information on how effectively the model is accomplishing its goal. Mostly we considered accuracy to measure our models.

A proposed general summary of DDoS classification Hybrid CNN-MLP DL model

- Collect the datasets
- Reading the datasets
- Combining the different types of attacks in one CSV file
- Pre-processing the data
- Applying feature selection
- Applying train test split
- Creating a hybrid model
- Compiling the hybrid model
- It starts by giving the input data into the CNN model
- By using this hyper-parameter, we constructed our hybrid model as the following Figure 4-2:

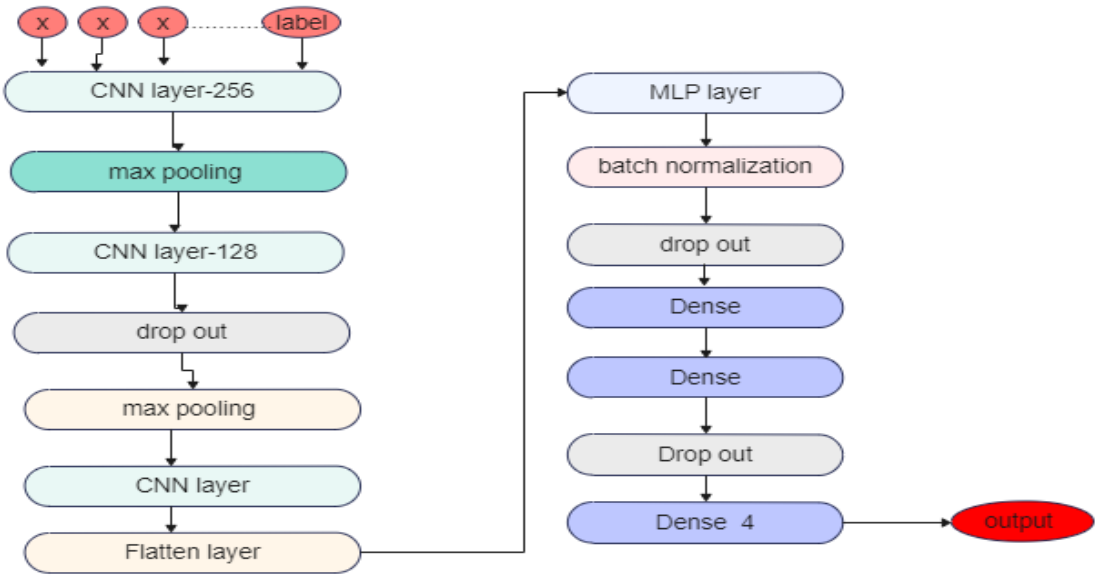


Figure 4-2: CNN-MLP Model

CHAPTER FIVE

Implementation of the Hybrid Model Solution

5. Chapter Overview

In this part we discussed the environment setup we used and additionally, the tools we used for our experiments to detect and classify attacks. We also added some implementation codes until the training process and the experiments were done to conduct our research work for the detection and mitigation of DDoS attacks on SDN controllers.

5.1. Environment Setup

The environmental setup is where we set our tools packages and deployment for our data to continue doing our research or to develop the proposed system to properly be trained, evaluated, and tested for better results. Many sets of tools, software, and others have been used to implement our work.

5.2. Tools and Packages

- **Python 3.9**- it's a programming language used for data analysis, pre-processing, training, and testing the deep learning model.
- **Jupiter notebook**: is a web-based interactive computing Lang form that allows to create and share documents.
- **Tensorflow:** is a machine learning framework for building or deploying models.
- **Pandas:** is a Python library for data manipulation and analysis.
- **NumPy:** is a Python library used for handling large numerical for computing.
- **Matplotlib:** is a Python library used for creating visualizations in various formats.
- **Scikit-learn:** is a machine learning library in Python that provides simple and efficient tools for data mining and data analysis.
- **Mininet:** is an emulator used for doing our experiment for doing our network topology. We have done our SDN network on it.
- **POX controller:** it's an essential component of mininet configuration which is an open-source SDN controller based on Python that enables open-flow protocol communication between switches and controllers.

5.3. Deployment of the Proposed Solution

Before training our models, we need to import Python libraries which our required during our model training, and need to load and integrate our CSV files from Google Drive to Google Colab. We did our training first by importing libraries, loading, and reading the data from Colab.

5.4. Dataset Pre-processing

Before the model training, we have done some processing steps for the datasets. We have done the following:

5.4.1. Removing Socket and Unnamed Features

From the 82 features in the InSDN and 88 in the CICDDoS2019 datasets, we removed socket features like unnamed, src IP, des ip, and port because these features differed from network to network shown in the appendix, and for not the deep learning to be biased to socket data because some users may share IP address and also avoid overfitting problems. After removing, we obtained 76 and 81 features respectively from our two datasets.

5.4.2. Cleaning Dataset

We have cleaned all of the missing values 163320 in CICDDoS2019 data, while InSDN has cleaned the data itself shown in the appendix.

5.4.3. Standardization of Datasets

Is the setting of the dataset's numerical feature scale to a uniform range. When analyzing or training deep learning models, the main objective was to bring disparate characteristics or variables to a common scale such that no feature dominates the others. We have used a MinMax scaler for the InSDN dataset and Standardscaler for the CICDDoS2019 for better performance. MinMax scaler was used to scale the feature's magnitude within the given range. We have shown it below in Figure 5-1.

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
```

```
scaler.fit(x)
scaled_train = scaler.transform(x)
scaled_data_fit_transform = scaler.fit_transform(x)
```

```
print(scaled_data_fit_transform)
```

```
[[3.49916238e-05 2.19599714e-06 1.17326137e-04 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [3.64999553e-06 1.09799857e-06 0.00000000e+00 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [3.71332878e-05 2.19599714e-06 1.17326137e-04 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 ...
 [3.23749603e-05 1.09799857e-06 5.86630687e-05 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [1.72874788e-04 0.00000000e+00 2.93315343e-05 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [2.66083007e-05 1.09799857e-06 5.86630687e-05 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]]
```

Figure 5-1: Standardization of Dataset

5.4.4. Data Labeling

For our model to understand the attack types, we need to label the attack types into 1's and 0's by categorical classification because deep learning handles float or numerical values. Our data types InSDN4dataset are BFA (1, 0, 0, 0), and DDoS (0, 1, 0, 0), DoS(0,0,1,0), normal(0,0,0,1). For CICDDoS2019 we countered 4(SYN, Normal, UDP, DrDoS) attacks from 12 categories they have due to the capacity of our computer cause the dataset is huge. So, by using the OneH otEncoder we labeled our data shown in Figure 5-2.

```
# Define the categorical labels
#categories = [ 'Probe', 'BFA', 'Web-Attack', 'BOTNET','Normal']
categories = np.unique(y)
# Create an instance of the OneHotEncoder
encoder = OneHotEncoder(categories=[categories],sparse=False)

# Define the input labels
labels = [ 'DoS','DDoS', 'Probe', 'BFA', 'Web-Attack', 'BOTNET','Normal']

# Convert the categorical labels to numerical data using one-hot encoding
encoded_labels = encoder.fit_transform(np.array(Y).reshape(-1, 1))

# Print the encoded labels
for label, encoded_label in zip(Y, encoded_labels):
    print(label, encoded_label)
```

Figure 5-2: Data Labeling

5.5. Feature selection

In our research work, we have more than 75 features in both our DDoS datasets. So, if we consider all the features to do the training it takes a long time for the models, and some features

are not that relevant to detect and classify DDoS attacks. (Khempetch & Wuttidittachotti, 2021) stated that the best features can be selected by using PCA, decision tree, and chi-square tests. We tested the PCA and chi-square to examine the frequencies of specified classes and turned out that the chi-square has more effect than PCA and continued our research by Chi-squared test. The chi-square is flexible in handling huge datasets and assessing the independence or dependence of the variables and has the best performance concerning data distribution. The Chi-square test is a statistical approach for identifying significant relationships between categorical data. It examines if the observed frequencies of variable co-occurrences deviate significantly from the anticipated rates if the variables were independent.

This test was straightforward, computationally efficient, and best suited for datasets containing categorical data. It detects relevant properties for forecasting target variables, hence increasing model efficiency. Furthermore, its resilience to outliers provides accurate association discovery even in noisy datasets. Overall, the Chi-square test is a useful tool for feature selection and association analysis in categorical datasets because of its simplicity, efficiency, and ability to detect significant correlations.

We have applied the chi-square, to our dataset to choose the relevant features that were correlated with independent data (Gebremeskel et al., 2023). To start our learning model, we have used 25 features from the original DDoS dataset we have shown in Figure 5-3.

Top 25 Features:

	Feature	Score
36	Bwd Pkts/s	26142.169728
14	Flow Pkts/s	26111.314741
76	Unix_Timestamp	13301.296635
50	Down/Up Ratio	13259.186391
51	Pkt Size Avg	10211.142282
39	Pkt Len Mean	10193.576986
40	Pkt Len Std	10019.896375
38	Pkt Len Max	9772.272677
46	ACK Flag Cnt	9588.394586
12	Bwd Pkt Len Std	9373.713580
9	Bwd Pkt Len Max	9231.263018
11	Bwd Pkt Len Mean	9143.323238
53	Bwd Seg Size Avg	9143.323238
4	TotLen Bwd Pkts	8873.082569
63	Subflow Bwd Byts	8873.082569
41	Pkt Len Var	8610.535312
43	SYN Flag Cnt	8399.281567
8	Fwd Pkt Len Std	7730.488425
7	Fwd Pkt Len Mean	7708.244003
52	Fwd Seg Size Avg	7708.244003
42	FIN Flag Cnt	5923.586334
5	Fwd Pkt Len Max	5894.742493
13	Flow Byts/s	5611.914760
32	Bwd URG Flags	3042.210764
47	URG Flag Cnt	3042.210764

Figure 5-3: Selected Features

DDoS attacks mainly involve flooding the target with enormous amounts of traffic, which causes the average packet size to be higher than usual. By monitoring the above-listed feature's behavior like departures from the typical traffic patterns, average packet size monitoring aids in the detection of these types of assaults. We can see some of the selected feature descriptions in Table 5-1.

Table 5-1: Selected Features

No.	Features	Description
1	Min Packet Length	Is the minimum size of the packet in a flow
2	ACK Flag Count	Is the number of packets with the ack flag set in a flow?
3	Flow Bytes/s	Is the average number of bytes transferred per second in a network flow.
4	Fwd. Packet Length Min	It represents the minimum size of forwarding packets in a flow.
5	Average Packet Size	Is the average size of packets in a flow.
6	URG Flag Count	Is the number of packets with an urgent flag set in a flow.
7	TimestampNanosec	Represents timestamp of network flow in nanoseconds
8	AvgFwd Segment Size	Is the average size of forward segments in a flow.
9	act_data_pkt_fwd	Represents the number of actual data packets forwarded in a flow.
10	Protocol	Represents the network protocol used in flow such as TCP, UDP, or ICMP

5.6. Model Building

To enhance the performance of a DL model, a hybrid DL model combines two or more DL models or architectures to help more. Due to their ability to integrate the strengths of multiple deep learning techniques, these models have the potential to outperform those that rely solely on one. We have built three deep-learning models and three hybrid models to do our research.

5.6.1. LSTM

LSTMs can identify DDoS attacks by continuously examining network traffic data. They are good at spotting patterns suggestive of DDoS attacks because they are good at capturing long-term dependencies in sequential data. LSTMs are capable of real-time DDoS attack detection due to their ability to identify anomalous traffic patterns and departures from typical behavior (Khempetch& Wuttidittachotti, 2021). LSTMs can distinguish between malicious and

legitimate incoming traffic by training on past network traffic data, which lessens the effect of DDoS attacks. Figure 5-4, shows the training of our model.

```
# LSTM model
lstm_model = Sequential()
lstm_model.add(LSTM(64,activation='relu', input_shape=(W.shape[1], 1)))
lstm_model.add(Dense(64,activation='relu'))
lstm_model.add(Dropout(0.4))
lstm_model.add(Dense(32,activation='relu'))
lstm_model.add(Dropout(0.2))
lstm_model.add(Dense(7, activation='softmax'))  # num_classes is the number of output classes
learning_rate = 0.01
optimizer = Adam(learning_rate=learning_rate)
lstm_model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
lstm_model.fit(X_train, Y_train, validation_data=(X_test,Y_test), epochs=50, batch_size=500, callbacks=[early_stopping])
```

```
266/266 [==============================] - 28s 107ms/step - loss: 0.0741 - accuracy: 0.9737
Epoch 23/30
266/266 [==============================] - 27s 103ms/step - loss: 0.0744 - accuracy: 0.9733 - val_loss: 0.0711 - val_accurac
Epoch 24/30
266/266 [==============================] - 27s 102ms/step - loss: 0.0728 - accuracy: 0.9741 - val_loss: 0.0694 - val_accurac
Epoch 25/30
266/266 [==============================] - 26s 97ms/step - loss: 0.0724 - accuracy: 0.9746 - val_loss: 0.0711 - val_accuracy
Epoch 26/30
266/266 [==============================] - 28s 105ms/step - loss: 0.0726 - accuracy: 0.9746 - val_loss: 0.0761 - val_accurac
Epoch 27/30
266/266 [==============================] - 27s 103ms/step - loss: 0.0723 - accuracy: 0.9743 - val_loss: 0.0706 - val_accurac
Epoch 28/30
266/266 [==============================] - 28s 106ms/step - loss: 0.0735 - accuracy: 0.9740 - val_loss: 0.0712 - val_accurac
Epoch 29/30
266/266 [==============================] - 28s 106ms/step - loss: 0.0744 - accuracy: 0.9734 - val_loss: 0.0726 - val_accurac
Epoch 30/30
266/266 [==============================] - 28s 106ms/step - loss: 0.0736 - accuracy: 0.9740 - val_loss: 0.0693 - val_accurac
1295/1295 [==============================] - 5s 4ms/step - loss: 0.0779 - accuracy: 0.9738
Accuracy: 97.38
```


Figure 5-4: LSTM Training Model

5.6.2. RNN

RNNs can be used to represent the temporal dynamics of network traffic data, which might be useful for DDoS attack detection and classification. They can recognize suspicious patterns suggestive of DDoS attacks and process consecutive data streams in real-time. RNNs can identify anomalies or departures from typical behavior by learning to record the temporal connections between network events. RNNs can categorize incoming traffic into distinct attack categories and offer early warning of possible DDoS attacks by training on past network traffic data and their capacity to manage it. Figure 5-5, below shows the implementation of the RNN model. As we saw, the model was defined and then adjusted with learning rate and optimizer, and compiled using the accuracy as metrics and categorical cross entropy as loss function.

```
[ ]  # SimpleRNN model

     rnn_model = Sequential()
     rnn_model.add(SimpleRNN(128,activation='relu', input_shape=(W.shape[1], 1)))
     rnn_model.add(Dropout(0.01))
     rnn_model.add(Dense(64, activation='relu'))
     rnn_model.add(Dropout(0.2))
     rnn_model.add(Dense(7, activation='softmax'))

     learning_rate = 0.001
     optimizer = Adam(learning_rate=learning_rate)

     # Compile the SimpleRNN model
     rnn_model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])

     # Train the SimpleRNN model
     rnn_history = rnn_model.fit(X_train, Y_train, validation_data=(X_test, Y_test), epochs=20, batch_size=1000, callbacks=[early_
```

```
166/166 [==============================] - 14s 86ms/step - loss: 0.1259 - accuracy: 0.9474 -
Epoch 12/20
166/166 [==============================] - 14s 87ms/step - loss: 0.1199 - accuracy: 0.9487 - val_loss: 0.1111 - val_accuracy:
Epoch 13/20
166/166 [==============================] - 14s 84ms/step - loss: 0.1194 - accuracy: 0.9504 - val_loss: 0.1254 - val_accuracy:
Epoch 14/20
166/166 [==============================] - 14s 85ms/step - loss: 0.1210 - accuracy: 0.9479 - val_loss: 0.1121 - val_accuracy:
Epoch 15/20
166/166 [==============================] - 15s 88ms/step - loss: 0.1162 - accuracy: 0.9486 - val_loss: 0.1202 - val_accuracy:
Epoch 16/20
166/166 [==============================] - 14s 85ms/step - loss: 0.1171 - accuracy: 0.9486 - val_loss: 0.1146 - val_accuracy:
Epoch 17/20
166/166 [==============================] - 15s 88ms/step - loss: 0.1147 - accuracy: 0.9499 - val_loss: 0.1208 - val_accuracy:
Epoch 18/20
166/166 [==============================] - 14s 85ms/step - loss: 0.1211 - accuracy: 0.9467 - val_loss: 0.1164 - val_accuracy:
Epoch 19/20
166/166 [==============================] - 14s 85ms/step - loss: 0.1063 - accuracy: 0.9522 - val_loss: 0.1065 - val_accuracy:
Epoch 20/20
166/166 [==============================] - 14s 86ms/step - loss: 0.1061 - accuracy: 0.9522 - val_loss: 0.0975 - val_accuracy:
```

Figure 5-5: RNN Training Model

5.6.3. MLP

MLPs can be used to classify DDoS attacks by using attributes that are taken from data about network traffic. They can categorize traffic into various attack categories because they can understand the correlations between input attributes and attack labels. Multi-layer packet analyzers can discern patterns linked to several DDoS attack types by analyzing a range of network traffic parameters, including protocol type, traffic volume, and packet size. MLPs are capable of effectively classifying incoming traffic and differentiating between harmful (Seba et al., 2024) and legitimate activities through training on labeled datasets that contain instances of both normal and abnormal. Figure 5-6 below shows the implementation of the MLP model. As we saw the model was defined and then adjusted with learning rate and optimizer, compiled using the accuracy as metrics and categorical cross entropy as loss function.

```
# MLP model
mmlp_model = Sequential()
mmlp_model.add(Dense(64, activation='relu', input_shape=(W.shape[1],)))
mmlp_model.add(BatchNormalization())
mmlp_model.add(Dropout(0.1))
mmlp_model.add(Dense(64, activation='tanh'))
mmlp_model.add(Dropout(0.2))
mmlp_model.add(Dense(64, activation='tanh'))
mmlp_model.add(Dropout(0.1))
mmlp_model.add(Dense(7, activation='softmax'))
learning_rate = 0.01
optimizer = Adam(learning_rate=learning_rate)
# Compile the MLP model
mmlp_model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
# Train the MLP model
#l=lstm_model.fit(X_train, Y_train, validation_data=(X_test,Y_test), epochs=10, batch_size=500, callbacks=[early_stopping])
m= mmlp_model.fit(X_train, Y_train, validation_data=(X_test, Y_test), epochs=100, batch_size=50, callbacks=[early_stopping])
```

Figure 5-6: MLP Training Model

5.6.4. CNN with LSTM

Hybridizing two different DL models for better performance was implemented. The first model was a CNN, which took the input data. The output of the CNN was then fed to an LSTM model for training, where classification was done. The final output, which was the attack types, was produced. Figure 5-7, below shows the implementation of the combination of CNN and LSTM models. As depicted, the models were defined separately, followed by adjustment of the learning rate and optimizer. The two models were then combined and compiled using accuracy as a metric and categorical cross-entropy as a loss function.

```
cnn_model = Sequential()
cnn_model.add(Conv1D(32, kernel_size=2, activation='relu', padding='same', input_shape=(W.shape[1],1)))
cnn_model.add(MaxPooling1D(pool_size=2))
cnn_model.add(Conv1D(16, kernel_size=2, activation='relu', padding='same'))
cnn_model.add(MaxPooling1D(pool_size=2))
cnn_model.add(Flatten())

lstm_model = Sequential()
lstm_model.add(LSTM(128, input_shape=(X_train.shape[1], 1), return_sequences=True))
lstm_model.add(Dropout(0.2))
lstm_model.add(LSTM(64))
lstm_model.add(Dropout(0.2))
lstm_model.add(Dense(7, activation='softmax'))

#COMBINEMODEL
combined_model = Sequential()
combined_model.add(cnn_model)
combined_model.add(lstm_model)
```

Figure 5-7: CNN-LSTM Model

5.6.5. CNN with RNN

Hybridizing two different DL models for better performance was implemented. The first model was a CNN, which took the input data. The output of the CNN was then fed to an RNN model for training, where classification was done, and the final output produced was the attack types. Figure 5-8, below shows the implementation of the combination of CNN and RNN models. As depicted, the models were defined separately, followed by adjustment of the learning rate and optimizer. The two models were then combined and compiled using accuracy as a metric and categorical cross-entropy as a loss function.

```
#1D CNN
cnn_model = Sequential()
cnn_model.add(Conv1D(32, kernel_size=2, activation='relu', padding='same', input_shape=(W.shape[1],1)))
cnn_model.add(MaxPooling1D(pool_size=2))
cnn_model.add(Conv1D(16, kernel_size=2, activation='relu', padding='same'))
cnn_model.add(MaxPooling1D(pool_size=2))
cnn_model.add(Flatten())

rnn_model = Sequential()
rnn_model.add(SimpleRNN(64, activation='relu', input_shape=(W.shape[1], 1)))
rnn_model.add(Dropout(0.1))
#rnn_model.add(SimpleRNN(64))
#rnn_model.add(Dropout(0.01))
rnn_model.add(Dense(7, activation='softmax'))
```

```
combined_model = Sequential()
combined_model.add(cnn_model)
combined_model.add(rnn_model)

learning_rate = 0.001
# Create the Adadelta optimizer with the specified learning rate
#optimizer = Adadelta(learning_rate=learning_rate)
optimizer = Adam(learning_rate=learning_rate)
# Compile the MLP model
combined_model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
#mlp_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
# Train the CNN-RNN model


combined_model.fit(X_train, Y_train, validation_data=(X_test, Y_test), epochs=27, batch_size=1000, callbacks=[early_stopping]
```

Figure 5-8: MODEL CNN-RNN

5.6.6. CNN with MLP

Hybridizing two different DL models for better performance was implemented. The first model was a CNN, which took the input data. The output of the CNN was then fed to an MLP model for training, where classification was done. The final output, which was the attack types, was produced. Figure 5-9, below shows the implementation of the combination of CNN and MLP

models. As depicted, the models were defined separately, followed by adjustment of the learning rate and optimizer. The two models were then combined and compiled using accuracy as a metric and categorical cross-entropy as a loss function.

```
#1D CNN
cnn_model = Sequential()
cnn_model.add(Conv1D(32, kernel_size=2, activation='relu', padding='same', input_shape=(Z.shape[1],1)))
cnn_model.add(MaxPooling1D(pool_size=2))
cnn_model.add(Conv1D(16, kernel_size=2, activation='relu', padding='same'))
cnn_model.add(MaxPooling1D(pool_size=2))
cnn_model.add(Flatten())

# MLP model
mmlp_model = Sequential()
mmlp_model.add(Dense(64, activation='relu', input_shape=(W.shape[1],1)))
mmlp_model.add(Dropout(0.4))
mmlp_model.add(Dense(64, activation='relu'))
mmlp_model.add(Dropout(0.2))
mmlp_model.add(Dense(64, activation='relu'))
mmlp_model.add(Dropout(0.4))
mmlp_model.add(Dense(32, activation='relu'))
mmlp_model.add(Dropout(0.4))
```

```
mmlp_model.add(Dropout(0.4))
mmlp_model.add(Dense(7, activation='softmax'))

#COMBINEMODEL
combined_model = Sequential()
combined_model.add(cnn_model)
combined_model.add(mmlp_model)

learning_rate = 0.001
# Create the Adam optimizer with the specified learning rate
optimizer = Adam(learning_rate=learning_rate)

# Compile the MLP model
combined_model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])

# Train the MLP model

c=combined_model.fit(X_train, Y_train, validation_data=(X_test, Y_test), epochs=30, batch_size=800, callbacks=[early_stopping
```

Figure 5-9: MODEL CNN-MLP

5.7. Implementation of the Mitigation

For this part of the mitigation work, we installed a Mininet and POX controller to set up our environment to start our implementation and also used the Miniedit app to redesign our network area to simulate our mitigation strategy.

Why? POX controller Modularity: POX offers a modular framework in which various network services (such as routing, switching, and firewalling) can be implemented as individual

modules. Ease of Use: it's written in Python and simple to use for rapid prototyping and development. In terms of Extensibility: Developers can add new modules to POX or change existing ones to satisfy specific network requirements. POX can be used to create custom network management programs that dynamically alter network configurations in response to current traffic conditions.

POX is frequently used in academic and research contexts to investigate novel networking protocols and topologies (AbdulRaheem et al., 2023), (Sun et al., 2022), (Habib & Khursheed, 2024) and (Mansoor et al, 2023).

In Figure 5-10, we can see that we have constructed our hosts, controllers, and switches. We had taken our network area having 3 POX controllers, 8 switches, and 30 hosts where two of them are designated as attackers in Figure 5-11. Our primary objective is to demonstrate the source IP address of the attacker by flow rules and drop those packets.

Overview of Architecture

We have thirty hosts, eight switches, and three POX controllers in our mitigation-focused architecture. The network is set up in a way that

- Four switches were managed by one controller.
- Each of the other two controllers is in charge of three switches.
- Thirteen and seventeen hosts, respectively, were under the control of these controllers.
- This redundancy makes sure that the other controllers can keep the network stable even if the primary controller is impacted by a bottleneck or attack.

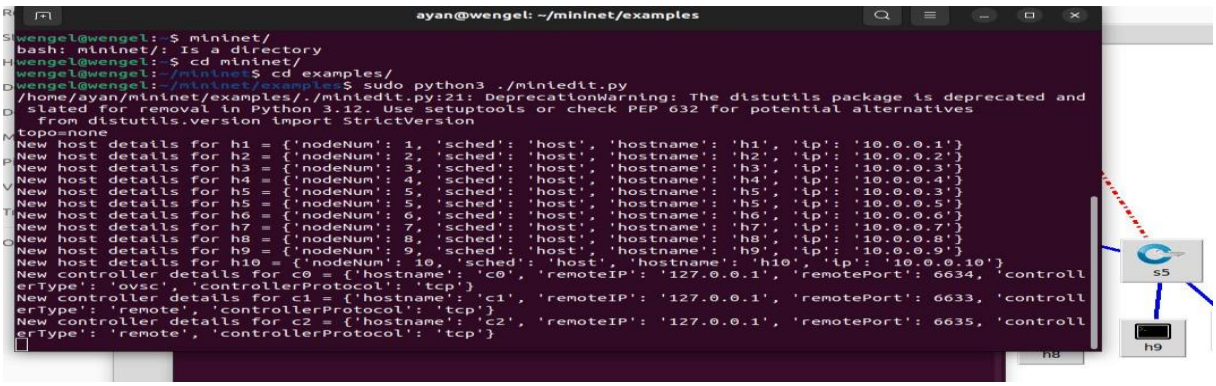


Figure 5-10: Configuring Host Controller and Switches

Process for Detection and Mitigation

- Real-time Traffic Monitoring: The system continuously keeps track of real-time network traffic information. Packet headers, flow statistics, and other pertinent metrics were included in this data.
- Deep Learning-based Detection: The monitored traffic is analyzed by a deep learning model that has been trained beforehand. The purpose of this model was to detect patterns that were common to DDoS attacks, like abnormally high traffic from particular IP addresses or particular traffic patterns.
- Attacker Identification: The deep learning model flags the IP addresses connected to the malicious traffic when it notices a possible DDoS attack. Based on the learned traits that distinguish attack traffic from legitimate traffic, this identification is made.
- Mitigation Triggering: The system initiates the mitigation process when it detects an attack. The POX controllers in charge of the impacted POXs were informed of the discovered attacker IP addresses.
- Flow Rules: The switches that were under the control of the POX controllers were subject to particular flow rules that were applied by them. The switches are instructed by these flow rules to reject any packets coming from the IP addresses of the attackers. "table=0, priority=100, ip, nw_src=ATTACKER_IP, actions=drop" is an example of a flow rule. This rule makes sure that any packets from the IP addresses of the detected attackers are rejected before they have a chance to cause more damage to the network.
- Maintaining Network Stability: The other controllers, who have been watching their network segments, take over to manage traffic and implement mitigation measures if the main controller encounters a bottleneck or is overpowered by the attack. The load balancing and redundancy contribute to the general stability of the network.

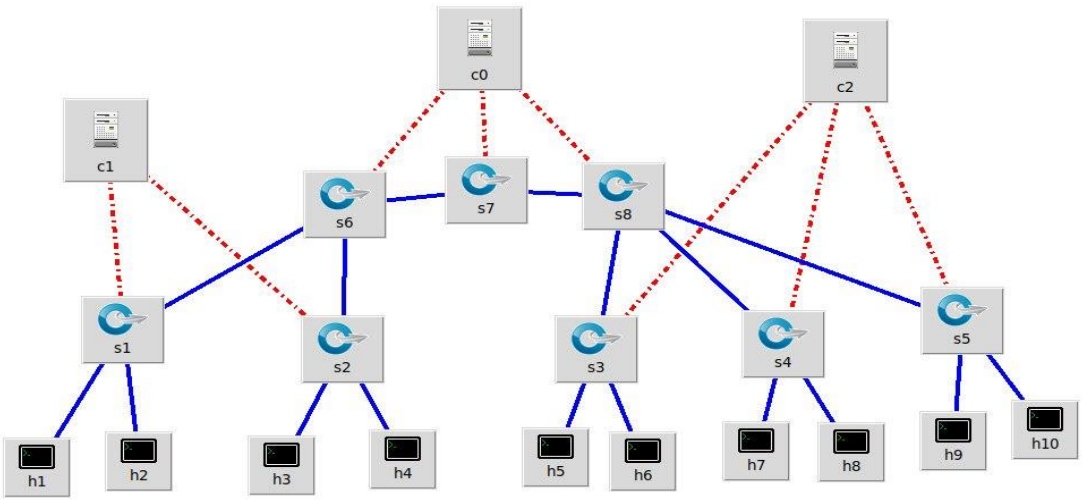


Figure 5-11: Mitigation Design of the SDN

This approach used deep learning for accurate detection and SDN capabilities for efficient and dynamic mitigation, offering a robust defense against DDoS attacks.

Step 1: Continuous monitoring of network traffic data.

Step 2: The deep learning model detects abnormal traffic patterns indicative of DDoS attacks.

Step 3: Identification of attacker IP addresses based on detected patterns.

Step 4: Notification of identified attackers to the relevant POX controllers.

Step 5: Application of flow rules to drop packets from attacker IP addresses.

Step 6: Backup controllers take over if the main controller is compromised.

After the initial configuration of the network environment on Mininet which should be running on Ubuntu. We then start up our POX controller shown in Figure 5-12, having mininet and POX directory. Inside the Mininet we have our configured network setup in a python file and inside the pox directory we have 13.edditing.py and hybriddetection.py files which are shown in Figure 5-13.

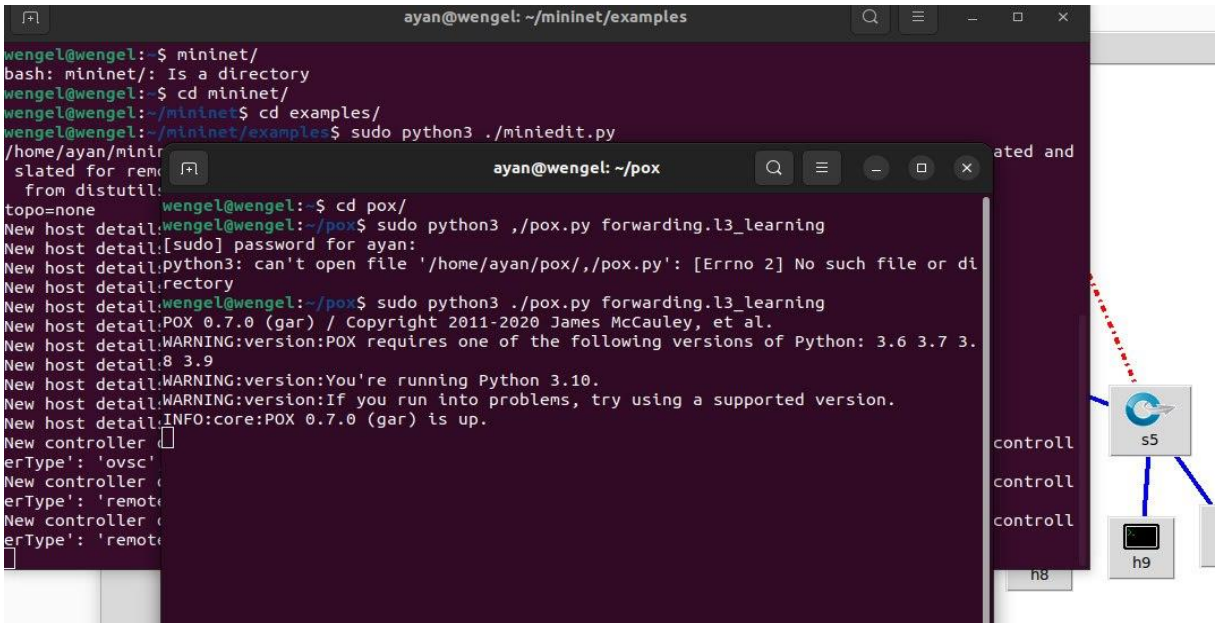


Figure 5-12: POX Controller UP

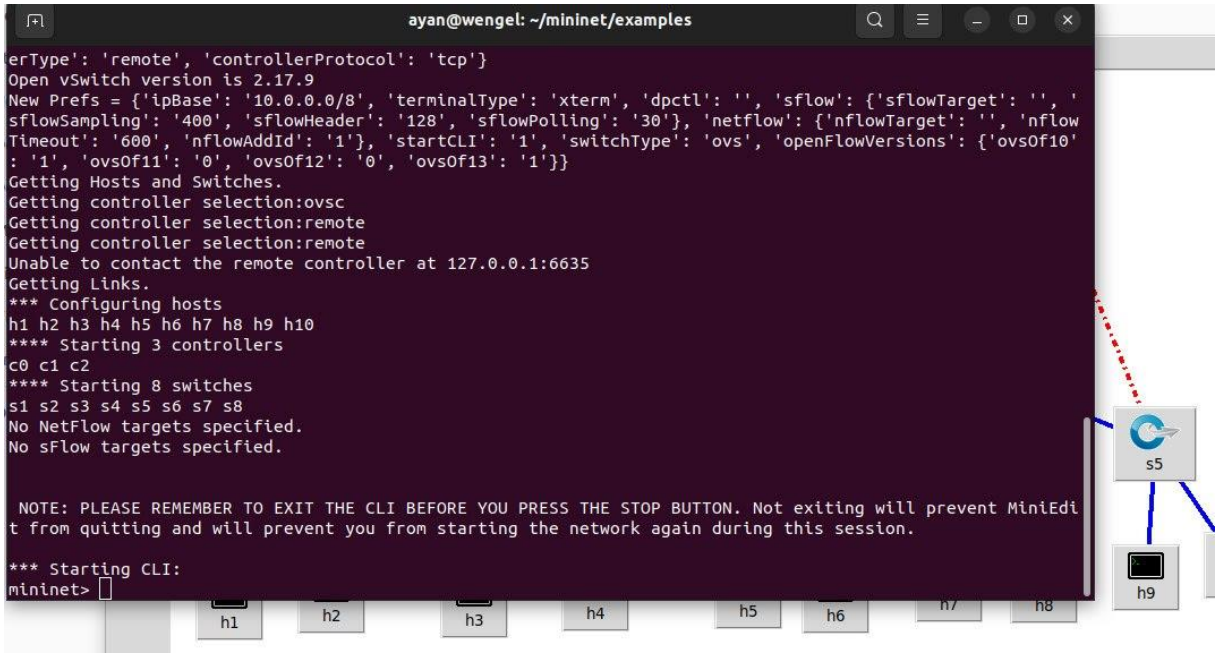


Figure 5-13: Run POX Controller and Create Mininet Topology

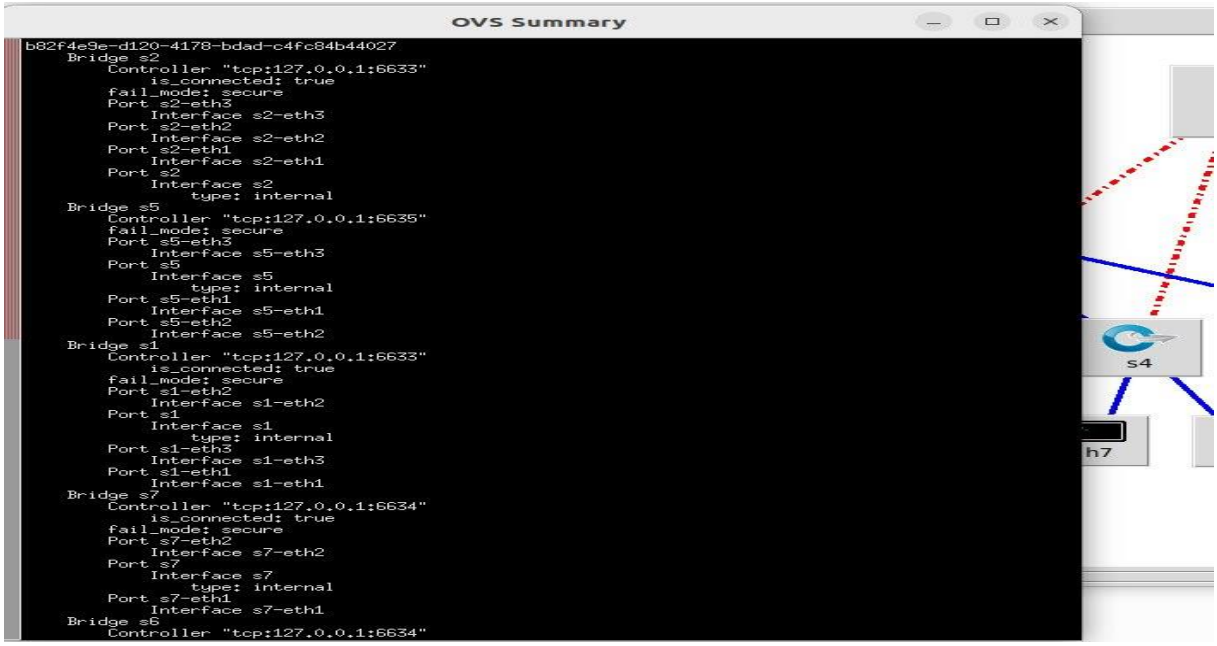


Figure 5-14: Summary of Controller

5.8. Model testing and evaluation

In our experiment, we evaluated the performance of the models using various metrics, mostly accuracy, outlined in Chapter 3. These metrics helped us evaluate our model in detecting and classifying different DDoS attacks. By accuracy, precision, recall, F1 score, confusion matrix, and ROC Curve. We have split our data into test size = 20 and training size = 80 after comparing it to 30/70 data splitting.

Table 5-2: Comparison of Train Test Split

Algorithm	Performance measure metrics Train test split data							
	(70 training and 30 tests)				(80 training and 20 tests)			
	Accuracy	precision	Recall	F1 score	Accuracy	Precision	Recall	F1 score
CNN-LSTM	98..88	94.23	97.65	95.79	99.79	95.30	98.21	96.91
CNN-MLP	99.01	98.56	97.62	97.89	99.89	99.27	98.50	98.88
CNN-RNN	97.77	97.67	97.69	96.98	98.99	98.59	98.59	97.17

CHAPTER SIX

Result and Description

6. Chapter Overview

In this chapter, we discussed the experiments on our proposed models and the results obtained using deep learning models to detect and classify. We proposed and tested the datasets with multiple experiments of models like LSTM, MLP, RNN, 1DCNN-LSTM, 1DCNN-MLP, and 1DCNN-RNN. We had done our research models to do both detection along classification. Secondly, we compared our two datasets which were InSDN and CICDDoS2019 having 25 features with different hyper-parameter combinations, and discussed our results according to our experimental models.

6.1. Model Experiments

In this experiment, we used the most common hyper-parameters to adjust our models like using the SoftMax layer to take in the input layer from the other layer and classify the data into six and four according to our dataset. We used categorical cross-entropy classification and the Adam optimizer having a learning rate of 0.001 and different drop-outs of 0.01, 0.4, 0.02, and 0.5 depending on the dataset. We added an early stopping with the monitor of loss and also considered different epochs and batch sizes of 27 and 1000.

Table 6-1: General Hyper-parameter Tuning

Hyper-parameter tuning								
Models	Filters	Kernel size	Pool size	Model units	Dropout rate	Learning rate	activation	Optimizer
LSTM	-	-	-	128,64,32	0.02,0.2	0.001,0.01	Relu	Adam
MLP	-	-	-	64,32	0.02, 0.2	0.001,0.01	Relu, tanh	Adam
RNN	-	-	-	64,32	0.02,0.02	0.001,0.01	Relu	Adam

1DCNN-LSTM	64,32	2	2	64,32	0.4,0.05	0.001,0.01	Relu. tanh	Adam
1DCNN-MLP	64,32	2	2	64,32	0.4, 0.05	0.001,0.01	Relu, tanh	Adam
1DCNN-RNN	64,32	2	2	64,32	0.4, 0.05	0.001,0.01	Relu, tanh	Adam

6.1.1. Experiment 1: LSTM Model

We obtained these results, by choosing our hyper-parameters SoftMax to classify the attack types and set our optimizer to Adam with a learning rate of 0.001 and Relu activation. We also used a dropout (0.01, 0.02, and 0.04). We mentioned the parameters we used in Table 6-1 above. From Figure 6-1 and 6-2, we can see that the model trained up to 27 epochs, and the training loss reduced from 0.187 to 0.017. The validation loss starts from 0.028 to 0.016, and the training accuracy has increased from 98.34 to 98.93 and training accuracy also improved from 85.78 to 99.42.

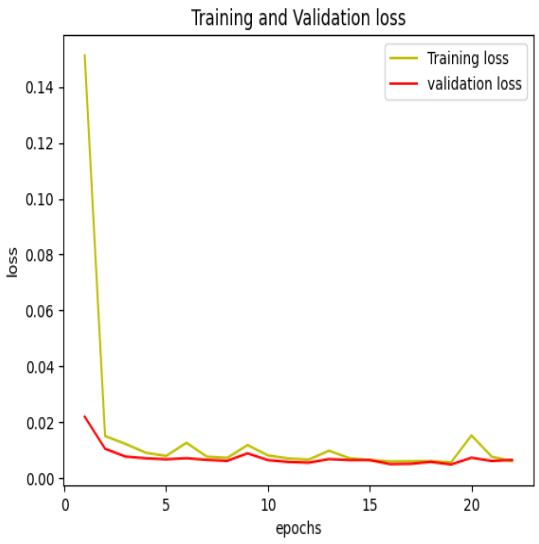


Figure 6-1:Training and Validation Loss LSTM CICDDoS2019

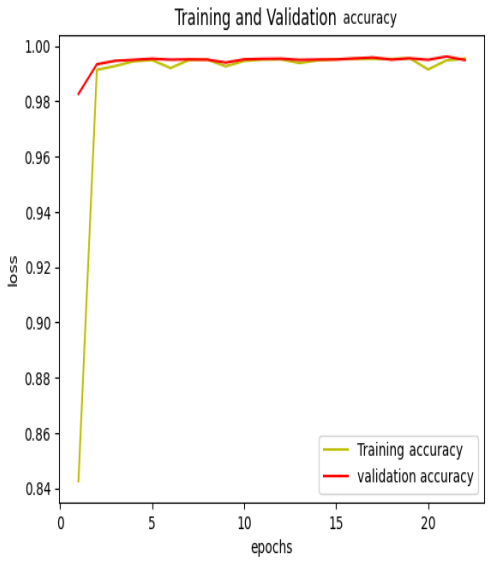


Figure 6-2:Training and Validation Accuracy LSTM CICDDoS2019

6.1.2. Experiment 2: MLP Model

We obtained these results, by choosing our hyper-parameters stated in Table 6-1. From Figure 6-3 and 6-4, we can see that the model trained up to 27 epochs, the training loss has reduced from 0.037 to 0.0036, and the testing loss from 0.0067 to 0.0032. The testing accuracy increased from 99.58 to 99.82 and the training accuracy also improved from 98.23 to 99.78.

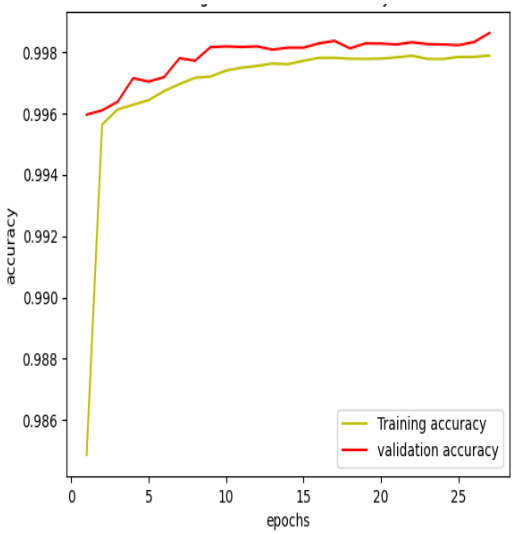


Figure 6-3:Training and Testing Accuracy of MLP

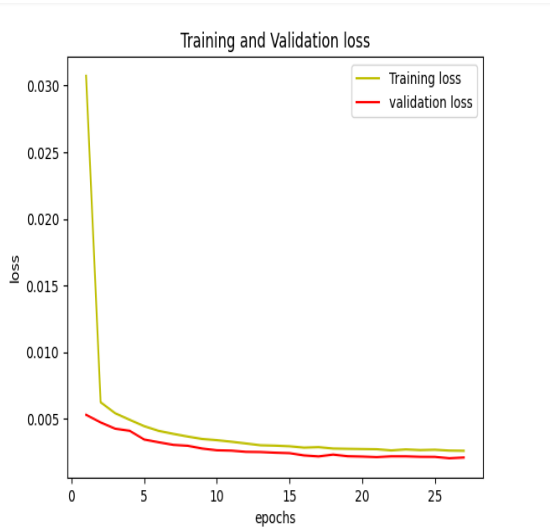


Figure 6-4: Training and Testing Loss of MLP

6.1.3. Experiment 3: RNN Model

We obtained these results by choosing our hyper-parameters in Table 6-1. From Figures 6-5 and 6-6, we can see that the model has trained up to 40 epochs, the training loss reduced from 1.189 to 0.149, and the testing loss from 0.5356 to 0.1672. The training accuracy increased from 67.62 to 95.45 and test accuracy also improved from 83.34 to 97.67.

Figure 6-5: Training and Testing Accuracy of RNN.

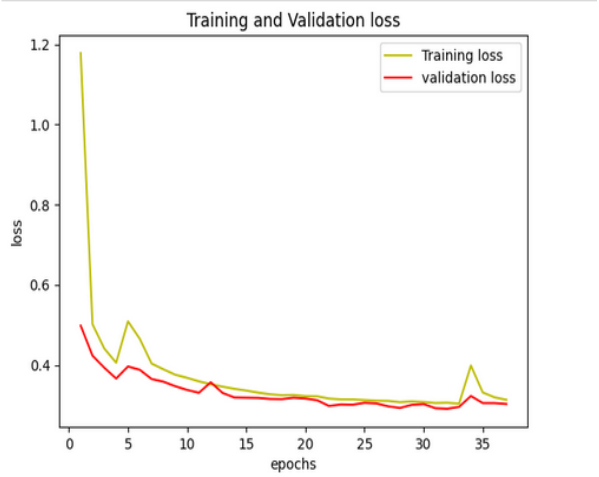


Figure 6-6: Training and Testing Loss of RNN.

6.1.4. Experiment 4: CNN-MLP Model

We obtained these results by choosing our hyper-parameters table 6-1, and this model is our proposed model which gives the highest detection and classification accuracy.

Algorithm for 1DCNN-MLP

Having the dataset: 25 features, optimizer, loss, metrics

Input: Preprocessed network traffic data containing features relevant to DDoS attack detection.
Output: A dictionary object containing the detected labels for each network flow, indicating whether it is normal traffic or DDoS attacks.

Begin

 # Step 1: Create a sequential model:

Model ⟵ Sequential ()

 # Step 2: Add a 1D convolutional layer:

Conv1D_layer ⟵ Conv1D (filters = 128, kernel_size = 3, activation = 'relu', input_shape = (sequence_length, num_features))

model.add (Conv1D_layer)

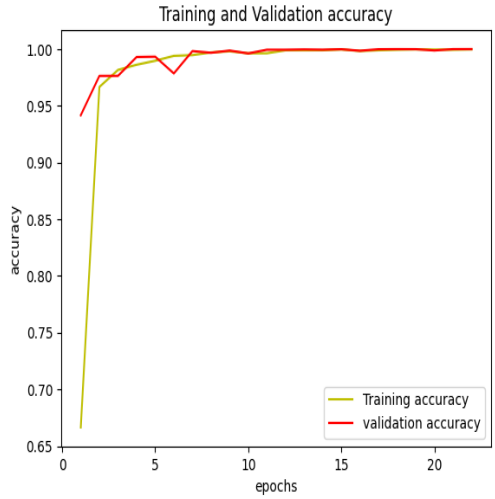
model.add (MaxPooling1D (pool_size=2))

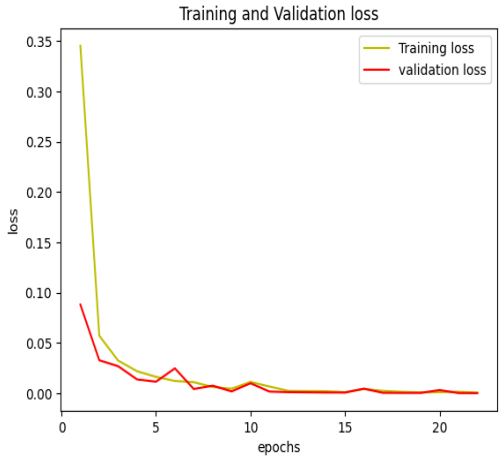
model.add (Conv1D_layer)

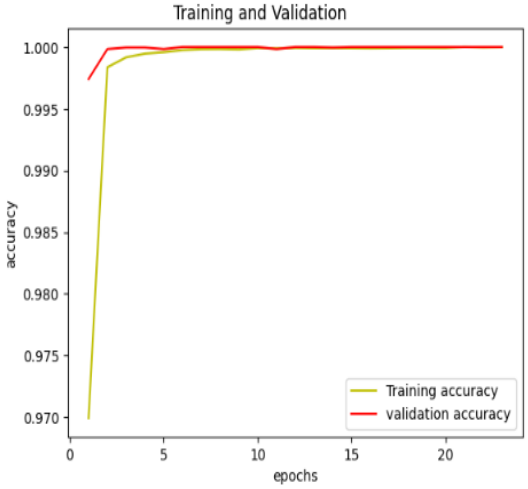
```
    model.add (MaxPooling1D (pool_size=2))
    model.add(Flatten())
# Step 3: Add MLP:
    Dense_layer1◄── (Dense(units = 256, activation = 'relu'))
    model.add (Dense_layer1)
    model.add (BatchNormalization ())
    model.add (Dropout (0.2))
    Dense_layer2 ◄── (Dense (units = 128, activation = 'relu'))
    model.add (Dense_layer2)
    model.add dense (num_classes, activation='softmax')
    Optimizer ◄── Adam(learning_rate=0.001)
#Step 4: compile the combined model
        model.compile(optimizer=optimizer, loss ='categorical_crossentropy',  metrics =
        ['accuracy'])
# Step 5: train the model:
       Model.fit (X_train, y_train, epochs=20, batch_size=64, validation_data=(X_val,
        y_val), callbacks=[EarlyStopping(monitor='val_loss', patience=3)], verbose=1)
# Step 6: Predict on test data:
        y_pred ◄── model.predict(X_test)
# Step 9: Assign labels based on predictions
        detected_labels ← {}
            for each flow_id, pred in zip(test_flow_ids, y_pred) do
                class_index ← argmax(pred)
                detected_labels[flow_id] ← class_labels[class_index]
             end for
        return detected_labels
End
```

From Figures 6-7 and 6-8, we can see that the model trained up to 27 epochs. The training loss was reduced from 0.329 to 0.003, and the testing loss from 0.018 to 0.003. The training accuracy increased from 96.67 to 99.79 and test accuracy improved from 99.67 to 99.89 in the CICDDoS2019 data set and also showed significant work in the other data, having the training

loss reduced from 0.189 to 0.0023, and the testing loss from 0.018 to 0.0023. The training accuracy increased from 76.67 to 99.79 and test accuracy also improved from 95.67 to 99.91.







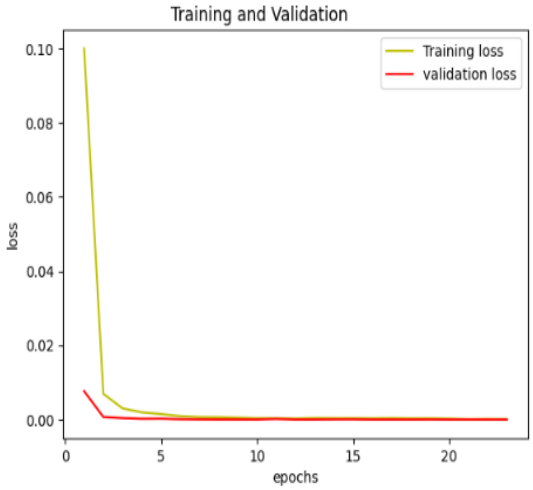


Figure 6-7: Training and Testing Accuracy of CNN-MLP Figure 6-8: Training and Testing Loss of CNN-MLP.

6.1.5. Experiment 5: CNN-LSTM Model

We obtained these results, by choosing our hyper-parameters in Table 6-1. From Figures 6-9 and 6-10, we can see that the model trained up to 29 epochs and the training loss reduced from 0.0289 to 0.0023 and testing from 0.0067 to 0.0024. The training accuracy has increased from 98.78 to 99.78 and test accuracy also improved from 99.56 to 99.79.

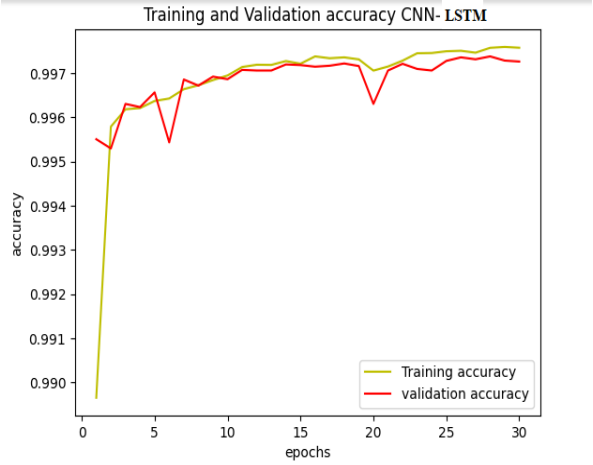


Figure 6-9: Training and Testing Accuracy of CNN-LSTM.

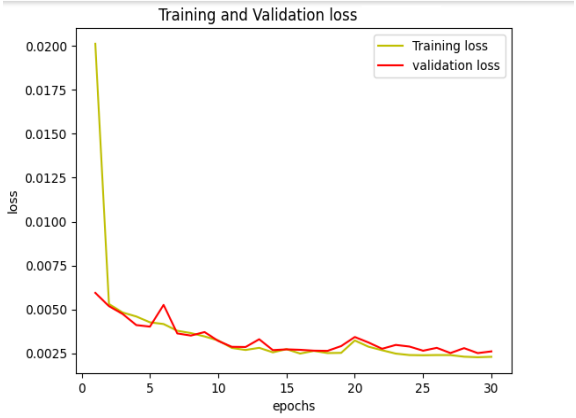


Figure 6-10: Training and Testing Loss of CNN-LSTM.

6.1.6. Experiment 6: CNN-RNN Model

We obtained these results, by choosing our hyper-parameters in Table 6-1. From Figures 6-11 and 6-12, we can see that the model trained up to 27 epochs, the training loss reduced from 0.0189 to 0.0112, and the testing loss from 0.054 to 0.0101. The training accuracy increased from 97.8 to 98.98 and test accuracy also improved from 98.67 to 98.99.

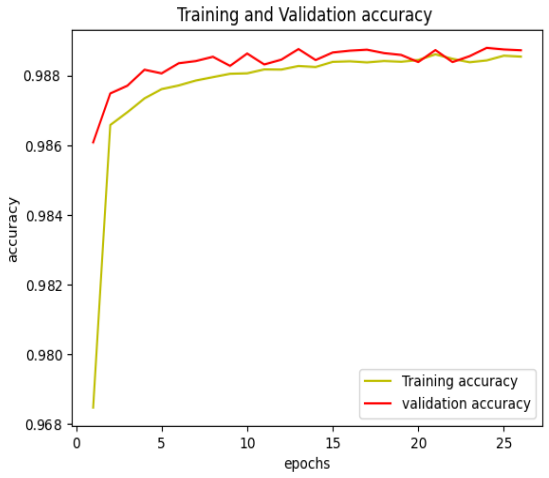


Figure 6-11:Training and Testing Accuracy of CNN-RNN

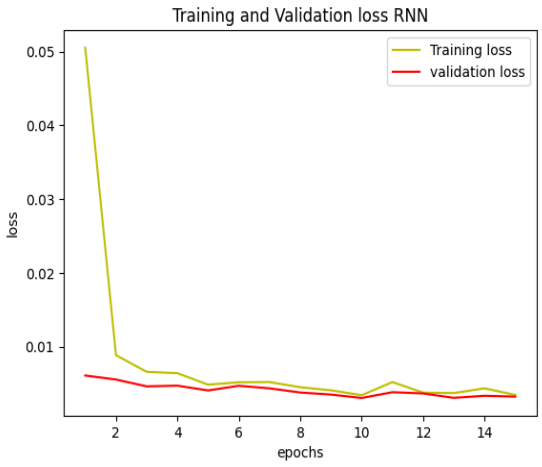


Figure 6-12: Training and Testing Loss of CNN-RNN.

6.2. Model Performance Evaluation

6.2.1. Classification Analysis Result

The main aim of our research work was to identify the optimized model to detect and classify DDoS attacks on SDN controllers using hybrid deep learning and to compare and contrast each performance on the attack types on the proposed models. To do our experiments, we used 6 types of attacks in InSDN datasets and 4 attack types out of 12. We applied training and testing split of 80 and 20 to our data respectively.

While comparing our data where the InSDN is small scale data having 207,146 after preproces sed and 894,567 larger scales for CICDDoS2019, we need to consider and differentiate the mo dels based on separate test data split (80% for training and 20% for testing) techniques and eva luated them by different metrics. We conducted experiments on each algorithm and finally too k the optimal one with the highest accuracy along with recall, F1-Score, precision, and loss of both training and testing. finally considered the ROC curve for false and true rates.

From our experimental analysis, we concluded that the hybrid DL model of 1DCNN-MLP wa s better than the rest of the models for detecting and classifying the attacks. We obtained these results, an accuracy of 99.91 for the InSDN dataset and 99.89 for the CICDDoS2019 dataset w hich is better than the rest. LSTM had an accuracy of 99.56, MLP had an accuracy of 99.78, R NN had an accuracy of 98.56, 1DCNN-LSTM had an accuracy of 99.79, 1DCNN-RNN had a n accuracy of 98.99, where1DCNN-MLP had reduced training_loss of 0.078 and testing_loss of 0.026 and training accuracy of 99.88 and validation accuracy = 99.89.

Generally, the following tables show the summary of the general and each attack type’s perfor mance of the algorithm by different measures metrics. To evaluate the performance mainly we considered the accuracy which shows the model’s ability to classify the DDoS attacks. As you can see from the above Table 6-2 above, almost all have the best accuracy, but in both cases o f the dataset, the hybrid model of the 1DCNN-MLP model has performed the highest accuracy among them. So, this indicates that our proposed model was capable of classifying six and fou r attack types properly along with normal traffic.

Table 6-2: Performance Measure of Two Datasets

Algorithm	Performance measure metrics Train test split data (80 training and 20 test)							
	InSDN				CICDDoS2019			
	Accuracy	precision	Recall	F1 score	Accuracy	Precision	Recall	F1 score
MLP	98.6	97.86	98.12.3	97.67	99.78	98.32	98.64	98.27
LSTM	99.12	95.78	97.68	96.98	99.42	94.34	98.92	96.36
RNN	97.89	94.54	92.45	89.98	98.54	93.98	91.87	88.89
CNN-LSTM	99.32	98.23	98.09	97.78	99.79	98.30	98.21	96.91
CNN-MLP	99.90	99.01	98.34	98.67	99.89	99.27	98.50	98.88
CNN-RNN	98.45	98.11	98.27	97.99	98.99	98.59	98.59	97.17

In Table 6-3 we saw each attack type's metrics like precision, recall, and F1 – score which indicates the attack type's individual. We used the performance metrics especially to assess the effectiveness of each of our models and checked which had the highest effectiveness among them to detect and classify the DDoS attacks.

Table 6-3: Performance Metrics CICDDoS2019

Algorithm Type	Attack Type	Performance metrics			
		Accuracy (%)	Precision	Recall	F1-score
LSTM	DrDoS_LDAP	99.45	82.71	97.09	86.33
	DrDOS_NTP		99.99	99.91	99,92
	SYN		99.88	98.70	99.92
	BENIGN		99.84	99.79	99.29
RNN	DrDoS_LDAP	98.69	81.47	96.30	88.27
	DrDOS_NTP		99.99	99.79	99.67
	SYN		99.83	99.01	99.42
	BENIGN		99.92	99.72	99.74
MLP	DrDoS_LDAP	99.67	93.42	97.57	95.45
	DrDOS_NTP		99.99	99.97	99.98
	SYN		99.88	99.72	99.80
	BENIGN		99.99	99.98	99.98

CNN-MLP PROPOSED SOLUTION	**DrDoS_LDAP**	**99.89**	**97.36**	**94.43**	**95.72**
	DrDOS_NTP		**99.99**	**99.98**	**99.98**
	SYN		**99.73**	**99.84**	**99.81**
	BENIGN		**99.99**	**99.98**	**99.99**
1DCNN-LSTM	DrDoS_LDAP	99.6	81.47	96.30	88.27
	DrDOS_NTP		99.99	99.79	99.67
	SYN		99.83	99.01	99.42
	BENIGN		99.92	99.72	99.74
1DCNN-RNN	DrDoS_LDAP	99.01	93.42	97.57	95.45
	DrDOS_NTP		99.99	99.97	99.98
	SYN		99.88	99.72	99.80
	BENIGN		99.99	99.98	99.98

Figures 6-13 and 6-14, demonstrated six classification algorithms used to detect and classify DDoS attacks by the datasets using train and test split (80/20). The Hybrid DL of 1DCNN-MLP had the highest accuracy while 1DCNN-LSTM, MLP, LSTM, 1DCNN-RNN, and RNN ranked second, third, fourth, fifth, and sixth based on their performance metrics achievements by accuracy.

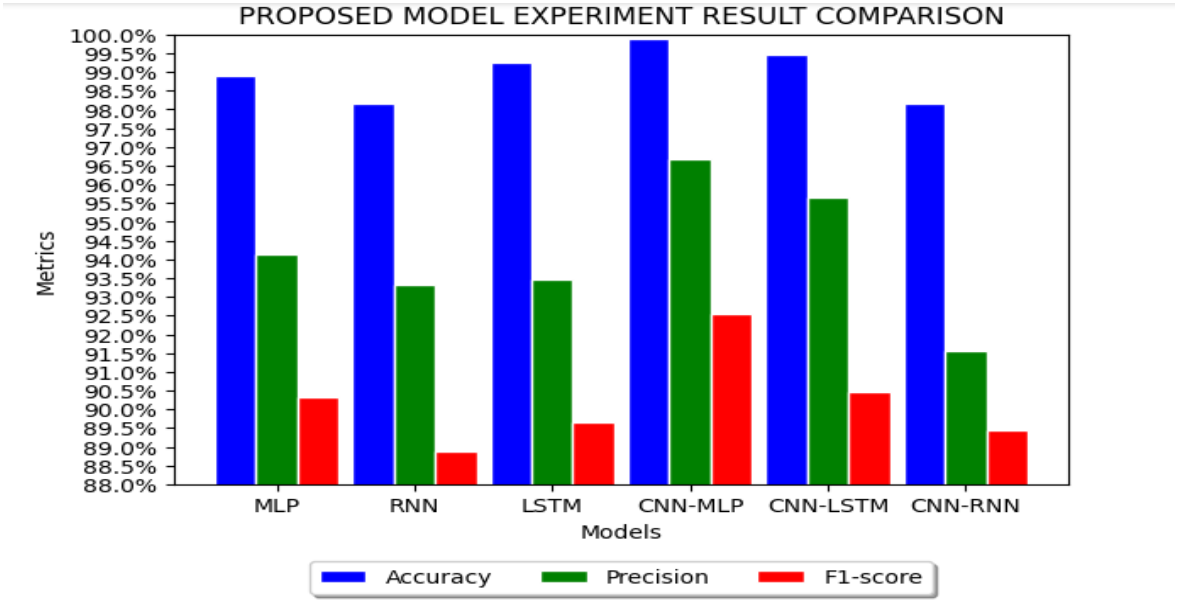


Figure 6-13: Experiment result in InSDN

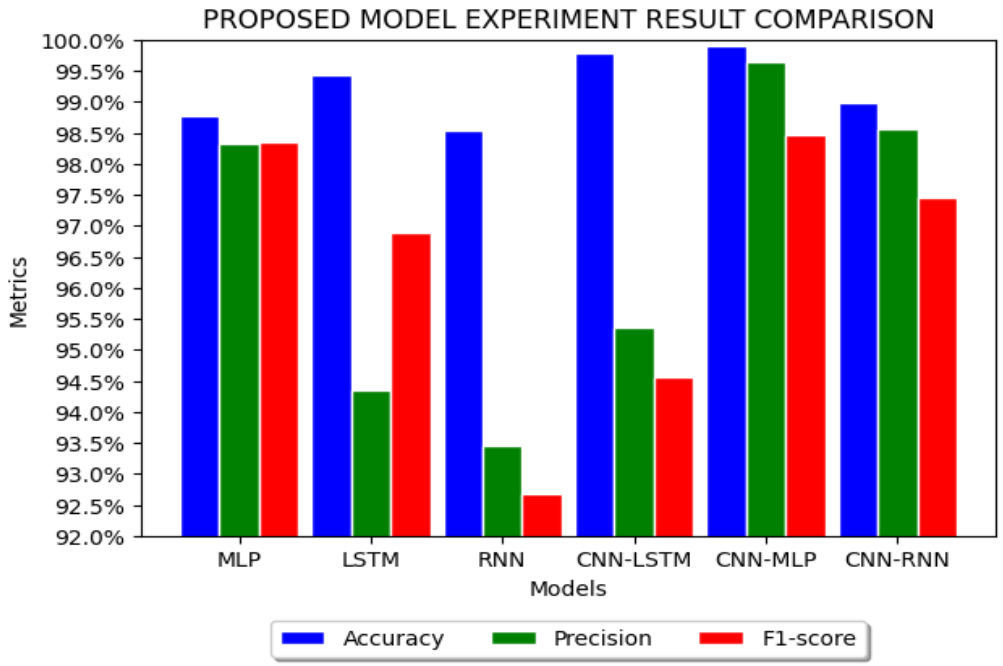
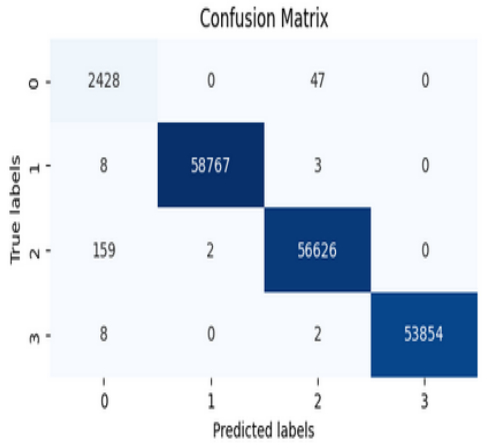


Figure 6-14: Experiment result in CICDDoS2019

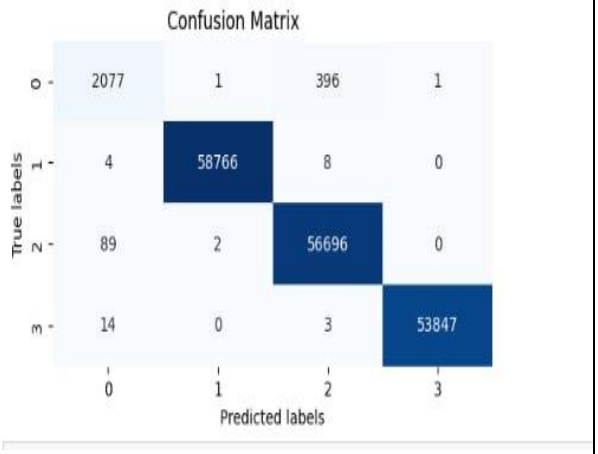
6.2.2. Confusion Matrix Visualization

Figure 6-15, shows the confusion matrices for six algorithms used to detect and classify DDoS attack types for BENIGN, DrDoS_LDAP, DrDoS_NTP, and SYN classes labeled as (0, 1, 2, and 3). For example, we saw that the proposed model metrics CNN-MLP from a total of 171,904 had four classes where the model was able to correctly classify 2461 as class 0, 58,768 as class 1, 56236 as class 2, and 53,854 as class 3. From class 0 (BENIGN), 2461 were classified correctly as true negative whereas the diagonals which are the sum of (56376 + 58768+ 53854 = 169,357) correctly classified as negative. For false positive (0+13+1 =14) model incorrectly classified 14 class 0 as negative and false negative (7+408+9 = 424) as class 0. As shown in Figure 6-15 below, we have used six algorithms to show classification algorithms used for detecting and classifying the DDoS attacks by showing the result of the confusion matrices. From the below Table 6-4 below, we have summarized the classified attacks displayed in the confusion metrics.

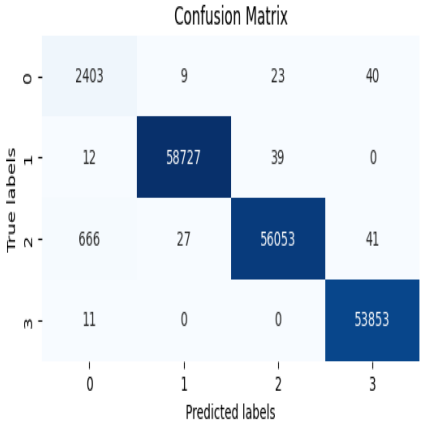
Mlp



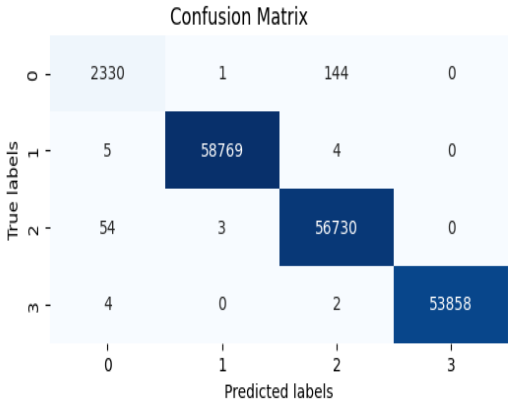
Cnn-RNN



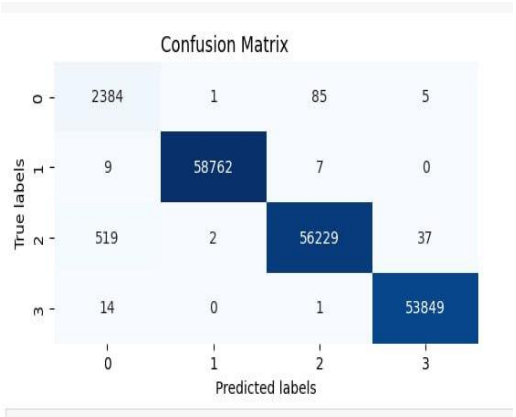
LSTM



CNN-MLP



CNN-LSTM



RNN

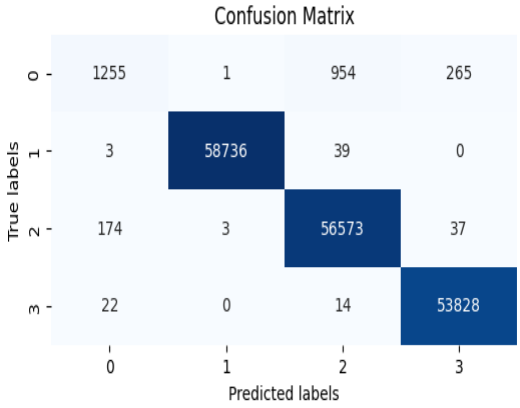


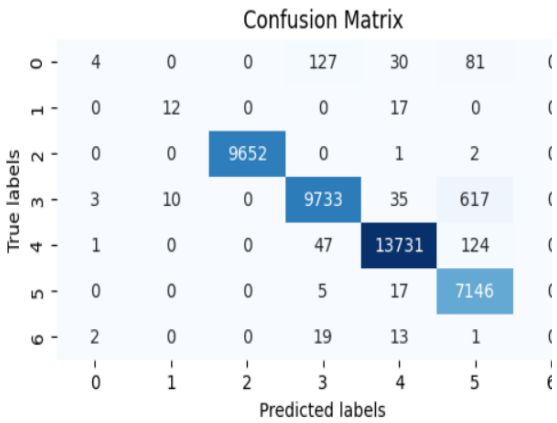
Figure 6-15: Confusion Matrices CICDDoS2019

Table 6-4: Confusion Metrics of CICDDoS2019 Datasets Result

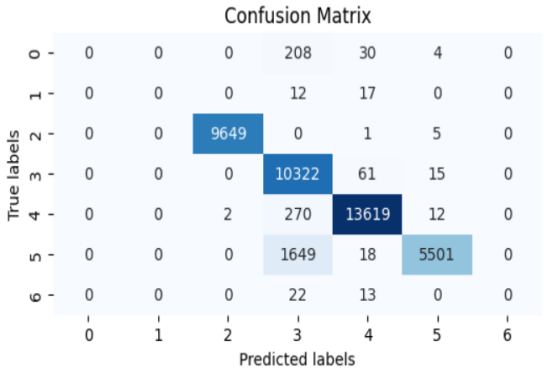
NO instance 171,904	Correctly classified	Incorrectly classified	Accuracy	Model
Class 0 (BENGIN)	2384	633	99.60	CNN-LSTM
Class 1 (DrDoS_LDAP)	58762	9		
Class 2 (DrDoS_NTP)	56229	38		
Class 3 (SYN)	53849	-		
Total	171224	680		
Class 0	2330	208	99.89	CNN-MLP
Class 1	58769	7		
Class 2	56730	2		
Class 3	53858	-		
Total	171,459	217		
Class 0	2077	505	99.01	CNN-RNN
Class 1	58766	10		
Class 2	56696	3		
Class 3	53853	-		
Total	171192	568		

Figure 6-13, shows the confusion metrics of the dataset of InSDN of the hybrid DL models. As shown, the hybrid model of the CNN-MLP outperforms the rest like the above data set used having an accuracy of 99.91% for classifying the DDoS attacks.

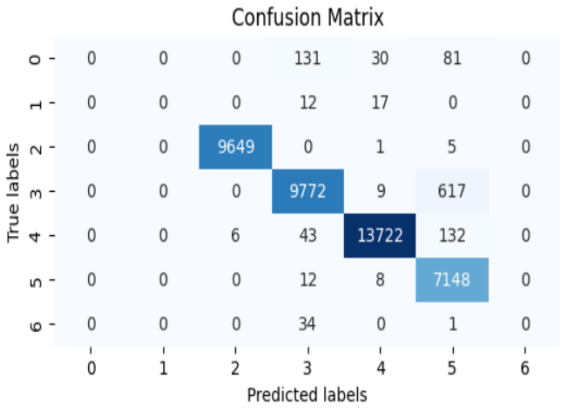
MLP



CNN-LSTM



CNN-RNN



CNN-MLP

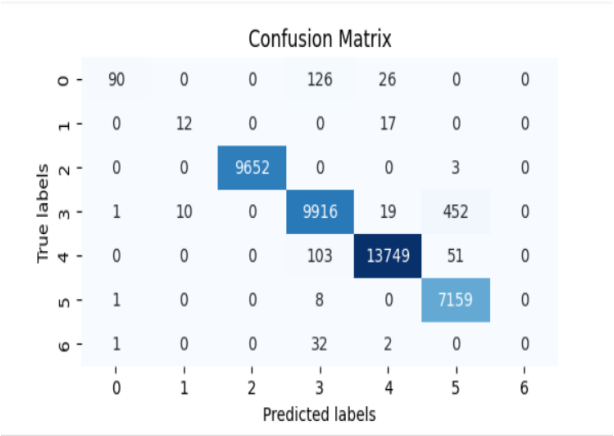


Figure 6-16: InSDN Confusion Matrix

6.2.3. Performance Based on AUC-ROC Metric

Roc (receiver operating characteristic) curve is a graph showing the performance of a classification model at all classification thresholds that has a true positive rate and a false positive rate where it's measured by the area under the ROC curve. Figure 6-18 below, shows that our proposed model which is the hybrid DL 1DCNN-MLP can separate well on AUC of 99.98 for BEGIN, 99.99 for DrDoS_LDAP, 99.99 for DrDoS_NTP and 99.96 for SYN.

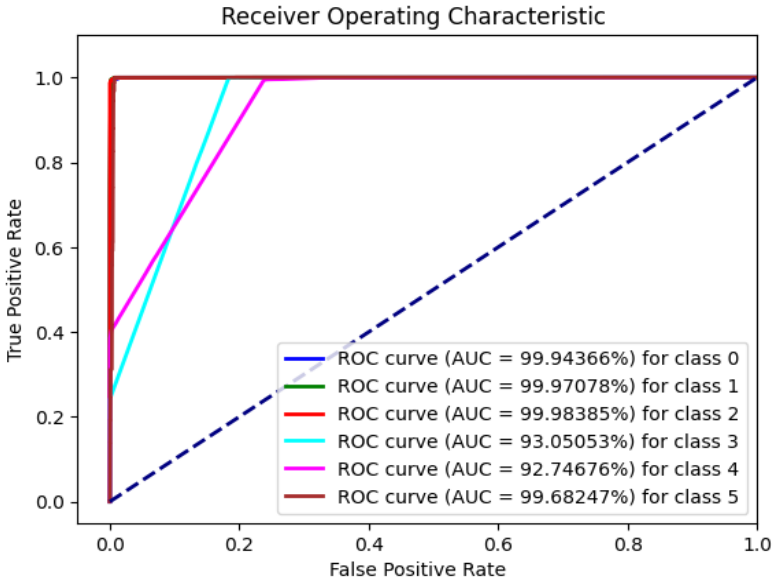


Figure 6-17: AUC-ROC InSDN

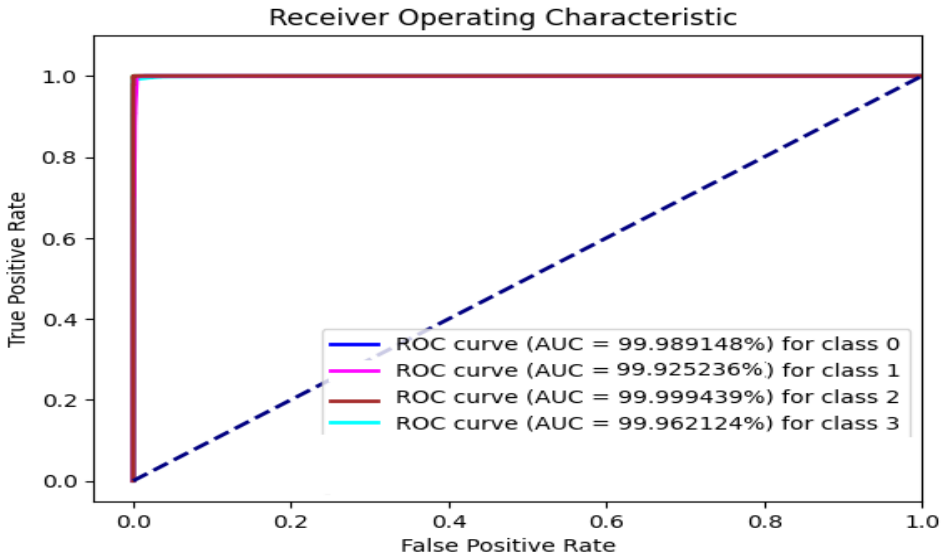


Figure 6-18: AUC-ROC CICDDoS2019

6.3. Experiment Result in Comparison with Other Related Work Models

Table 6-5, illustrates the comparison of our proposed work with other related works. We saw that our work had significantly shown an improvement regarding the performance metrics, dataset, and feature selection.

Table 6-5: Proposed work comparison with related work

Research Papers	technique	domain	Classification	Dataset	Performance metrics			
					Accuracy	precision	recall	F1-score
(Ahuja, Singal, & Mukhopadhyay, 2021)	SAE-MLP	SDN	Binary	CICDDoS 2019	99.75	99.96	99.69	99.82
(A. Singh et al., 2023)	SVM-RF	KSLND	Binary	custom	99.1%	99.16	-	99.23

(AbdulRaheem et al., 2023)	ML	Traditional	Binary	custom	97.2	94.9	98.5	95.8
(Mansoor et al, 2023)	RNN	IDS	Binary	Custom	94.186	92.14	81.89	94.27
(Gebremeskel et al., 2023)	LSTM	SDN	Categorical	CICDDoS2019	99.41	99.4	99.46	99.42
Our proposed model	CNN-MLP	SDN	Categorical	CICDD0S2019	99.89	99.74	99.78	99.79
		SDN	Categorical	InSDN	99.91	99.84	99.01	99.34

6.4. Result Discussion on Proposed Model

Our proposed model which is CNN-MLP was used to detect and classify the DDoS attack types to do our research and done many experiments. The first part of this was collecting data and using two public datasets for this research these were CICDDoS2019 and InSDN2020 datasets. We then preprocessed our data by cleaning, normalizing, and labeling it to a categorical cause we didn't only detect it as binary instead we classified some of the attack types found in the datasets. Next, we applied chi-square feature selection to have relevant features related to the target label in their effective score result. We then split our dataset into training and testing (80/20) and reshaped it to a suitable feed to the DL we had used to train on. We used three DLs; LSTM, RNN, MLP, and three hybrid DLs to show the effectiveness of the models; CNN-MLP, CNN-LSTM, and CNN-RNN. We trained our data on all of the models evaluated them in different evaluation metrics and applied many experiments to conclude our proposed solution.

The experiments started from:

A. **Preprocessing and feature selection,** we applied different techniques to choose the best datasets to be trained while preprocessing we finally considered normalizing it by MinMax standard scaler to transform the data and also converted the data type of the feature

containing the timestamp from object to data time form. For feature selection, we used **chi-square** feature selection to analyze class frequencies and compared the effects of PCA and chi-square. The results of this comparison showed that the chi-square was more significant than PCA, which led us to investigate the Chi-square test further because it ranks the scoring of importance or correlation to the target value. The Chi-square test is a statistical technique that's used to identify significant relationships among categorical data by determining whether observed variable co-occurrences significantly differ from expected rates under independence. It is well-known for its flexibility in handling large datasets. Some highly relevant features were selected and some of them were: flow pkts, Bwdpkts, timestamp_nanosec, pkt size avg, Fwdpktlen min, Ugr flag count, pktleng mean, ACK flag count, avgfwd segment size these features highly used in detecting DDoS attacks because they capture various characteristics and behaviors commonly exhibited during the attacks. So, monitoring these features and analyzing their values, patterns, and deviations from expected norms makes it highly possible to identify traffic patterns that indicate each attack type because, it often involves flooding target systems with massive volumes of traffic, exploiting vulnerabilities, or exhibiting abnormal traffic behaviors.

B. Hyper-parameter tuning: In this experiment, we did lots of hyper-parameter tuning, meaning it's a process of selecting the perfect hyper-parameters by changing the parameters until we find the one that is suitable for the solution we want to propose. This tuning helped us find the optimal solution for our proposed solution. In our work, we used 80/20 data splitting for our datasets, reshaped of data, pool_size = 2,3, units = 64, learning rate = 0.001, dropout_rate = 0.04, and 0.02, optimizer = Adam, early stopping = 5 where these parameters were found to be the best fit for our model with an accuracy of 99.89%, precision of 99.27%, along with detecting rate of 99.99%. All the outcomes of the detection and classification of our model results are displayed in Table 6-2 in both our datasets.

C. Hybrid model construction: In this study, we used the hybrid model to enhance the improvement of detection and classifying the DDoS attacks in two different dataset networks. We evaluated all the models using accuracy, precision, recall, ROC curve based on their performance metrics achieved by their accuracy, CNN-LSTM having 99.6%, MLP having 99.67%, LSTM having 99.45 %, CNN-RNN 99.01%, and RNN having 98.69% are ranked second, third, fourth, fifth, and sixth, respectively, with the Hybrid DL of CNN-

MLP having the highest accuracy of 99.89% with feature selection on CICDDoS2019 dataset. Further classified it into categorical attacks of six and four types and this constructed model where the input data is fed to CNN model and does max pooling, dense layer, dropout rate, flatten layers and output is feed to MLP layers having batch normalization, dropout dense and SoftMax activation for the classification purpose and finally the output. We summarized it in Tables 6-3 and 6-4, each evaluation metric of the attacks to show how much each attack had been classified properly and conducted a confusion matrix of data's Table 6-5, where the other models also had high detection and classification rates.

D. **Further tested the scalability and generality:** by using two datasets InSDN and CICDDoS2019 which have small-scale and large-scale datasets of 207,146 and 869,783 respectively, by conducting our three hybrid models we concluded that in both datasets our proposed model CNN-MLP had the highest evaluation metrics in both case of 99.90 and 99.89% not only to see the different data volumes effect also complexities difference and variations in DDoS attack patterns. This provides a more robust assessment of the model's potential for particular deployment in real-world network security systems.

E. **Mininet pox controller:** For the mitigation part of our work, we installed a Mininet and POX controller to adjust our network simulator and see our results. We designed our network with a Miniedit simulator and constructed 3 controllers, 8 switches, and 30 hosts. We have seen the effect of the mitigation to drop out the attacks by IP address by the flow rules adjusted on the switches.

6.5. Research Question Discussion

RQ1. How to identify and preprocess the features of the InSDN and CICDDoS2019 dataset?

As the main thing in the research world is the data in most learning models, we used two well-known public data sets where both were generated in an SDN environment showing different DDoS attack types. We applied all the necessary preprocessing steps and chose the relevant features to detect and classify DDoS attacks. We had done our research and identified the behavior of DDoS from related work baseline papers and literature reviews before we applied the feature selection to identify which features were more relevant than the others. Because we had more than 76 features and not all features are equally important to identify the DDoS

attacks. The most relevant ones were low pkts, Bwdpkts, timestamp_nanosec, pkt size avg, Fwdpktlen min, Ugr flag count, pktleng mean, ACK flag count, avgfwd segment size

To select these features, we applied the chi-square method which provides a better selection of high-weighted features from the datasets. By **our experimental analysis (A, C)** the hybrid model of CNN-MLP with feature selection had the highest accuracy according to our result including low error classification. It classified our attacks further to the known type of attacks that had occurred on the controller. We selected 25 features based on our research and finally trained and tested our models to detect and classify DDoS attacks on SDN controllers.

RQ2. How to design effective hybrid deep learning algorithms to enhance the detection of DDoS attacks on SDN?

More complex detection systems are desperately needed, as the threat of DDoS attacks against SDN setups is growing. Traditional protection measures become less effective as attackers can more easily exploit weaknesses due to SDN's programmable nature and centralized control. As a result, the emphasis now is on using hybrid deep learning techniques to improve DDoS detection and classification on SDN, as **we can see from our experiment (B, C).** These algorithms gave us the capacity to identify complex patterns from large amounts of network data, adjust to changing attack strategies, and deliver real-time insights that were essential for successfully containing DDoS threats in SDN environments.

Therefore, developing efficient hybrid deep learning algorithms that are customized to the special features of SDN networks is essential to strengthening cybersecurity defenses against DDoS attacks. As we mentioned in our literature review, we selected three models and hybrid them with the 1D-CNN model. So, due to their various functionality and related work done on them, a hybrid of CNN-MLP, CNN-LSTM, and CNN-MLP are the most relevant models to detect and classify the DDoS attack types into six and four types.

As network security has become more vulnerable to having a more increasing reliability and high performance in detecting and classifying the attacks, we used a hybrid deep learning model to increase the accuracy.

RQ3. How do we evaluate the results of the model we have chosen?

In our experiment (B, C), we developed three hybrid models of CNN with three different DL models of LSTM, MLP, and RNN and three deep learning models on both InSDN and CICDDoS2019 datasets. After the analysis of our experiments, the hybrid model CNN-MLP with feature selection showed the highest accuracy among the three hybrid models in detecting and classifying DDoS attacks. Our proposed model learns from high-weighted features. We saw the result in small-scale, and large-scale datasets and in both cases, got high accuracy, but got higher from the large-scale datasets which is CICDDoS2019.

Our proposed hybrid model CNN-MLP has 99.89% accuracy, where the hybrid model combines the strength of CNN, which excels at learning complex patterns and is well suited for classification tasks and MLP is a feedforward neural network known for better classification of the DDoS attack types further it has faster training and interface of the data and makes it well suited for the detection and classification of the DDoS attacks. When compared CNN-MLP to the baseline paper, it has improved by 0.48% and the other models too as shown in **Table 6-5.**

The CNN-MLP had a 99.89 accuracy model which has improved by 1.84% which is higher than RNN, 0.84% higher than MLP, 0.32% higher than LSTM, 0.64% higher than CNN-RNN and 0.18% higher than CNN-LSTM. When we compared our work with baseline research, it improved by 0.48%. We included the confusion matrix to show the result of recall, precision, and F1-score describing in detail each attack, and are highly detected and discussed in Figure 6-15 and Table 6-5, including the AUC-ROC curve having a 99.99% rate.

RQ4. How can we mitigate the DDoS attack?

We did our mitigation part by an assumption of having 3 POX controllers for a bottleneck or attacks happening on the main controller there could be backup controllers for the network and packets not to be lost. We considered 8 switches with 30 hosts and assigned 2 hosts like attackers identified the IP address and dropped the packet of the attack happening from those hosts. For the mitigation part, we have dropped the IP address from where the attack happened. The experiment was done on (E) as if we can do further.

CHAPTER SEVEN

CONCLUSION AND FUTURE WORK

7. Conclusion

SDN nowadays represents a revolutionary paradigm in the network area, offering increasing solutions to the limitation of the traditional internet infrastructure. By enabling programmability through a logically centralized control plane, data plane, and application lane. It empowers developers with more flexibility and simplifies application maintenance. However, cybersecurity issues are of greater concern regarding this network infrastructure and one of the well-known attacks is the DDoS attack on the controller plane. Identifying, classifying, and further mitigating these attacks have become more of a concern in the research work areas of the DDoS area.

We proposed a solution to enhance the detection and classification of these DDoS attacks by using a hybrid model of CNN-MLP for more effectiveness and accuracy using InSDN and CICDDoS2019 datasets considering six and four attack types. We have employed LSTM, RNN, MLP, CNN-MLP, CNN-LSTM, and CNN-RNN to ensure great accuracy.

We have preprocessed our datasets and selected our features based on the chi-square method, we also used the common models to detect and classify attacks in cybersecurity. We trained our models by training and testing split (20% testing and 80% training). Finally, we evaluated our experiments by different metrics mostly considering the accuracy then verified that all our models have done great detection and categorical classification of DDoS attack types, where the accuracy of our classification model was 99.89% for CNN-MLP, 98.69% for RNN, 99.45% for LSTM, 99.67% for MLP, 99.6% for CNN-LSTM, 99.01 CNN-RNN. As we can see from our experiment CNN-MLP hybrid model outperforms the other five classification models in both datasets with six and four attack types.

7.1. Major Contribution

Our research work mainly improves the accuracy and efficiency of DDoS attack classification and mitigation and reduces loss and error rates for SDN controllers.

- Application of feature selection techniques to identify high-weighted features, enabling the hybrid model to focus on the most relevant aspect of the data and improve the accuracy.
- A report evaluation of different DDoS attack types in different datasets like CICDDoS2019 and InSDN to show their impacts on scalability and effectiveness in handling real-world, high-volume DDoS attacks.
- A comparison of different deep learning models to analyze the DDoS attack types which are LSTM, MLP, RNN, CNN-MLP, CNN-LSTM, and CNN-MLP mostly based on the accuracy and other metrics to identify and classify it well with baseline research paper and state of the art deep learning.
- We investigated on two different datasets and the hybrid deep learning model has shown improvement in different evaluation metrics we have increased the accuracy by 0.48%.
- We implemented a mitigation strategy to drop the attacks classified well in the controller.

7.2. Future work

In the future, to improve the work try to expand or include all the attack types in the dataset for further training. We recommend the following:

- Try further mitigation techniques using different controllers and observe the effect.
- Try predicting attacks before they occur, whether in a short or long period.
- Generalize the system to work for other attack types coming from the application layer, data layer, or the API.

Special Acknowledgement

This research was sponsored by Adama Science and Technology University with a grant number

ASTU/SM-R/1067/24 Adama, Ethiopia

Reference

AbdulRaheem, M., Oladipo, I. D., Imoize, A. L., Awotunde, J. B., Lee, C. C., Balogun, G. B., & Adeoti, J. O. (2023). Machine learning assisted Snort and Zeek in detecting DDoS attacks in software-defined networking. *International Journal of Information Technology (Singapore)*, *0123456789*. https://doi.org/10.1007/s41870-023-01469-3

Adekoya, O., Aneiba, A., Patwary, M., & Member, S. (2020). *Algorithm for SDN Load Balancing*. *1*(October).

Ahuja, N., Singal, G., & Mukhopadhyay, D. (2021). DLSDN: Deep learning for DDOS attack detection in software-defined networking. *Proceedings of the Confluence 2021: 11th International Conference on Cloud Computing, Data Science and Engineering*, 683–688. https://doi.org/10.1109/Confluence51648.2021.9376879

Ahuja, N., Singal, G., Mukhopadhyay, D., & Kumar, N. (2021). Automated DDOS attack detection in software-defined networking. *Journal of Network and Computer Applications*, *187*(May), 103108. https://doi.org/10.1016/j.jnca.2021.103108

Alashhab, A. A., Zahid, M. S. M., Azim, M. A., Daha, M. Y., Isyaku, B., & Ali, S. (2022). A Survey of Low Rate DDoS Detection Techniques Based on Machine Learning in Software-Defined Networks. *Symmetry*, *14*(8), 1–30. https://doi.org/10.3390/sym14081563

Aldaoud, M., Al-Abri, D., Awadalla, M., & Kausar, F. (2023). Leveraging ICN and SDN for Future Internet Architecture: A Survey. *Electronics (Switzerland)*, *12*(7). https://doi.org/10.3390/electronics12071723

Ali, M. N., Imran, M., din, M. S. ud, & Kim, B. S. (2023). Low Rate DDoS Detection Using Weighted Federated Learning in SDN Control Plane in IoT Network. *Applied Sciences (Switzerland)*, *13*(3). https://doi.org/10.3390/app13031431

Ali, T. E., Chong, Y. W., & Manickam, S. (2023). Machine Learning Techniques to Detect a DDoS Attack in SDN: A Systematic Review. *Applied Sciences (Switzerland)*, *13*(5). https://doi.org/10.3390/app13053183

Amin, R., Reisslein, M., & Shah, N. (2018). Hybrid SDN networks: A survey of existing approaches. *IEEE Communications Surveys and Tutorials*, *20*(4), 3259–3306. https://doi.org/10.1109/COMST.2018.2837161

Aslam, N., Srivastava, S., & Gore, M. M. (2023). ONOS DDoS Defender: A Comparative

Analysis of Existing DDoS Attack Datasets using Ensemble Approach. *Wireless Personal Communications*, *133*(3), 1805–1827. https://doi.org/10.1007/s11277-023-10848-9

Banitalebi Dehkordi, A., Soltanaghaei, M. R., & Boroujeni, F. Z. (2021). The DDoS attacks detection through machine learning and statistical methods in SDN. In *Journal of Supercomputing* (Vol. 77, Issue 3). Springer US. https://doi.org/10.1007/s11227-020-03323-w

Bawany, N. Z., Shamsi, J. A., & Salah, K. (2017). DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions. *Arabian Journal for Science and Engineering*, *42*(2), 425–441. https://doi.org/10.1007/s13369-017-2414-5

Bholebawa, I. Z., Jha, R. K., & Dalal, U. D. (2016). Performance Analysis of Proposed Network Architecture: OpenFlow vs. Traditional Network. *Wireless Personal Communications*, *86*(2), 943–958. http://link.springer.com/10.1007/s11277-015-2963-4

Bhuiyan, Z. A., Islam, S., Islam, M. M., Ullah, A. B. M. A., Naz, F., & Rahman, M. S. (2023). On the (in)Security of the Control Plane of SDN Architecture: A Survey. *IEEE Access*, *11*(August), 91550–91582. https://doi.org/10.1109/ACCESS.2023.3307467

Das, S., Venugopal, D., Shiva, S., & Sheldon, F. T. (2020). Empirical Evaluation of the Ensemble Framework for Feature Selection in DDoS Attack. *Proceedings - 2020 7th IEEE International Conference on Cyber Security and Cloud Computing and 2020 6th IEEE International Conference on Edge Computing and Scalable Cloud, CSCloud-EdgeCom 2020*, *Ml*, 56–61. https://doi.org/10.1109/CSCloud-EdgeCom49738.2020.00019

Elsayed, M. S., Le-Khac, N.-A., & Jurcut, A. D. (2020a). *Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000. InSDN: A Novel SDN Intrusion Dataset*. *8*.

Elsayed, M. S., Le-Khac, N. A., Dev, S., & Jurcut, A. D. (2020). DDoSNet: A Deep-Learning Model for Detecting Network Attacks. *Proceedings - 21st IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2020*, 391–396. https://doi.org/10.1109/WoWMoM49955.2020.00072

Elsayed, M. S., Le-Khac, N. A., & Jurcut, A. D. (2020b). InSDN: A novel SDN intrusion dataset. *IEEE Access*, *8*, 165263–165284. https://doi.org/10.1109/ACCESS.2020.3022633

Gadze, J. D., Bamfo-Asante, A. A., Agyemang, J. O., Nunoo-Mensah, H., & Opare, K. A. B. (2021). An Investigation into the Application of Deep Learning in the Detection and Mitigation of DDOS Attack on SDN Controllers. *Technologies*, *9*(1), 1–22. https://doi.org/10.3390/technologies9010014

Gebremeskel, T. G., Gemeda, K. A., Krishna, T. G., & Ramulu, P. J. (2023). DDoS Attack Detection and Classification Using Hybrid Model for Multicontroller SDN. *Wireless Communications and Mobile Computing*, *2023*, 1–18. https://doi.org/10.1155/2023/9965945

Habib, B., & Khursheed, F. (2024). Time-based DDoS attack detection through hybrid LSTM-CNN model architectures: An investigation of many-to-one and many-to-many approaches. *Concurrency and Computation: Practice and Experience*, *36*(9), 1–32. https://doi.org/10.1002/cpe.7996

Hassan, H. A., Hemdan, E. E., El-Shafai, W., Shokair, M., & Abd El-Samie, F. E. (2024). Detection of attacks on software-defined networks using machine learning techniques and imbalanced data handling methods. *Security and Privacy*, *7*(2), 1–14. https://doi.org/10.1002/spy2.350

HUSSEİN, M., & ÖZYURT, F. (2021). A New Technique for Sentiment Analysis System Based on Deep Learning Using Chi-Square Feature Selection Methods. *Balkan Journal of Electrical and Computer Engineering*, *9*(4), 320–326. https://doi.org/10.17694/bajece.887339

Jeon, H., & Oh, S. (2020). Applied sciences Hybrid-Recursive Feature Elimination for E ffi cient Feature Selection. *Applied Sciences*, 1–8.

Joëlle, M. M., & Park, Y. H. (2018). Strategies for detecting and mitigating DDoS attacks in SDN: A survey. *Journal of Intelligent and Fuzzy Systems*, *35*(6), 5913–5925. https://doi.org/10.3233/JIFS-169833

Kanber, B. M., Noaman, N. F., Saeed, A. M. H., & Malas, M. (2022). DDoS Attacks Detection in the Application Layer Using Three Level Machine Learning Classification Architecture. *International Journal of Computer Network and Information Security*, *14*(3), 33–46. https://doi.org/10.5815/ijcnis.2022.03.03

Khaire, U. M., & Dhanalakshmi, R. (2022). Stability of feature selection algorithm: A review. *Journal of King Saud University - Computer and Information Sciences*, *34*(4), 1060–

1073. https://doi.org/10.1016/j.jksuci.2019.06.012

Khempetch, T., & Wuttidittachotti, P. (2021). DDoS attack detection using deep learning. *IAES International Journal of Artificial Intelligence*, *10*(2), 382–388. https://doi.org/10.11591/ijai.v10.i2.pp382-388

Kshirsagar, D., & Kumar, S. (2022). A feature reduction-based reflected and exploited DDoS attack detection system. *Journal of Ambient Intelligence and Humanized Computing*, *13*(1), 393–405. https://doi.org/10.1007/s12652-021-02907-5

Li, J., Tengfei, T., Yongsheng, L., Sujuan, Q., Yijie, S., & Qiaoyan, W. (2022). DoSGuard : Mitigating Denial-of-Service Attacks in. *Sensors*, *22*(3), 1–17. https://www.mdpi.com/1424-8220/22/3/1061

Mansoor, A., Anbar, M., Bahashwan, A. A., Alabsi, B. A., & Rihan, S. D. A. (2023). Deep Learning-Based Approach for Detecting DDoS Attack on Software-Defined Networking Controller. *Systems*, *11*(6), 1–21. https://doi.org/10.3390/systems11060296

Mittal, M., Kumar, K., & Behal, S. (2023). Deep learning approaches for detecting DDoS attacks: a systematic review. *Soft Computing*, *27*(18), 13039–13075. https://doi.org/10.1007/s00500-021-06608-1

Mousavi, S. M., & St-Hilaire, M. (2015). Early detection of DDoS attacks against SDN controllers. *2015 International Conference on Computing, Networking and Communications, ICNC 2015*, 77–81. https://doi.org/10.1109/ICCNC.2015.7069319

Munir, M., Ardiansyah, I., Santoso, J. D., Mustopa, A., & Mulyatun, S. (2022). Detection and Mitigation of Distributed Denial of Service Attacks on Network Architecture Software Defined Networking Using the Naive Bayes Algorithm. *Journal of Information System Management (JOISM)*, *3*(2), 51–55. https://doi.org/10.24076/joism.2022v3i2.656

Murtuza, S., & Asawa, K. (2024). Early Prevention and Mitigation of Link Flooding Attacks in Software Defined Networks. *Journal of Network and Computer Applications*, *224*(November 2023), 103832. https://doi.org/10.1016/j.jnca.2024.103832

Myint Oo, M., Kamolphiwong, S., Kamolphiwong, T., & Vasupongayya, S. (2019). Advanced Support Vector Machine-(ASVM-) based detection for Distributed Denial of Service (DDoS) attack on Software Defined Networking (SDN). *Journal of Computer Networks and Communications*, *2019*. https://doi.org/10.1155/2019/8012568

Nadeem, M. W., Goh, H. G., Ponnusamy, V., & Aun, Y. (2022). Ddos detection in SDN

using machine learning techniques. *Computers, Materials and Continua*, *71*(1), 771–789. https://doi.org/10.32604/cmc.2022.021669

Najar, A. A., & Manohar Naik, S. (2022). DDoS attack detection using MLP and Random Forest Algorithms. *International Journal of Information Technology (Singapore)*, *14*(5), 2317–2327. https://doi.org/10.1007/s41870-022-01003-x

Nasir, I. M., Khan, M. A., Yasmin, M., Shah, J. H., Gabryel, M., Scherer, R., & Damaševičius, R. (2020). Pearson correlation-based feature selection for document classification using balanced training. *Sensors (Switzerland)*, *20*(23), 1–18. https://doi.org/10.3390/s20236793

Neira, A. B. De, Araujo, A. M. De, & Nogueira, M. (2023). An Intelligent System for DDoS Attack Prediction Based on Early Warning Signals. *IEEE Transactions on Network and Service Management*, *20*(2), 1254–1266. https://doi.org/10.1109/TNSM.2022.3223881

Nowsin, M., Sheikh, A., Halder, M., Kabir, S. S., Wasim, M., Khatun, S., Shalauddin Kabir, S., & Miah, Ꞡ. (2019). *SDN-Based Approach to Evaluate the Best Controller: Internal Controller NOX and External Controllers POX, ONOS, RYU SDN-BasedApproachtoEvaluatetheBestControllerInternalControllerNOXandExternalControllersPOXONOSRYU SDN-Based Approach to Evaluate the Best C*. *19*(1).

Nugraha, B., & Murthy, R. N. (2020). Deep Learning-based Slow DDoS Attack Detection in SDN-based Networks. *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks, NFV-SDN 2020 - Proceedings*, 51–56. https://doi.org/10.1109/NFV-SDN50289.2020.9289894

Oluwasogo Adekunle, O. (2015). A Security Architecture for Software Defined Networks (SDN). *IJCSIS) International Journal of Computer Science and Information Security*, *13*(7), 56–61. http://sites.google.com/site/ijcsis/

Pandithurai, O., Venkataiah, C., Tiwari, S., & Ramanjaneyulu, N. (2024). DDoS attack prediction using a honey badger optimization algorithm based feature selection and Bi-LSTM in a cloud environment. *Expert Systems with Applications*, *241*(November 2023), 122544. https://doi.org/10.1016/j.eswa.2023.122544

Rawat, D. B., & Reddy, S. R. (2017). Software Defined Networking Architecture, Security and Energy Efficiency: A Survey. *IEEE Communications Surveys and Tutorials*, *19*(1), 325–346. https://doi.org/10.1109/COMST.2016.2618874

Roopak, M., Tian, G. Y., & Chambers, J. (2020). An Intrusion Detection System Against DDoS Attacks in IoT Networks. *2020 10th Annual Computing and Communication Workshop and Conference, CCWC 2020*, 562–567. https://doi.org/10.1109/CCWC47524.2020.9031206

Sahoo, K. S., Tripathy, B. K., Naik, K., Ramasubbareddy, S., Balusamy, B., Khari, M., & Burgos, D. (2020). An Evolutionary SVM Model for DDOS Attack Detection in Software Defined Networks. *IEEE Access*, *8*, 132502–132513. https://doi.org/10.1109/ACCESS.2020.3009733

Said Elsayed, M., Le-Khac, N. A., Dev, S., & Jurcut, A. D. (2020). Network Anomaly Detection Using LSTM-Based Autoencoder. *Q2SWinet 2020 - Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 37–45. https://doi.org/10.1145/3416013.3426457

Santos, R., Souza, D., Santo, W., Ribeiro, A., & Moreno, E. (2020). Machine learning algorithms to detect DDoS attacks in SDN. *Concurrency and Computation: Practice and Experience*, *32*(16), 1–14. https://doi.org/10.1002/cpe.5402

Seba, A. M., Gemeda, K. A., & Ramulu, P. J. (2024). Prediction and classification of IoT sensor faults using a hybrid deep learning model. *Discover Applied Sciences*, *6*(1). https://doi.org/10.1007/s42452-024-05633-7

Sekar, R. R., Jenny, A. M., Sreshta, D., Vikas, M., Ajay, D. B. N., & Ganesh, M. (2023). Prediction of Distributed Denial of Service Attacks in SDN using Machine Learning Techniques. *2023 3rd International Conference on Intelligent Technologies, CONIT 2023*, 1–5. https://doi.org/10.1109/CONIT59222.2023.10205887

Singh, A., Kaur, H., & Kaur, N. (2023). A novel DDoS detection and mitigation technique using a hybrid machine learning model and redirects illegitimate traffic in the SDN network. *Cluster Computing*, *1*. https://doi.org/10.1007/s10586-023-04152-1

Singh, J., & Behal, S. (2020). Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions. *Computer Science Review*, *37*, 100279. https://doi.org/10.1016/j.cosrev.2020.100279

Singh, K., Scholar, R., Mahajan, A., & Mansotra, V. (2021). 1D-CNN-based Model for Classification and Analysis of Network Attacks. *International Journal of Advanced Computer Science and Applications*, *12*(11), 604–613.

https://doi.org/10.14569/IJACSA.2021.0121169

Songma, S., Sathuphan, T., & Pamutha, T. (2023). Optimizing Intrusion Detection Systems in Three Phases on the CSE-CIC-IDS-2018 Dataset. *Computers*, *12*(12). https://doi.org/10.3390/computers12120245

Sumathi, S., Rajesh, R., & Lim, S. (2022). Recurrent and Deep Learning Neural Network Models for DDoS Attack Detection. *Journal of Sensors*, *2022*. https://doi.org/10.1155/2022/8530312

Sun, W., Guan, S., Wang, P., & Wu, Q. (2022). A hybrid deep learning model based low-rate DoS attack detection method for software defined network. *Transactions on Emerging Telecommunications Technologies*, *33*(5), 1–17. https://doi.org/10.1002/ett.4443

Taheri, R., Ahmed, H., & Arslan, E. (2023a). Deep learning for the security of software-defined networks: a review. *Cluster Computing*, *26*(5), 3089–3112. https://doi.org/10.1007/s10586-023-04069-9

Taheri, R., Ahmed, H., & Arslan, E. (2023b). Deep learning for the security of software-defined networks: a review. In *Cluster Computing* (Vol. 26, Issue 5, pp. 3089–3112). https://doi.org/10.1007/s10586-023-04069-9

Thakur, K., Sain, D., Scholar, Mt., & Professor, A. (2022). A Literature Review on DDOS Attack Detection Using Artificial Neural Network. *JOURNAL OF COMPUTING TECHNOLOGIES (JCT) International Journal*, *3404*(11), 77–85. http://www.jctjournals.com

Turner, S. W., Karakus, M., Guler, E., & Uludag, S. (2023). A Promising Integration of SDN and Blockchain for IoT Networks: A Survey. *IEEE Access*, *11*(March), 29800–29822. https://doi.org/10.1109/ACCESS.2023.3260777

Umar, M. A., Zhanfang, C., & Liu, Y. (2020). Network Intrusion Detection Using Wrapper-based Decision Tree for Feature Selection. *ACM International Conference Proceeding Series*, 5–13. https://doi.org/10.1145/3424311.3424330

Vedhapriyavadhana, R., Francy Irudaya Rani, E., & Theepa, M. (2018). Simulation and performance analysis of Security issue Using Floodlight controller in Software Defined Network. *2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research, ICETIETR 2018*, 1–6. https://doi.org/10.1109/ICETIETR.2018.8529143

Yeom, S., Choi, C., & Kim, K. (2022). LSTM-Based Collaborative Source-Side DDoS Attack Detection. *IEEE Access*, *10*, 44033–44045. https://doi.org/10.1109/ACCESS.2022.3169616

Z. long, W.Jinsong ”A hybrid method of entropy and SSAE-SVM based DDoS detection and mitigation mechanism in SDN”, Computers and security, vol. 115,2022, doi: 10.1016/j.cose.2022.102604

Appendix

Code for CNN-MLP hybrid model starting from preprocessing

```
 # importing libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
from sklearn import preprocessing
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from tensorflow import keras
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout, SimpleRNN
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot as plt
import seaborn as sns
from keras.layers import LeakyReLU
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping
import tensorflow as tf
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten
from tensorflow.keras.optimizers import Adadelta, Adam

        # mount data

from google.colab import drive
drive.mount('/content/drive')

        # integrating of data

UPDLag_data = pd.read_csv('/content/drive/MyDrive/UDPLag.csv',nrows =
number_of_samples, low_memory=False)
Syn_data = pd.read_csv('/content/drive/My Drive/Syn.csv', nrows =
number_of_samples,low_memory=False)
```

```
DrDoS_NTP_data = pd.read_csv('/content/drive/My Drive/DrDoS_NTP.csv', nrows =
number_of_samples, low_memory=False)
DrDoS_LDAP_data = pd.read_csv('/content/drive/My Drive/DrDoS_LDAP.csv', nrows =
number_of_samples, low_memory=False)
data =
pd.concat([Syn_data,DrDoS_NTP_data,DrDoS_LDAP_data,UPDLag_data],ignore_index=Tr
ue)

        # converting the timestamp to nanosecond

datax = data[' Timestamp']
z = pd.DataFrame(data)
z[' Timestamp'] = pd.to_datetime(z[' Timestamp'], format='%Y-%m-%d %H:%M:%S.%f',
errors='coerce')
z = z.dropna(subset=[' Timestamp'])
unix_timestamp_seconds = z[' Timestamp'].apply(lambda x: x.timestamp())

 # Convert Unix timestamp to nanoseconds

timestampnano = unix_timestamp_seconds * 1e9
nanoseconds_df = pd.DataFrame({'TimestampNanosec': timestampnano}, index=z.index)

   # Concatenate the nanoseconds column with the original dataset

final_df = pd.concat([z, nanoseconds_df], axis=1)

   # Drop the original timestamp column

final_df = final_df.dropna(subset=[' Timestamp'])
final_df = final_df.drop(columns=[' Timestamp'])

print(final_df)

    # removing features

data = final_df.drop([' Source IP', ' Source Port', 'SimillarHTTP', ' Destination IP', '
Destination Port','Unnamed: 0','Flow ID'], axis = 1)

   # cleaning of data

data_re = data.replace(np.inf, np.nan)
data_re.isnull().sum().sum()
data_dff = data_re.dropna(axis = 0)
data_dff.isnull().sum().sum()

  # Normalizing of the data

X=data_dff.drop([' Label'], axis=1)
Y=data_dff[' Label']
```

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
scaler.fit(X)
scaled_train = scaler.transform(X)
scaled_data_fit_transform = scaler.fit_transform(X)
  # labeling the target values
categories = np.unique(Y)

# Create an instance of the OneHotEncoder

encoder = OneHotEncoder(categories=[categories],sparse=False)

# Define the input labels

labels = ['DrDoS_NTP', 'BENIGN', 'DrDoS_LDAP', 'SYN', 'UDP-lag', 'WebDDoS']

# Convert the categorical labels to numerical data using one-hot encoding

encoded_labels = encoder.fit_transform(np.array(Y).reshape(-1, 1))

# feature selection by chi-square
feature_names = X.columns
selected_feature_names = [feature_names[i] for i in selected_feature_indices]

# Print the list of selected feature names
print("Selected feature names:", selected_feature_names)
# Get the chi-square scores and p-values
scores = selector.scores_
p_values = selector.pvalues_

# Create a DataFrame to store feature scores and p-values
feature_scores = pd.DataFrame({'Feature': feature_names, 'Score': scores,'PVALUE':
p_values})

# Sort the DataFrame based on feature scores in descending order
feature_scores = feature_scores.sort_values(by='Score', ascending=False)

# Print the top features and their scores
print("Top", k, "Features:")
print(feature_scores.head(k))
# split our preprocessed data into x and y
upper_limit = 6
X = processed.drop([f'Label_{i}' for i in range(upper_limit)], axis=1)
Y = processed[[f'Label_{i}' for i in range(upper_limit)]]

# train and test split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

```
W = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))  # Assuming input_dim is 1
Z = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
Y_train_reshaped = np.reshape(Y_train, (Y_train.shape[0], Y_train.shape[1], 1))
Y_test_reshaped = np.reshape(Y_test, (Y_test.shape[0], Y_test.shape[1], 1))
# Convert the target data to one-hot encoding
num_classes = len(np.unique(Y_train))  # Number of unique classes in the target variable
Y_train_encoded = to_categorical(Y_train, num_classes=num_classes)
Y_test_encoded = to_categorical(Y_test, num_classes=num_classes)
# traning of the CNN-MLP model
early_stopping = EarlyStopping(patience=3)
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
modelCM = Sequential()

# Add 1D CNN layers
modelCM.add(Conv1D(256, kernel_size=3, activation='relu', input_shape=(W.shape[1], 1)))
modelCM.add(MaxPooling1D(pool_size=2))
modelCM.add(Conv1D(128, kernel_size=3, activation='relu'))
modelCM.add(MaxPooling1D(pool_size=2))
modelCM.add(Conv1D(64, kernel_size=3, activation='relu'))
modelCM.add(Flatten())

# Add MLP layers
modelCM.add(Dense(128, activation='relu'))
modelCM.add(BatchNormalization())
modelCM.add(Dropout(0.1))
modelCM.add(Dense(64, activation='relu'))
modelCM.add(Dense(32, activation='relu'))
modelCM.add(Dropout(0.04))
modelCM.add(Dense(Y.shape[1], activation='sigmoid'))
# Compile and train the model
learning_rate=0.001
optmizer=Adam(learning_rate=learning_rate)
optimizer = tf.keras.optimizers.Adam()
# Manually build the optimizer with all trainable variables
optimizer.build(modelCM.trainable_variables)
modelCM.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])
#modelCM.fit(X_train, Y_train, validation_data=(X_test, Y_test), epochs=300,
batch_size=100, callbacks=[early_stopping])
CM = modelCM.fit(X_train, Y_train, batch_size=1000, epochs=30, validation_split=0.2)
#modelCM.fit(X_train, Y_train, batch_size=500, epochs=34, validation_split=0.3)
#modelCM.fit(X_train, Y_train, batch_size=1000, epochs=33, validation_split=0.2)
# Evaluate the model
_, accuracy = modelCM.evaluate(X_test, Y_test)
print('Accuracy: %.2f' % (accuracy * 100))
```