

KLE Society's  
KLE Technological University, Hubballi.



A Minor Project Report  
on  
**DDoS Attack Detection in SDN environment**  
*submitted in partial fulfillment of the requirement for the degree of*

Bachelor of Engineering  
in  
Computer Science and Engineering

Submitted by

Madhukeshwar Hegde	01FE18BCS107
Vinay S Itagi	01FE18BCS256
Rashmi	01FE18BCS281
Mayur S Javali	01FE18BCS288

Under the guidance of  
Ms. Pooja Shettar

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Hubballi – 580 031

2020 -21

KLE Society's

KLE Technological University, Hubballi.

2020 - 2021



SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

## CERTIFICATE

This is to certify that Minor Project titled DDoS Attack Detection in SDN environment is a bonafied work carried out by the student team comprising of Madhukeshwar Hegde (01FE18BCS107), Vinay S Itagi (01FE18BCS256) , Rashmi (01FE18BCS281), Mayur S Javali (01FE18BCS288) for partial fulfillment of completion of sixth semester B.E. in Computer Science and Engineering during the academic year 2020-21.

Guide

Head, SoCSE

Ms. Pooja Shettar

Dr. Meena S. M

Viva -Voce:

Name of the Examiners

Signature with date

1.

2.

# ABSTRACT

Security is a critical concern for the networking devices. Distributed Denial of Service (DDoS) attacks are the most frequent of all network security threats, and the consequences of damage caused by DDoS are very serious. Thus, the design of an efficient DDoS detection system plays an important role in monitoring suspicious activity in the network. Software-defined networking is a fast-growing technology and has been a huge success because the control plane separates from the data plane of the networking devices. This is remarkable progress in the field of networking has helped to accelerate service delivery and provide more agility to provide virtual and physical network devices from a central location. But this SDN is threatened by a DDoS attack because the entire controller is down. Distributed Denial of service occurs when an attacker uses multiple machines and bots to flood target host with a large number of TCP requests causing denial (eg SYN flood) service. With the proliferation of DDoS attacks in several countries, the need to deploy is a protective mechanism against this is highly significant. In this work, we propose a real time detection of a DDoS attacks using BRNN model. We compare the classification performance using benchmark and real time datasets. Results of the experiments reveal that the Bi-directional Recurrent neural network (BRNN) offers better classifier accuracy.

**Keywords :** *SDN, DDOS attack, BRNN model.*

# ACKNOWLEDGEMENT

We would like to thank our faculty and management for their professional guidance towards the completion of the project work. We take this opportunity to thank Dr. Ashok Shettar, Vice-Chancellor, Dr. N.H Ayachit, Registrar, and Dr. P.G Tewari, Dean Academics, KLE Technological University, Hubballi, for their vision and support. We also take this opportunity to thank Dr. Meena S. M, Professor and Head, SoCSE for having provided us direction and facilitated for enhancement of skills and academic growth. We thank our guide Ms. Pooja Shettar, SoCSE for the constant guidance during interaction and reviews. We extend our acknowledgement to the reviewers for critical suggestions and inputs. We also thank Project Co-ordinator Dr. Sujatha C. and faculty in-charges for their support during the course of completion. We express gratitude to our beloved parents for constant encouragement and support.

Madhukeshwar Hegde - 01FE18BCS107

Vinay S Itagi - 01FE18BCS256

Rashmi - 01FE18BCS281

Mayur S Javali - 01FE18BCS288

# CONTENTS

<b>ABSTRACT</b>	<b>i</b>
<b>ACKNOWLEDGEMENT</b>	<b>ii</b>
<b>CONTENTS</b>	<b>iv</b>
<b>LIST OF TABLES</b>	<b>v</b>
<b>LIST OF FIGURES</b>	<b>vi</b>
<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Background study . . . . .	2
1.2.1 Software-Defined Network . . . . .	2
1.2.2 DDoS attack . . . . .	4
1.2.3 RYU controller . . . . .	6
1.2.4 Related work . . . . .	7
1.3 Problem Statement . . . . .	9
1.4 Applications . . . . .	9
1.5 Objectives and Scope of the project . . . . .	9
1.5.1 Objectives . . . . .	9
1.5.2 Scope of the project . . . . .	9
<b>2 REQUIREMENT ANALYSIS</b>	<b>10</b>
2.1 Functional Requirements . . . . .	10
2.2 Non Functional Requirements . . . . .	10
2.3 Hardware Requirements . . . . .	10
2.4 Software Requirements . . . . .	11
<b>3 SYSTEM DESIGN</b>	<b>12</b>
3.1 Proposed system . . . . .	12
3.2 Advantages of Proposed system . . . . .	13
3.3 Level zero DFD . . . . .	14
<b>4 IMPLEMENTATION</b>	<b>15</b>
4.1 Proposed Deep-learning models . . . . .	15

4.1.1	Feed forward neural network . . . . .	15
4.1.2	Bi-directional neural network . . . . .	15
4.2	Training Data Collection . . . . .	16
4.2.1	How we have collected Data? . . . . .	16
4.2.2	RAW Dataset . . . . .	17
4.3	Attributes used . . . . .	18
4.4	Feature analysis . . . . .	20
4.4.1	Dataset Description . . . . .	23
<b>5</b>	<b>RESULTS AND DISCUSSIONS</b>	<b>24</b>
5.1	Topology . . . . .	24
5.2	Splitting of training and testing samples . . . . .	25
5.3	Feed Forward Neural Network . . . . .	25
5.4	Bi-directional Neural Network . . . . .	26
5.5	Comparison of accuracy of Deep-learning models . . . . .	27
5.6	Performance comparison of deep-learning models . . . . .	28
5.7	Confusion matrix of BRNN model . . . . .	29
5.8	Normal Traffic . . . . .	30
5.9	Attack Traffic . . . . .	31
<b>6</b>	<b>CONCLUSION AND FUTURE SCOPE</b>	<b>32</b>
6.1	Conclusion . . . . .	32
6.2	Future scope . . . . .	32
	<b>REFERENCES</b>	<b>36</b>

# LIST OF TABLES

5.1	Splitting of Training and testing samples . . . . .	25
5.2	Comparison of accuracy and Error rate . . . . .	27

# LIST OF FIGURES

1.1	SDN architecture . . . . .	3
3.1	DDoS detection system . . . . .	12
3.2	Level zero DFD . . . . .	14
4.1	BRNN . . . . .	15
4.2	RAW dataset . . . . .	17
4.3	SSIP formulae . . . . .	18
4.4	SDFP formulae . . . . .	18
4.5	SDFB formulae . . . . .	19
4.6	SFE formulae . . . . .	19
4.7	REIP formulae . . . . .	19
4.8	speed of source IP address . . . . .	20
4.9	standard deviation of flow bytes . . . . .	20
4.10	standard deviation of flow packets . . . . .	21
4.11	Speed of flow entries . . . . .	21
4.12	Ratio of pair flow entries . . . . .	22
4.13	Description of dataset . . . . .	23
5.1	Topology . . . . .	24
5.2	FFNN model Accuracy . . . . .	25
5.3	FFNN model Loss . . . . .	26
5.4	BRNN model Accuracy . . . . .	26
5.5	BRNN model loss . . . . .	27
5.6	Accuracy graph . . . . .	28
5.7	Confusion matrix for BRNN . . . . .	29
5.8	Normal traffic. . . . .	30
5.9	Detection of incoming traffic when there is attack. . . . .	31



# Chapter 1

## INTRODUCTION

A Software-Defined Network(SDN) is a dynamic, manageable, flexible, and cost-effective architecture that is suited to today's high-bandwidth, dynamic applications. Software-defined network (SDN) security has to be implemented everywhere within SDNs. To protect the confidentiality, availability, and integrity of all connected resources, SDN security has to be built into the architecture and offered as a service.

Denial-of-Service (DoS) attacks are cyber-attacks in which an attacker attempts to make a machine or network resource unavailable to its intended users by temporarily or permanently interrupting its services. DDoS attacks are one of the major threats that are faced in network security. The main strategy behind a DDoS attack is to target the victim by submitting a lot of requests in a distributed manner in order to exhaust the victim's resources, and therefore rendering the target resources unavailable, over a specific period of time, to the legitimate users. Classification algorithms are utilized to classify traffic packets and to predict intrusions. Selection of the appropriate deep-learning algorithm for each dataset is an important issue. For accurate classification of data, one must select the right classifier. Thus, the estimation of the algorithms and the comparison of their performance are necessary while choosing the classifier. Furthermore, there is a trade-off between choosing classifiers based on their accuracy, precision and computational time. This motivates researchers to seek new approaches enabling more efficient and faster detection of DDoS attacks.

### 1.1 Motivation

DDoS attacks aim to disrupt the normal functioning of a server, service, or network by flooding the target or its infrastructure with an overwhelming amount of Internet traffic. The SDN architecture is more prone to DDoS attacks than the convolution networks currently in use because the flow rules are added to the switch continually, and eventually, the switch's flow table is full, requiring the controller to process all packets arriving on the switch and the controller will go down resulting in the whole network to go down. Since SDN is centralized, the controller is targeted by attackers to take down the whole network. Centralized architectures of SDN are advantageous for deploying a solution across the network.

## 1.2 Background study

### 1.2.1 Software-Defined Network

A Software-Defined Network is a dynamic, manageable, flexible, and cost-effective architecture that is suited to today's high-bandwidth, dynamic applications. With this architecture, the network control and forwarding functions are decoupled, enabling the control of the network to become directly programmable and the underlying infrastructure to be abstracted for applications and network services.

There are two main types of planes:

- Data plane
- Control plane

The data plane participates in the activities and is benefited by the data packets sent by the end user. Various functions may be performed, such as the replication of packets for multiple casting and the forwarding of data packets. This is the primary layer of the network, also known as the brain of the network. It consists of all activities necessary to perform data plane activities. It does not involve end user packets.

Traditional networks employ data planes and control planes for each switch. According to the control plane of all switches, the forwarding table is constructed as the switches exchange topology information. Packets are forwarded based on the forwarding table. The switches do not have a controller, as SDN has a centralized control unit called SDN controller.

A flow table consists of match fields like an input port range and therefore the packet header with directions. Then the directions of the flow entry get dead. The directions within the flow entry may be forwarding a packet to a specific port or multiple ports or dropping that packet or adding headers to the packet. If there's no match has found of a specific packet then the switch queries the controller that sends in a very new flow entry to the switch.

## SDN ARCHITECTURE

The origin of SDN is followed to a quest coordinated an effort between Stanford University and also the University of California at Berkeley that ultimately yielded the Open flow convention within 2008 period. Open flow is one in every of the foremost necessary options in SDN which may be a programmable network protocol that help in managing and guiding traffic between routers and switches. Some distinguished key sellers embrace CISCO, Juniper, and VMware.

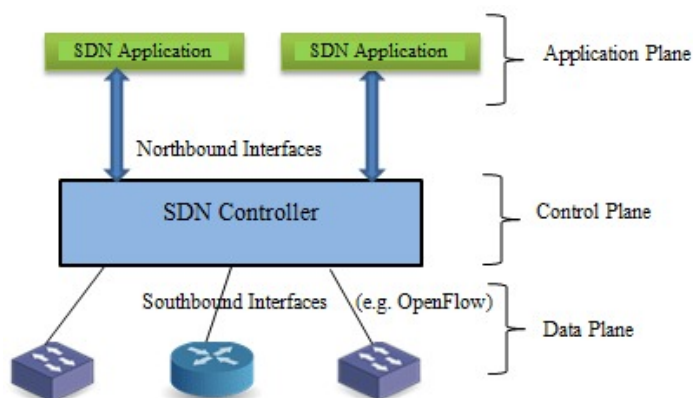


Figure 1.1: SDN architecture

SDN architecture consists of three layers:

1. **Application layer:** it consists of network applications such as intrusion detection, firewall and load balancing.
2. **Control layer:** It consists of the SDN controller which is the main part of the network. It allows hardware abstraction to the applications written on top of it.
3. **Infrastructure layer:** This consists of physical switches and carries out the packet's movement.

These three layers have APIs to communicate with each other:

1. **North bound API:** this is used to communicate between application layer and control layer.
2. **South bound API:** this is used to communicate between control layer and infrastructure layer.

Advantages of SDN:

- The network is programmable which implied we are able to simply edit or modify through a controller instead of individual switches. It's simply programmable as a result of it's decoupled from the forwarding functions.
- The administrator can dynamically adjust the network-wide traffic flow by removing a control from the forwarding functions.
- The network managers of SDN can manage, secure the network, configure it and optimize the resources easily and quickly through dynamic and automated SDN programs which could be written by the network manager or the administrator.
- Applications can be written on the top of the controller.
- More Secured and easy to manage as compared to the traditional network.

Disadvantage of SDN:

- There are high chances of single point failures in centrally based networks, as the controller can become corrupted, causing the whole network to malfunction.

### 1.2.2 DDoS attack

A distributed denial-of-service attack may be a malicious try to disrupt normal traffic of a targeted server, service, or network by overwhelming the target or its surrounding infrastructure with a flood of Internet traffic. Exploited machines can include computers and other networked resources like IoT devices. From a high level, a DDoS attack is sort of a hold up clogging up with highway, preventing regular traffic from arriving at its desired destination.

1. DNS Flood : Domain Name System (DNS) servers are the “phonebooks” of the Internet; they're the trail through which Internet devices can lookup specific web servers to access Internet content. A DNS flood may be a form of DDoS where an attacker floods a specific domain's DNS servers in an endeavor to disrupt DNS resolution for that domain. If a user is unable to search out the phonebook, it cannot find the address to create the decision for a selected resource. By disrupting DNS resolution, a DNS flood attack will compromise an internet site, API, or web application's ability to retort to legitimate traffic. DNS flood attacks will be difficult to differentiate from normal heavy traffic because the big volume of traffic often comes from a mess of unique locations, querying for real records on the domain, mimicking legitimate traffic.

2. SYN Flood : A SYN flood (half-open attack) a kind of DoS attack which aims to create a server unavailable to legitimate traffic by consuming all available server resources. By repeatedly sending initial connection request (SYN) packets, the attacker can overwhelm all available ports on a targeted server machine, causing the targeted device to retort to legitimate traffic sluggishly or not in the least.

3. UDP Flood : A UDP flood is a type of denial-of-service attack in which many numerous User Data gram Protocol packets are sent to a targeted server to overwhelm that device's ability to process and respond. The firewall protecting the targeted server can even become exhausted as a result of UDP flooding, leading to a denial-of-service to legitimate traffic.

4. Ping of Death : A Ping of Death attack a denial-of-service attack, within which the attacker aims to disrupt a targeted machine by sending a packet larger than the utmost allowable size, causing the target machine to freeze or crash. The initial Ping of Death attack is a smaller amount common today. A related attack referred to as an ICMP flood attack is more prevalent.

### 1.2.3 RYU controller

The Ryu Controller is a software-defined networking (SDN) controller created to increase the agility of the network by making it easy to manage and adapt traffic flows. An SDN Controller is the brain of the SDN environment, communicating information down to the switches and routers via southbound APIs and up to the applications and business logic via northbound APIs. Ryu Controller provides software components, with well-defined application program interfaces (APIs), that make it easy to develop network management and control applications.

This component approach helps organizations customize deployments to meet their specific needs; developers can quickly and easily modify existing components or implement their own to ensure the underlying network can meet the changing demands of their applications.

### 1.2.4 Related work

In [1], authors described a dual-level security system for detecting DDoS attacks on SDN. Snort is used to detect signature-based attacks first. Additionally, they used machine learning algorithms to detect anomaly-based attacks. They have used two algorithms namely the Support Vector Machine (SVM) classifier and the Deep Neural Network (DNN) to create a trained model based on the KDD Cup dataset. The system was evaluated using the Mininet emulator and Ryu controller in the SDN environment. The results reveal that DNN outperforms SVM in the results. In [2], authors proposed a deep learning-based DDoS attack detection approach (called DeepDefense). Deep learning can be used to automatically extract high-level features from low-level ones and gain powerful representation and inference capabilities. To trace network attack activities, they designed a recurrent deep neural network to learn patterns from network traffic sequences. The experimental results demonstrate that their model performs better than conventional machine learning models. Comparing the larger dataset to conventional machine learning, they reduced the error rate from 7.517percent to 2.103percent. In [3], authors in this study analyzed the effects of Distributed Denial of Service attacks on a software-defined networking environment and suggested an entropy-based method for detecting these attacks. To mitigate the attacks, the study uses the OpenFlow protocol, and an OpenFlow controller (POX). First, the results of the detection algorithm were observed through simulation, then implemented into a small-scaled network testbed, and finally, the results of the proposed algorithm were presented and analyzed.

In[4], an approach based on K-means++ and Fast K-Nearest Neighbors (K-FKNN) for DDoS detection in SDN is presented, and the modular detection system is implemented in the controller. To evaluate the performance of the system, detailed experiments are conducted. Based on the results of the experiments, K-FKNN proves to be more effective at detecting DDoS than K-Nearest Neighbors (KNN) and additionally provides higher precision and stability in SDN detection. In[5] , authors presents a flexible modular architecture that identifies and mitigates LR-DDoS attacks in SDN settings. Specifically, they train the intrusion detection system (IDS) in their architecture using six machine learning (ML) models (i.e., J48, Random Tree, REP Tree, Random Forest, Multi-Layer Perceptron (MLP), and Support Vector Machines (SVM)) and evaluate their performance using the Canadian Institute of Cybersecurity (CIC) DoS dataset. Despite the difficulty in detecting LR-DoS attacks, their approach achieved a detection rate of 95 percent in the evaluation. In addition, their deployment uses an open network operating system (ONOS) controller running on a Mininet virtual machine to simulate real-world production networks as closely as possible. Their testing topology shows that the intrusion prevention detection system mitigates all attacks previously detected by the IDS system. These results demonstrate the usefulness of their architecture in identifying and

mitigating LR-DDoS attacks. In [6], paper aims to classify the traffic into normal and attack classes based on features given in dataset by using various deep learning techniques. The classification of the traffic is done after pre-processing of the dataset. This Model can classify TCP-SYN, UDP flooding and ICMP flooding. Stacked Auto-Encoder Multi-layer Perceptron is used to classify the normal and attack traffic. They have got 99.75 percent accuracy.

In [7], SDN-based DDoS detection and defense system is presented in this paper that uses deep learning on Software-Defined Networks (SDNs). From historical patterns in network traffic, the model can find network attack activities over time. With the defense system using the model, DDoS attacks can be effectively cleaned in Software-Defined Networks. Furthermore, it reduces the dependence on the environment, simplifies the update of the detection system in real-time, and makes it easier to upgrade or change the detection strategy. In[8] , authors proposes a method for detecting DDoS attacks based on information entropy and deep learning. As a first step, the controller can examine suspicious traffic by detecting information entropy. An advanced convolutional neural network (CNN) model then executes fine-grained packet-based detection to differentiate between normal and attack traffic. At last, the controller performs a defense strategy in order to intercept the attack. This method is capable of detecting DDoS attack traffic in an SDN environment with 98.98 percent accuracy, which means it is a highly effective method for detecting DDoS attacks on SDN networks. In[9] , authors proposed deep-learning based multi-vector DDoS detection system in a software-defined network (SDN) environment. With SDN, devices are programmed to meet specific objectives and there is no need for vendor-specific hardware. On top of an SDN controller, the system was implemented as a network application. Deep-learning was used to reduce the number of features derived from network traffic headers. Applied different performance metrics to traffic traces collected from different scenarios to evaluate their system. In their proposed system, they observed high accuracy with low false-positive rates for attack detection.

The work in [10], presented in this study uses machine learning classifiers on a distributed processing platform to detect DDoS attacks in real-time. Analyzed the DDoS detection mechanism in an OpenStack-based cloud testbed using the Apache Spark framework. Based on the benchmark dataset and real-time cloud data, they compared the performance of classification. Based on the results of the experiments, the random forest method is more accurate than other methods. In addition, they demonstrated the effectiveness of the proposed distributed approach in terms of training and detection time.



## 1.3 Problem Statement

To detect different Distributed Denial of Service(DDoS) attacks using deep learning based algorithms in Software defined networks (SDN) environment.

## 1.4 Applications

- Hyper-converged storage.
- Video applications.
- Orchestration of mobile network services.
- Data center network's scalability.

## 1.5 Objectives and Scope of the project

### 1.5.1 Objectives

- To capture the real time traffic(normal and attack) in SDN environment.
- To extract suitable features using feature extraction techniques.
- To detect anomaly based attacks using Deep learning techniques.

### 1.5.2 Scope of the project

- We have deployed a Deep-learning model in the mininet emulator, but our model can also be deployed in real time.
- Deep learning is used to separate attack traffic from normal traffic in the project.
- Our SDN project aims to detect and stop DDoS attacks.

# Chapter 2

## REQUIREMENT ANALYSIS

### 2.1 Functional Requirements

Functional requirements are:

1. System should be able to detect signature-based attack in SDN using the deployed module.
2. System should be able to keep track of normal system activities and should be able to classify it as normal and abnormal activities.

### 2.2 Non Functional Requirements

Non-Functional requirements are:

1. System should be able to classify between normal and malicious traffic in less than 5 seconds.
2. The system should add mitigative flow entry every 3 seconds after the detection of DDOS attack until the attack is happening.
3. The model should have an accuracy of classification above 9

### 2.3 Hardware Requirements

The hardware requirements are:

1. Computer/laptop with 4GB RAM
2. I3+ processor

## 2.4 Software Requirements

Software requirement:

1. Mininet 2.2.2
2. Wireshark
3. Jupyter notebook.
4. python
5. Ubuntu 16.04

# Chapter 3

## SYSTEM DESIGN

### 3.1 Proposed system

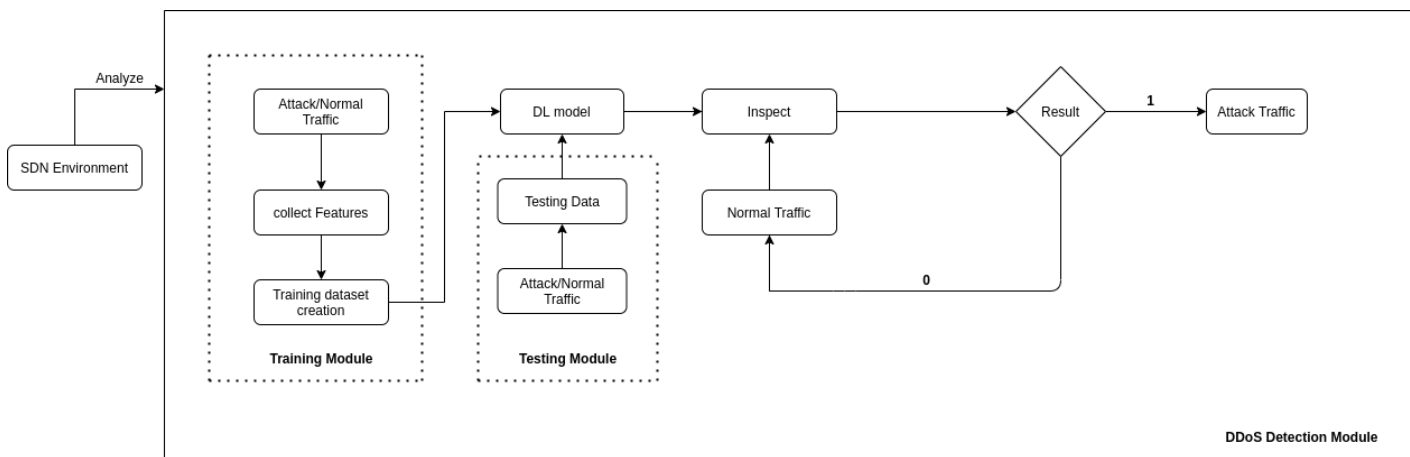


Figure 3.1: DDoS detection system

In the Figure 3.1, Attack and normal traffics are generated using hping3 tool and the script. The attributes are taken from flow table of switch 1 and is collected using a script and is stored in a text file. Then using those attributes, feature extraction is done. Using those characteristics training dataset is created. One more attribute is added which is named as target and the target attribute is filled with the value 1 or 0 depending on the type of tuple i.e 1 is added for the attack and 0 is added to the normal traffic. Then the model is trained using the extracted features. Then in testing module again the attack and normal traffic is generated using the script which is been written and with the help of a another script the attributes of the flow table are again collected and the feature is made every 3 seconds and feeded to the trained DL model. Now based on the features that are created the traffic is classified as normal or anomalous. If the traffic is found to be anomalous then the normal traffic is sent to the intended destination.

## 3.2 Advantages of Proposed system

The advantages of our proposed system are:

- (a) It is deployed on controller i.e. it is centralized.
- (b) It uses DL model which is mainly used to classify data into different classes.
- (c) After the detection of the DDoS attack, the controller mitigates the attack by adding mitigative flows and allows the normal traffic to intended destination host.
- (d) It is efficient.

### 3.3 Level zero DFD

A diagram illustrating the attack traffic and normal traffic using the Hping3 tool can be found in Figure 3.2 script which is real time data. Then the feature extraction is done. Then the data is generated. Deep learning classifier is used to classify the traffic as normal or anomalous. The data generated is used to train the Deep learning model. Then again for testing the same code is run to generate the traffic. The feature is extracted every 3 seconds and the data is also generated every 3 seconds. Then the data which is generated every 3 seconds is feeded the Deep learning model which is trained. The Deep learning model then detects the attack if it exists and classifies it as attack traffic or normal traffic. First we train the Deep learning model with the realtime data that we generated in sdn environment using virtual machine and mininet emulator. The topology was created using the miniedit GUI. Then the realtime traffic is sent to collect the data for training. The training dataset is realtime and is generated using the hping3 tool and bash script. The attack traffic and normal traffic is generated using hping3 tool and script which is real time data. Then the feature extraction is done. Then the data is generated. Then the real time data that is generated is used to train the Deep learning model. Then that Deep learning classifier is used to classify the traffic as normal or anomalous. The data generated is used to train the Deep learning model. Then again for testing the same code is run to generate the traffic. The feature is extracted every 3 seconds and the data is also generated every 3 seconds. Then the data which is generated every 3 seconds is feeded the Deep learning model which is trained. The Deep learning model then detects the attack if it exists and classifies it as attack traffic.

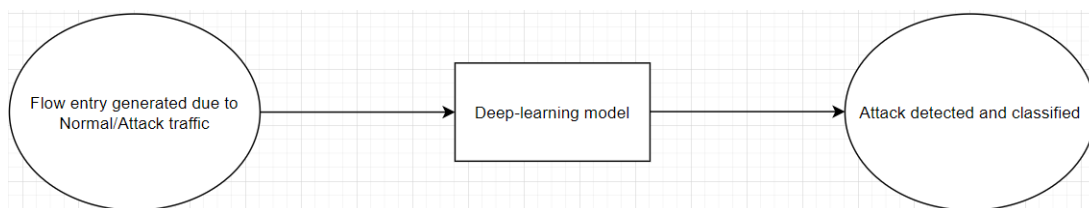


Figure 3.2: Level zero DFD

# Chapter 4

## IMPLEMENTATION

### 4.1 Proposed Deep-learning models

#### 4.1.1 Feed forward neural network

The feed forward process is an artificial neural network with a multilayered structure. It consists of an input layer and passes from one layer to another hidden layer. It has a function for calculating its output when receiving an output from the node in the previous layer. The function is called activation function. Each layer does not need to be the same function. The function converts incoming data to distinguish using a single line called linearly separable. Before the data has been sent to the output layer, it is sometimes necessary to use more than one hidden layer in order to convert the data into Linearly Separable until it reach the output layer.

#### 4.1.2 Bi-directional neural network

These networks have input, output and hidden components, and the recurrent units represent the hidden components hidden unit completes the most important work. The RNN model essentially has a one way flow of information from the input units to the hidden units, and the synthesis of the one-way information flow from the previous temporal concealment unit to the current timing hiding unit.

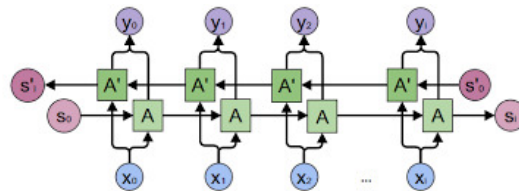


Figure 4.1: BRNN

## 4.2 Training Data Collection

### 4.2.1 How we have collected Data?

To collect the real time data for training the Deep learning model we first create the topology of 9 host with 6 openflow switches. The topology is created using miniedit gui and mininet emulator. To generate the attack traffic and normal traffic hping3 tool and bash script is written. The traffic is sent from two or more host to one of the target host. The flow table of the openflow switch which is on the way to the destination host is monitored. Then a script collects the data from the that particular openflow switch and strores values in the text file. Then that text file is converted into the csv file. Then the csv file is split into the further 4 csv files containing one attribute in each file. Then that 4 csv file is used to create the features and finally a data set is created to train the Deep learning model.

#### Steps for data-collection

- (a) Begin
- (b) Send the traffic from two or more hosts to one of the target host.
- (c) Monitors flows of S1.
- (d) Store the flow in text file.
- (e) Convert text file to csv file.
- (f) Put the data of important individual attributes of csv to individual csv file for each attribute.
- (g) use the now created individual csv files to calculate the features.
- (h) Append the values of all the features to dataset.csv file (Final csv).
- (i) Add another attribute value to the file called "target" and add 0 to normal traffic and 1 if the data collected is from the attack.
- (j) Read the processed csv file.
- (k) Feed the processed csv file into trained model.
- (l) End Begin





### 4.3 Attributes used

1. Speed of Source IP (SSIP): This is incoming number of the source IPs per unit of time. The way it modifies is presented in Figure 4.3.

$$SSIP = \frac{SumIP_{SRC}}{T}$$

Figure 4.3: SSIP formulae

Where T is sampling time and sumSRC is the source IPs. The T sampling monitors to flows that detects in three 3 seconds. This update new flows to controller.

2. Standard Deviation of Flow Packets (SDFP): This is incoming number of packet in the T period. Which is depicted in Figure 4.4.

$$SDFP = \sqrt{\frac{1}{N} \sum_{i=0}^N (packets_i - MeanPackets)^2}$$

Figure 4.4: SDFP formulae

This shows packets<sub>i</sub> is number of packets in the i<sup>th</sup> Meanpackets in the average number. It has high correlation of DDoS attack because the attacker can send large number of attack packets.

3. Standard Deviation of Flow Bytes (SDFB): This is incoming number of bytes in T period in the network. Represented in Figure 4.5.

$$SDFP = \sqrt{\frac{1}{N} \sum_{i=1}^N (bytes_i - MeanBytes)^2}$$

Figure 4.5: SDFB formulae

the number of bytes ith flow and MeanBytes is the average number. This is same like packets has high correlation to the DDoS attack that's expected value is normal traffic flows.

4. Speed of Flow Entries (SFE): Number of flow entries into the switch per unit of time. This is common parameter in DDoS detection represented in Figure 4.6.

$$SFE = \frac{N}{T}$$

Figure 4.6: SFE formulae

this increase number of flows significantly into a fixed intermission of time this is also normal traffic flows in case of SEF valued compared.

5. Ratio of Pair-Flow Entries (RPF): This is number of interactive flows divided by total number of flows entries. Number of flow entries in the switch that divided by total number of flows in T period. This is represented in Figure 4.7.

$$REIP = \frac{IntIP}{N}$$

Figure 4.7: REIP formulae

## 4.4 Feature analysis

- Figure 4.8 : There is steep increase during DDoS attack case for the speed of source IP address since we are making use of IP spoofing technique to simulate the DDoS attack. There is a new IP which is entering the network at every instance so the switch has to tackle with this issue and create a new flow entry for each and every IP address entering the pool and that results in increase number of flows per unit time and that is been demonstrated by this graph.

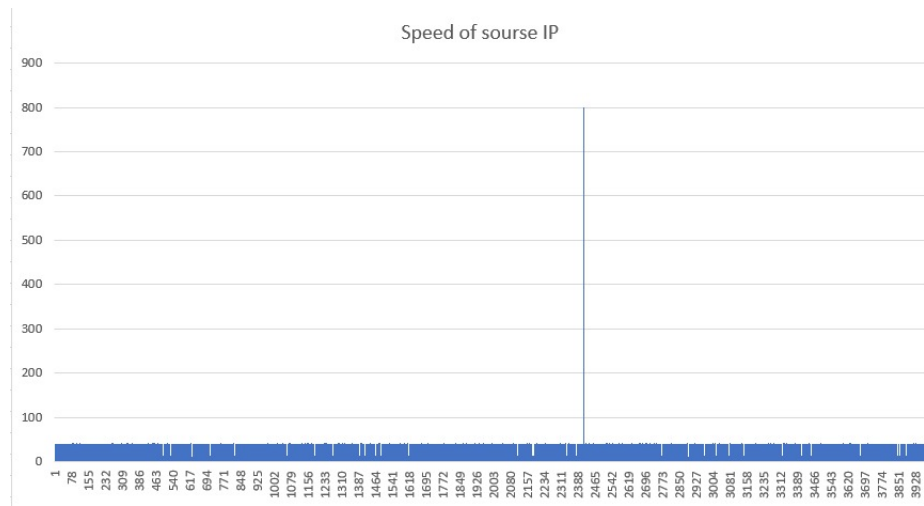


Figure 4.8: speed of source IP address

- Figure 4.9 and Figure 4.10: Also the standard deviation of flow packets and standard deviation of flow bytes decreases due to the increased number of packets resulting in increased number of flows, but there is very slight variation in the packet size as well as the byte size which results in lower deviation.

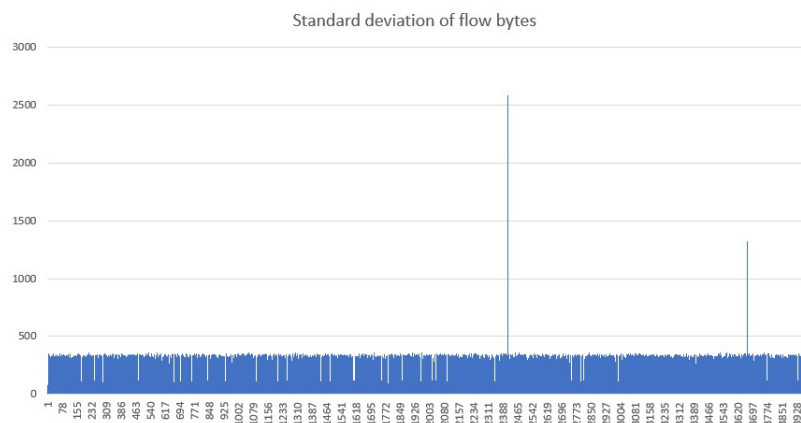


Figure 4.9: standard deviation of flow bytes

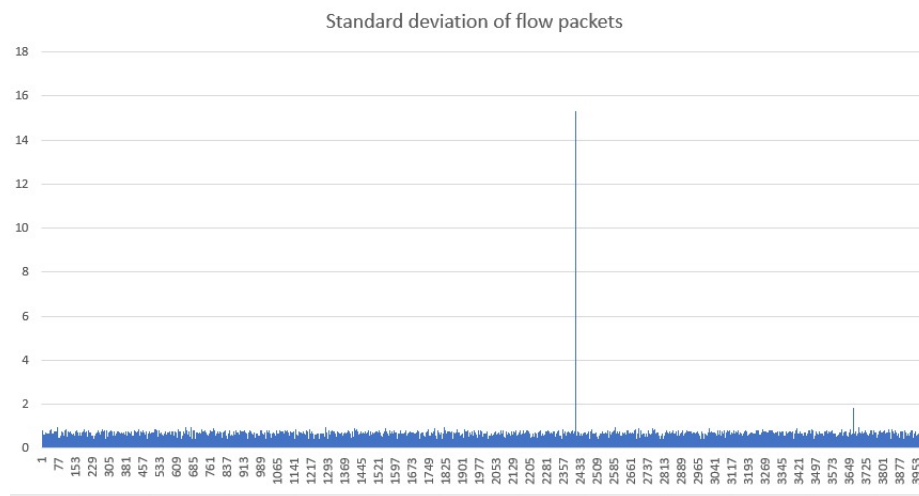


Figure 4.10: standard deviation of flow packets

- Figure 4.11 : Speed of flow entries has steep increase during DDoS attack scenario since more number of flows are created into switches to tackle with the large number of incoming IP's.

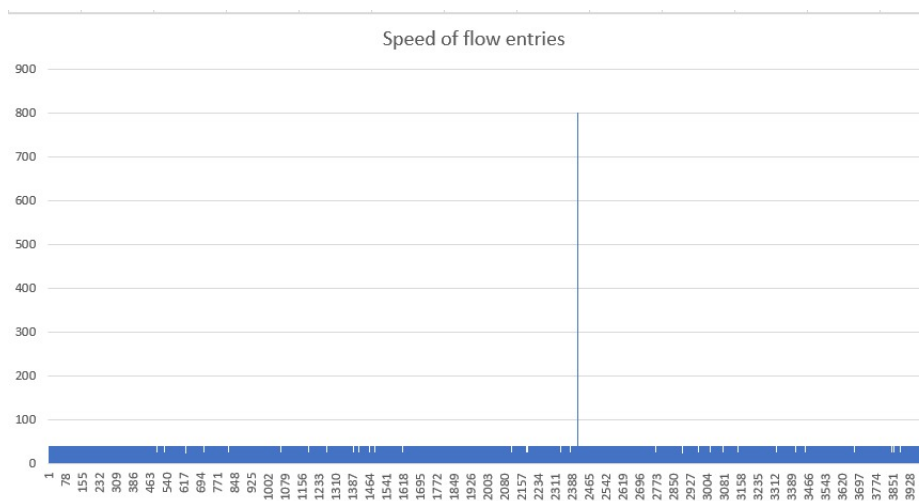


Figure 4.11: Speed of flow entries

- Figure 4.12 : Ratio of pair flow entries decreases during DDoS attack scenario since the response sent out by the target host machine is lost out in address space and as you can observe the ratio of pair flow entries is approximately equal to 1 during the normal traffic case where in bidirectional flow is being created for every incoming IP.

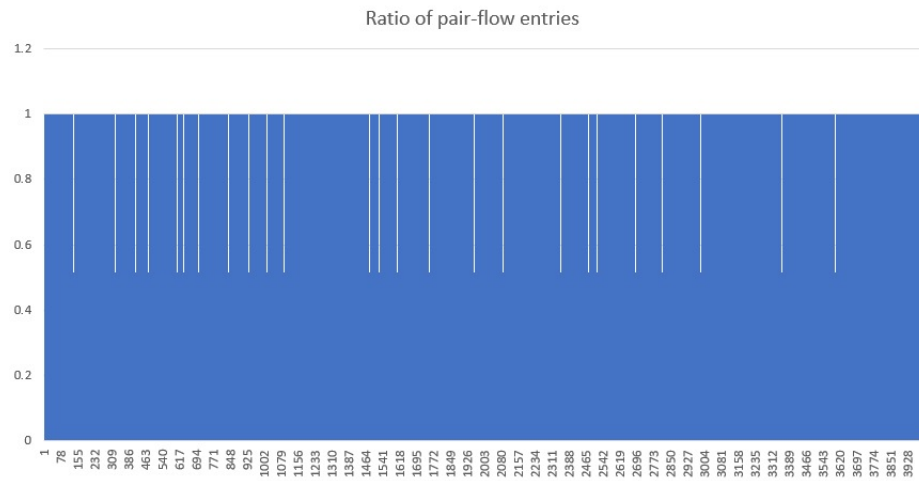


Figure 4.12: Ratio of pair flow entries



### 4.4.1 Dataset Description

SSIP	Stdevpack	Stdevbyte	NbFlow	NbIntFlow	Class
41	0.389775676567567	75.804607046879	41	0.516129032258065	1
41	0.450748019757488	119.721585624799	41	0.516129032258065	1
13	0.835164654424503	351.018887490403	26	1	0
13	0.714142842854285	307.680791365012	26	1	0
11	0.588235294117647	334.996291876394	22	1	0
12	0.644899264791328	329.904769040707	24	1	0
41	0.530054087908225	115.610779070348	41	0.516129032258065	1
41	0.125971768966236	53.1600865037517	41	0.516129032258065	1
12	0.532203905688334	299.208498812739	25	1	0
41	0.450748019757488	117.719312895174	41	0.516129032258065	1
13	0.627992834354024	338.696101800715	26	1	0
41	0.450748019757488	115.717501106941	41	0.516129032258065	1
41	0.450748019757488	120.722887675622	41	0.516129032258065	1
12	0.433012701892219	324.570937099853	24	1	0
41	0.450748019757488	114.216459338253	41	0.516129032258065	1
41	0.450748019757488	105.966126564994	41	0.516129032258065	1
12	0.735430979604322	328.037023491501	25	1	0
41	0.450748019757488	110.715152319697	41	0.516129032258065	1
41	0.279828251175766	78.2869996769471	41	0.516129032258065	1
41	0.450748019757488	102.718820512158	41	0.516129032258065	1
41	0.450748019757488	105.216594403037	41	0.516129032258065	1
13	0.556776436283002	342.189339949391	26	1	0
41	0.125971768966236	55.931465421009	41	0.516129032258065	1
12	0.566859453382579	327.97125490555	25	1	0
12	0.673002302199207	296.84386094002	24	1	0
41	0.450748019757488	112.715701410375	41	0.516129032258065	1
41	0.450748019757488	97.2276526570866	41	0.516129032258065	1
41	0.279828251175766	72.3905413618938	41	0.516129032258065	1
13	0.661437827766148	350.966519058442	26	1	0
12	0.574745517574778	331.283443929223	25	1	0
41	0.279828251175766	74.6483065247533	41	0.516129032258065	1
41	0.279828251175766	79.5420294139409	41	0.516129032258065	1
41	0.450748019757488	113.215921818328	41	0.516129032258065	1
12	0.674013077624511	330.732172736646	25	1	0
12	0.816496580927726	302.245611584721	24	1	0
41	0.450748019757488	112.215513811872	41	0.516129032258065	1
13	0.476969600708473	333.103722585023	26	1	0
12	0.653140718210045	322.380433718179	25	1	0
41	0.279828251175766	77.6595417349123	41	0.516129032258065	1
12	0.853460638652068	323.383372416634	24	1	0
12	0.590788008437991	321.732373598938	25	1	0
12	0.715866868880813	338.427258067033	25	1	0
41	0.450748019757488	110.715152319697	41	0.516129032258065	1
13	0.446514277487294	347.757228537381	26	1	0

Figure 4.13: Description of dataset

In the Figure 4.13: There are a total of 4000 rows and 6 columns in our data set. Here one more feature is added that is attack type. After collecting the normal traffic 0 is added to it and for attack type 1 is added. So this attack type feature is used to classify the traffic as normal or anomalous.

# Chapter 5

## RESULTS AND DISCUSSIONS

### 5.1 Topology

- Figure 5.1 : After the execution of `sudo python ./miniedit.py` command miniedit GUI is opened.
- Miniedit is a tool where you can create a new topology using virtual switches, router, links, and controller. You can also open already existing topology by clicking open file.

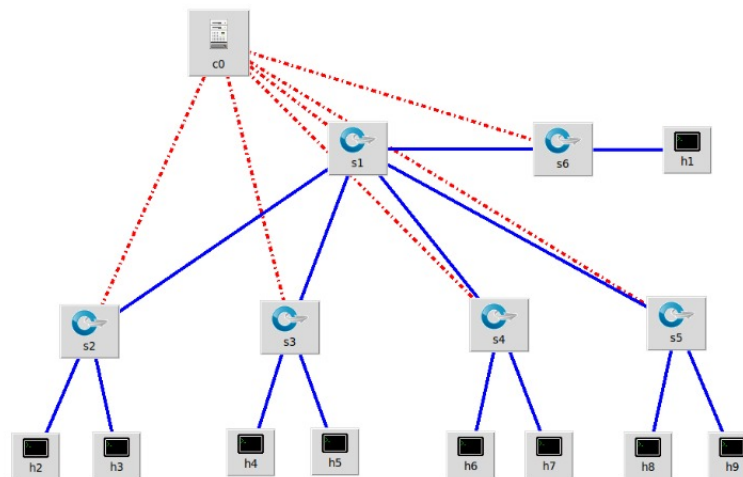


Figure 5.1: Topology



## 5.2 Splitting of training and testing samples

Table 5.1: Splitting of Training and testing samples

Dataset	x train	y train	x test	y test
Benchmark	(73041, 12)	(73041, 1)	(31304, 12)	(31304, 1)
Real-time	(2800, 5)	(2800, 1)	(1200, 5)	(1200, 1)

As shown in Table 5.1, the dataset is splitted into 70 percent training data and 30 percent testing data. There are 2800 rows and 5 columns in training data, 1200 and 1 column in testing data in real time dataset (sdn). There are 73041 rows and 12 columns in training data, 31304 rows and 1 column in testing data in benchmark dataset.

## 5.3 Feed Forward Neural Network

### Accuracy and Loss of FFNN model

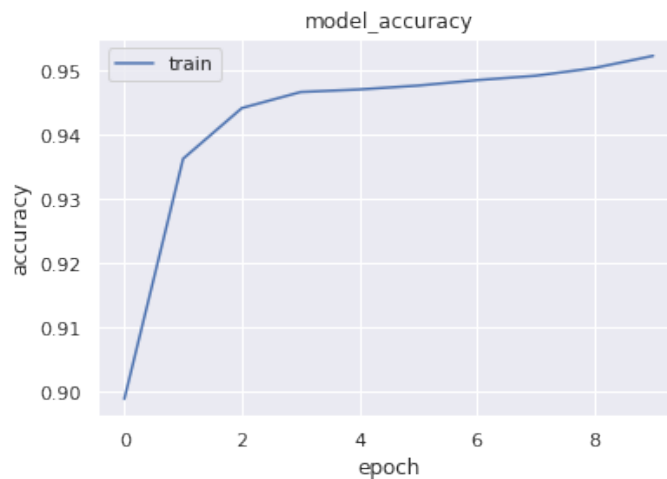


Figure 5.2: FFNN model Accuracy

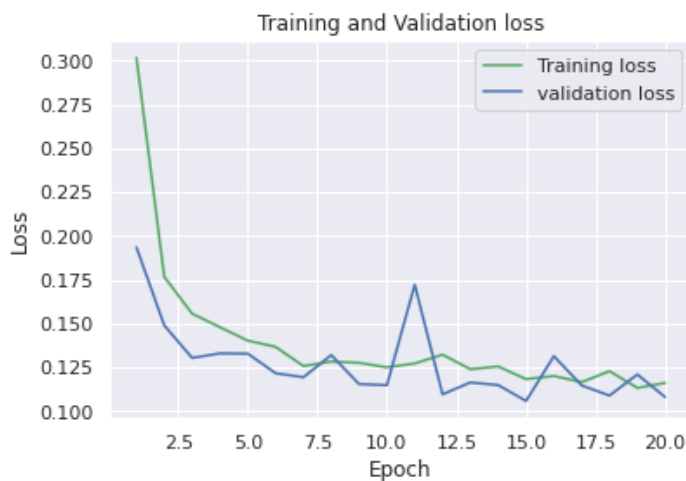


Figure 5.3: FFNN model Loss

In Figure 5.2 and Figure 5.3, We plotted the FFNN model accuracy and error-rate graph over real-time dataset. FFNN achieves a model accuracy of 96.32 percent and error rate of 3.68 percent for an epoch size of 100.

## 5.4 Bi-directional Neural Network

### Accuracy and Loss of BRNN model

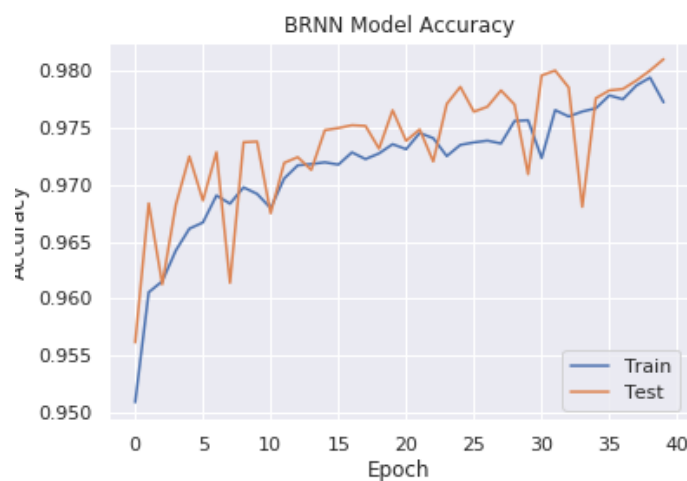


Figure 5.4: BRNN model Accuracy



Figure 5.5: BRNN model loss

In Figure 5.4 and Figure 5.5, We plotted the BRNN model accuracy and error-rate graph over real-time dataset. BRNN achieves a model accuracy of 99.21 percent and error rate of 0.79 percent for an epoch size of 100.

## 5.5 Comparison of accuracy of Deep-learning models

Table 5.2: Comparison of accuracy and Error rate

Deep-learning model	Benchmark Accuracy	Real-time Accuracy	Benchmark Error rate	Real-time Error-rate
BRNN	97.17	99.21	2.83	0.79
FFNN	95.43	96.32	4.57	3.68

## 5.6 Performance comparison of deep-learning models

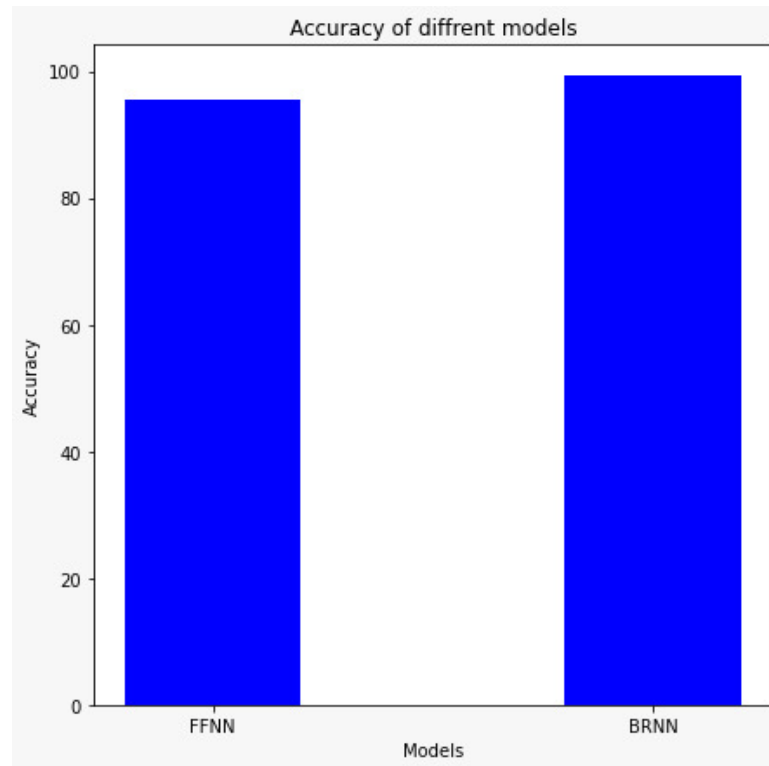


Figure 5.6: Accuracy graph

As shown in Figure 5.6 , The real-time data set is used to train the model and predict traffic types using the attributes. As the training data is fed into the model, it learns it, and based on this learning, it predicts the target variable which is traffic type. We have used 2 models, BRNN and FFNN. Among these models, BRNN performs better than the other model.

## 5.7 Confusion matrix of BRNN model

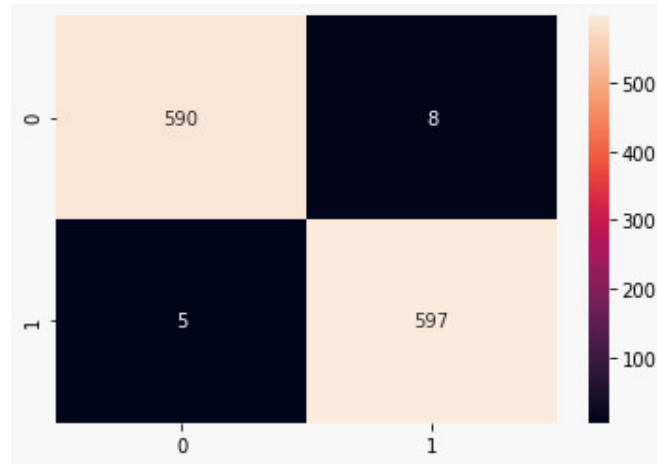


Figure 5.7: Confusion matrix for BRNN

The heatmap in the Figure 5.7 represents the confusion matrix for the BRNN model. Here, '0' represents the normal traffic and '1' represents attack traffic. There are 1200 test data tuples and among them, 590 are True Positive (TP), 8 are False Positive (FP), 5 are False Negative (FN) and 597 are True Negative (TN). This shows that the model has classified nearly every class label correctly and only 13 among 1200 have been classified inaccurately by the model.

## 5.8 Normal Traffic

Using the command `sh monitor.sh` to detect whether traffic is normal or anomalous, a script named `monitor.sh` identifies whether traffic is normal in Figure 5.8 below.

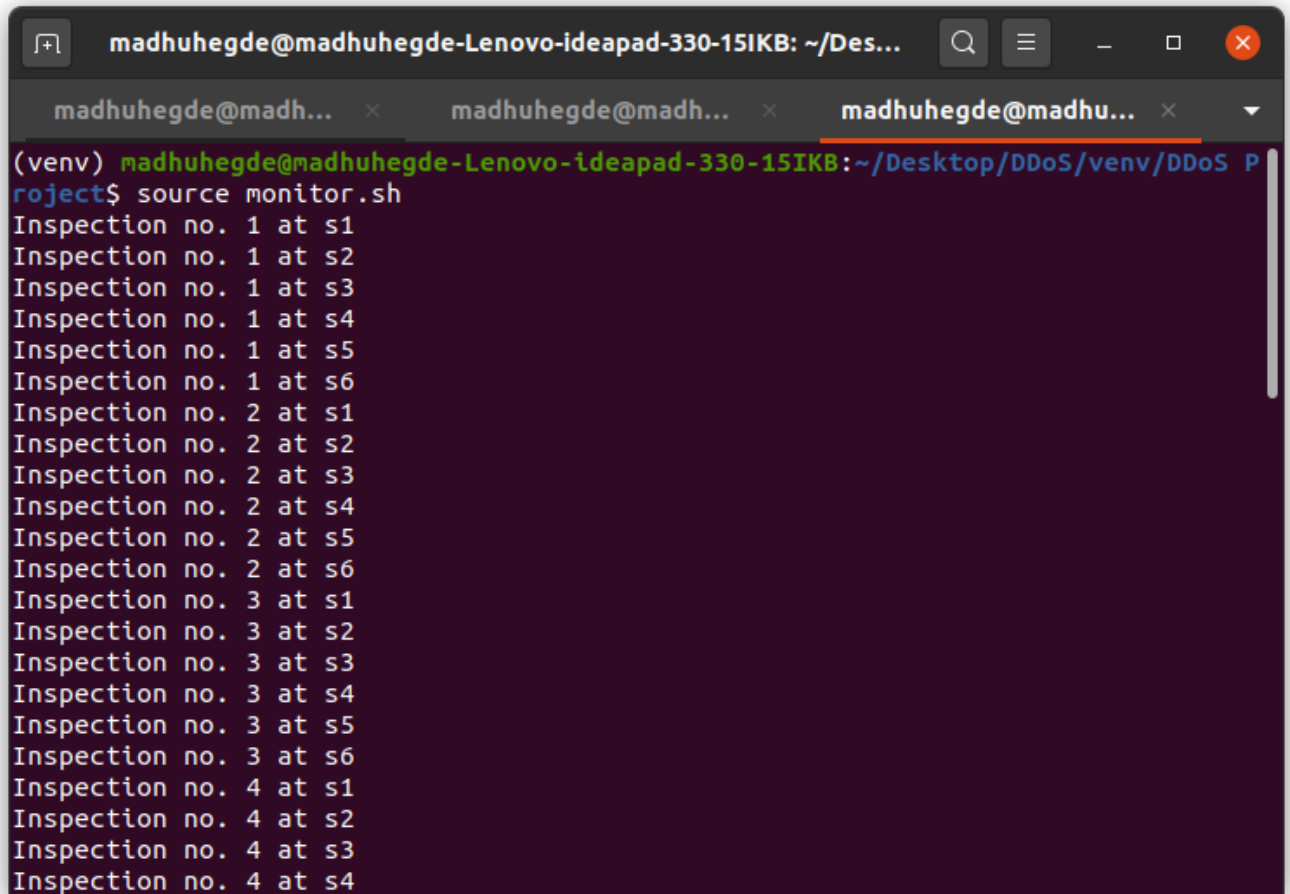
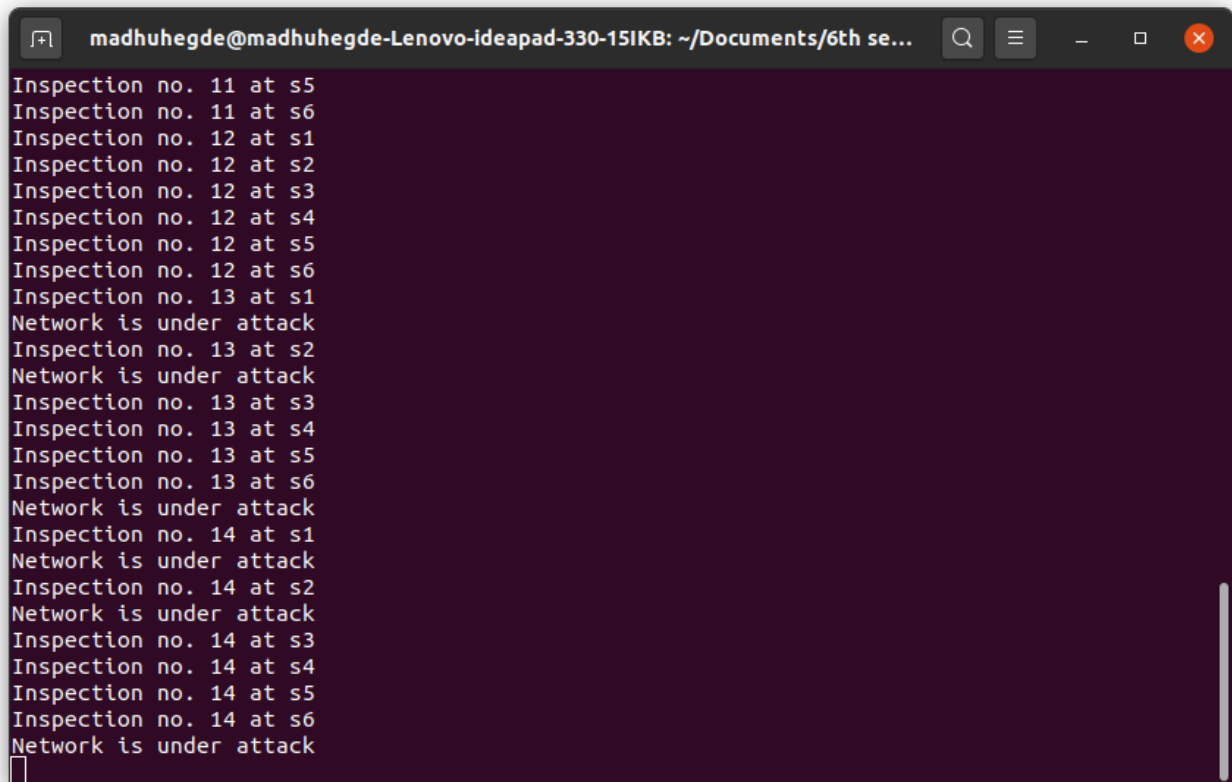
A terminal window with a dark purple background. The title bar shows the user 'madhuhegde' on a 'Lenovo-ideapad-330-15IKB' machine, with the current directory as '~/Desktop/DDoS/venv/DDoS'. The terminal shows the command '(venv) madhuhegde@madhuhegde-Lenovo-ideapad-330-15IKB:~/Desktop/DDoS/venv/DDoS P roject\$ source monitor.sh' being executed. The output consists of 20 lines of text, grouped into four sets of five lines each. Each set starts with 'Inspection no. X at sY', where X is 1, 2, 3, or 4, and Y is 1, 2, 3, 4, or 5. The output is: 'Inspection no. 1 at s1', 'Inspection no. 1 at s2', 'Inspection no. 1 at s3', 'Inspection no. 1 at s4', 'Inspection no. 1 at s5', 'Inspection no. 1 at s6', 'Inspection no. 2 at s1', 'Inspection no. 2 at s2', 'Inspection no. 2 at s3', 'Inspection no. 2 at s4', 'Inspection no. 2 at s5', 'Inspection no. 2 at s6', 'Inspection no. 3 at s1', 'Inspection no. 3 at s2', 'Inspection no. 3 at s3', 'Inspection no. 3 at s4', 'Inspection no. 3 at s5', 'Inspection no. 3 at s6', 'Inspection no. 4 at s1', 'Inspection no. 4 at s2', 'Inspection no. 4 at s3', 'Inspection no. 4 at s4'.

Figure 5.8: Normal traffic.

## 5.9 Attack Traffic

After the execution of the script `monitor.sh` using the command `sh monitor.sh`, the attack traffic is detected. The result of detection is shown in Figure 5.9 below.



```
madhuhegde@madhuhegde-Lenovo-ideapad-330-15IKB: ~/Documents/6th se...
Inspection no. 11 at s5
Inspection no. 11 at s6
Inspection no. 12 at s1
Inspection no. 12 at s2
Inspection no. 12 at s3
Inspection no. 12 at s4
Inspection no. 12 at s5
Inspection no. 12 at s6
Inspection no. 13 at s1
Network is under attack
Inspection no. 13 at s2
Network is under attack
Inspection no. 13 at s3
Inspection no. 13 at s4
Inspection no. 13 at s5
Inspection no. 13 at s6
Network is under attack
Inspection no. 14 at s1
Network is under attack
Inspection no. 14 at s2
Network is under attack
Inspection no. 14 at s3
Inspection no. 14 at s4
Inspection no. 14 at s5
Inspection no. 14 at s6
Network is under attack
```

Figure 5.9: Detection of incoming traffic when there is attack.

## Chapter 6

# CONCLUSION AND FUTURE SCOPE

### 6.1 Conclusion

SDN is characterized by several key concepts, including dynamic programmability in forwarding, decoupling of the control and data planes, and logical centralization of the "brain" of the network. Data plane elements have become dumb, highly efficient, and programmable packet forwarding devices, while the control plane elements are now represented by a single entity, the controller. A DDoS attack attempts to exhaust the resources of the network to render its services unavailable. The effect of this DDoS attack is even worse in an SDN than in a traditional network. Therefore, if the attacker blasts too many packets into a network within a short period of time, the controller will be damaged, followed by the network's collapse. As such, we used a BRNN model to detect both normal and anomalous traffic this time. When the attack on a network has been detected, normal traffic can be directed to the intended destination unhindered. No matter how large or small the topology is, the above model will work well.

### 6.2 Future scope

To be able to use the model above in our topology. Hence, the future scope includes detecting and mitigating attacks anywhere within the network, i.e. globally by using Deep-learning models. To centralize the model as well.



Project title: DDoS attack detection in SDN environment.

Team no.: N11

Project domain: Network security.

#### ORIGINALITY REPORT

---

**22%**  
SIMILARITY INDEX

**15%**  
INTERNET SOURCES

**16%**  
PUBLICATIONS

**13%**  
STUDENT PAPERS

---

#### PRIMARY SOURCES

1	Submitted to B.V. B College of Engineering and Technology, Hubli Student Paper	8%
2	<a href="http://www.il-pib.pl">www.il-pib.pl</a> Internet Source	3%
3	Submitted to Indus International School Student Paper	2%
4	<a href="http://www.slideshare.net">www.slideshare.net</a> Internet Source	2%
5	<a href="http://www.geeksforgeeks.org">www.geeksforgeeks.org</a> Internet Source	2%
6	<a href="http://doaj.org">doaj.org</a> Internet Source	2%
7	Tanaphon Roempluk, Olarik Surinta. "A Machine Learning Approach for Detecting Distributed Denial of Service Attacks", 2019 Joint International Conference on Digital Arts,	2%

# Computer and Telecommunications Engineering (ECTI DAMT-NCON), 2019

Publication

8	<a href="http://networkenhancers.blogspot.com">networkenhancers.blogspot.com</a> Internet Source	1 %
9	<a href="http://ieeexplore.ieee.org">ieeexplore.ieee.org</a> Internet Source	1 %
10	"Machine Learning for Cyber Security", Springer Science and Business Media LLC, 2020 Publication	1 %
11	<a href="http://97g5x.88815812.cn">97g5x.88815812.cn</a> Internet Source	1 %
12	Submitted to University of Lancaster Student Paper	1 %
13	<a href="http://www.cloudflare.com">www.cloudflare.com</a> Internet Source	1 %
14	"Detection of DDoS in SDN Environment Using Entropy-based Detection", 2019 IEEE International Symposium on Technologies for Homeland Security (HST), 2019 Publication	1 %
15	Submitted to University of Hertfordshire Student Paper	1 %
16	<a href="http://www.infona.pl">www.infona.pl</a> Internet Source	1 %

17

Submitted to Champlain College

Student Paper

1%

18

ajast.net

Internet Source

1%

20

Rajib Biswas, Sambuddha Roy. "Botnet traffic identification using neural networks",

Multimedia Tools and Applications, 2021

Publication

1%

1%

21

Nisha Ahuja, Gaurav Singal, Debajyoti

Mukhopadhyay. "DLSDN: Deep Learning for

DDOS attack detection in Software Defined Networking", 2021 11th International

Conference on Cloud Computing, Data

Science &amp; Engineering (Confluence), 2021

Publication

1%

22

Submitted to Clemson University

Student Paper

&lt;1%

23

AfafD. Althobiti, RababM. Almohayawi, OmainahO. Bamsag. "Machine Learning

approach to Secure Software Defined

Network", The 4th International Conference

on Future Networks and Distributed Systems (ICFNDS), 2020

Publication

&lt;1%

# REFERENCES

- [1] P. S. Hiremath Karan B. V, Narayan D. G. "Detection of DDoS Attacks in Software Defined Networks". pages 6(6):265–270, 2018.
- [2] Xiaolin Li Large-scale Intelligent Systems Laboratory University of Florida  
†Zhejiang Gongshang University Xiaoyong Yuan, Chuanhuang Li†. "ddos Deep-Defense: Identifying DDoS Attack via Deep Learning". pages 8(8):1–8, 2018.
- [3] Brian Urbina Department of Electrical Tamer Omar, Anthony Ho and Pomona Computer Engineering, California State Polytechnic University. "Detection of DDoS in SDN environment using Entropy-based detection". pages 7(6):1–6, 2019.
- [4] Feng Xiang Zhixin Sun Yuhua Xu, Houtao Sun. "Efficient DDoS Detection Based on K-FKNN in Software Defined Networks". pages 7(6):160536 – 160545, 01 November 2019.
- [5] Jian Zhu Luting Feng Ling Song Jin Ye, Xiangyang Cheng. "A DDoS Attack Detection Method Based on SVM in Software Defined Network". pages 8:1–8, 24 April 2018.
- [6] Debajyoti Mukhopadhyay Nisha Ahuja, Gaurav Singal. "DLSDN: Deep Learning for DDOS attack detectionin Software Defined Networking". pages 5(13):683–688, 2021.
- [7] Xiaoyong Yuan Zhengjun Sun Weiming Wang Zhengjun Sun Weiming Wang Xiaolin Li Liang Gong Chuanhuang Li, Yan Wu. "Detection and defense of DDoS attack-based on deep learning in OpenFlow-based SDN". (15):1–15, January 2018.
- [8] Ying Liu Lu Wang. "A DDoS Attack Detection Method Based on Information Entropy and Deep Learning in SDN". (13):155859–155872, 13 May 2020.
- [9] Ahmad Y Javaid Quamar Niyaz, Weiqing Sun. "A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN)". pages 18(1-18):155859–155872, 22 November 2016.
- [10] Sumedha Shinde Shweta Gumaste, D. G. Narayan. "Detection of DDoS Attacks in OpenStack-based Private Cloud Using Apache Spark)". pages 4(62-71):9, December 2020.