



**NAMIBIA UNIVERSITY**  
OF SCIENCE AND TECHNOLOGY

**Faculty of Computing and Informatics**

**Department of Computer Science**

**EFFICIENT HYBRID DEEP REINFORCEMENT LEARNING  
MECHANISM FOR DISTRIBUTED DENIAL OF SERVICE ATTACK  
DETECTION IN SOFTWARE DEFINED NETWORKS**

Thesis submitted in fulfilment of the requirements for the degree of

**Master of Computer Science**

at the

Namibia University of Science and Technology

**Presented by:**

**Gabriel Precious Musa**

**Student Number:**

**219147523**

**Supervisor:**

**Professor Fungai Bhunu Shava**

**Submission Date:**

**May 2022**

## META DATA

TITLE: Mr.

STUDENT NAME: Gabriel Precious Musa

SUPERVISOR: Professor Fungai Bhunu Shava

CO-SUPERVISOR: N/A

DEPARTMENT: Computer Science

QUALIFICATION: Master of Computer Science

SPECIALISATION: Computer Security

STUDY TITLE: Efficient Hybrid Deep Reinforcement Learning Mechanism for DDoS Attacks

Detection in Software Defined Networks

MAIN KNOWLEDGE AREA: Information Security, Cyber Security and Networking, Machine Learning

KEYWORDS: Software Defined Networks, Distributed Denial of Service, Machine Learning, Intrusion Detection Systems, Deep Reinforcement Learning, CICIDS2017, Network Security.

TYPE OF RESEARCH: Experimental Research

METHODOLOGY: Mixed Method

STATUS: Thesis

SITE: Namibia University of Science and Technology

DOCUMENT DATE: 12 February 2022

SPONSOR: German Academic Exchange Service or Deutscher Akademischer Austauschdienst (DAAD)

CLUSTER: Digital Forensics and Information Security

## DECLARATION

I, **Gabriel Musa** hereby declare that the work contained in the thesis, entitled Efficient Hybrid Deep Reinforcement Learning Mechanism for DDoS attacks detection in Software Defined Networks, is my own original work and that I have not previously in its entirety or in part submitted it at any university or other higher education institution for the award of a degree.

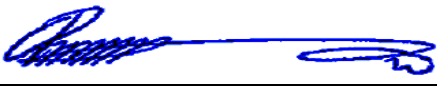
Signature: \_\_\_\_\_

Date: 10/05/2022

## RETENTION AND USE OF THESIS

I, **Gabriel Musa** being a candidate for the degree of Master of **Computer Science** accept the requirements of the Namibia University of Science and Technology relating to the retention and use of the thesis deposited in the Library and Information Services.

In terms of these conditions, I agree that the original copy of my thesis deposited in the Library and Information Services will be accessible for purposes of study and research, in accordance with the normal conditions established by the Librarian for the care, loan or reproduction of the thesis.

Signature: \_\_\_\_\_

Date: **10/05/2022**

## ACKNOWLEDGEMENT

Firstly, I thank God for His grace and divine orchestration of my academic path this far. If it was not for God this would not have been possible.

I am immensely grateful to my supervisor Professor Fungai Bhunu Shava for the support, belief, and guidance you provided which is out of this world. You ensured I stayed on track and saw me to the end.

To my family, thank you for the support you gave me throughout. Your positive encouragement and sacrifice are priceless.

The support from DAAD has enabled me to have a memorable academic experience.

To my research mates, the DAAD family, the cyber security cluster and the staff at the Faculty of Computing and Informatics, I say thank you for the lighter moments and the support and encouragement during the tough moments of the Covid-19 pandemic.

## Abstract:

The Internet architecture remains fixed since its invention but the Software Defined Network (SDN) comes with more flexibility, innovation, and programmability aspects being a very promising network architecture. However, the centralized control architecture in SDN represents a single point of failure. This vulnerability is prone to Distributed Denial of Service attack (DDoS) which remains a common and sophisticated attack on computer networks. With the controller faced with DDoS attacks while already overloaded with decision making, it raises a major security concern for SDN and therefore necessitates an efficient DDoS attack detection mechanism. This study aimed at designing a mechanism that accurately detects DDoS attacks while using minimum computational resources. We introduced a Hybrid Deep Reinforcement Learning Mechanism (HDRLM) for the SDN at the controller. An evaluation of literature was conducted to identify DRL algorithms that are accurate at the same time efficient. Double Deep Q-Network and Deep Q Network (DQN) were identified, and Deep Q-Network (DQN) was adopted in the study. To confirm the performance, simulated experimentation was used. Using the Design Science approach, a hybrid mechanism using the Deep Q Network algorithm that combines two different Deep Learning Neural networks for value approximation was designed. The HDRLM was demonstrated through experimentation in which the CICIDS2017 dataset was used to train and evaluate its performance. Detection accuracy of 98.16% was obtained and an 8% on CPU usage during detection, an improvement of the resource usage of the state-of-the-art detection mechanism. A positive upward trajectory of the accumulated rewards demonstrated that the mechanism was able to learn the environment by itself. Despite not achieving the highest accuracy, the HDRLM achieved a reasonably higher detection rate without consuming more computational resources compared to available mechanisms. This study provides a mechanism and an approach to designing mechanisms that reduce the cost of detection.

**Keywords:** *Software Defined Networks, Distributed Denial of Service, Machine Learning, Intrusion Detection Systems, Deep Reinforcement Learning, CICIDS2017, Network Security.*

## Table of Contents

|                                                          |             |
|----------------------------------------------------------|-------------|
| <b>META DATA.....</b>                                    | <b>I</b>    |
| <b>DECLARATION .....</b>                                 | <b>II</b>   |
| <b>RETENTION AND USE OF THESIS.....</b>                  | <b>III</b>  |
| <b>ACKNOWLEDGEMENT .....</b>                             | <b>IV</b>   |
| <b>ABSTRACT: .....</b>                                   | <b>V</b>    |
| <b>TABLE OF CONTENTS .....</b>                           | <b>VI</b>   |
| <b>LIST OF TABLES.....</b>                               | <b>VIII</b> |
| <b>LIST OF FIGURES.....</b>                              | <b>VIII</b> |
| <b>CHAPTER 1 : OVERVIEW OF THE STUDY .....</b>           | <b>1</b>    |
| 1.1 INTRODUCTION.....                                    | 1           |
| 1.2 BACKGROUND TO THE STUDY .....                        | 2           |
| 1.3 STATEMENT OF THE PROBLEM .....                       | 5           |
| 1.4 OBJECTIVES OF THE STUDY.....                         | 6           |
| 1.4.1 <i>Research sub-Objectives</i> .....               | 6           |
| 1.4.2 <i>Research Questions</i> .....                    | 6           |
| 1.5 SIGNIFICANCE OF THE STUDY .....                      | 7           |
| 1.6 THESIS STRUCTURE.....                                | 8           |
| <b>CHAPTER 2 : LITERATURE REVIEW .....</b>               | <b>9</b>    |
| 2.1 INTRODUCTION.....                                    | 9           |
| 2.2 SOFTWARE DEFINED NETWORK .....                       | 10          |
| 2.3 DISTRIBUTED DENIAL OF SERVICE ATTACK .....           | 13          |
| 2.4 INTRUSION DETECTION SYSTEM .....                     | 15          |
| 2.5 MACHINE LEARNING IN INTRUSION DETECTION SYSTEMS..... | 19          |
| 2.6 DEEP REINFORCEMENT LEARNING .....                    | 21          |
| 2.7 DDoS ATTACK BENCHMARK DATASET .....                  | 23          |
| 2.8 EVALUATION DATASETS ANALYSIS .....                   | 25          |
| 2.8.1 <i>KDD CUP 99</i> .....                            | 26          |
| 2.8.2 <i>NSL KDD</i> .....                               | 27          |
| 2.8.3 <i>DARPA DDoS 2009</i> .....                       | 28          |
| 2.8.4 <i>CAIDA DDoS</i> .....                            | 29          |
| 2.8.5 <i>CICIDS2017</i> .....                            | 30          |
| 2.9 ANALYSIS OUTCOME.....                                | 32          |
| 2.10 CICIDS2017 DDoS DATASET .....                       | 33          |
| 2.11 SUMMARY .....                                       | 35          |
| <b>CHAPTER 3 : METHODOLOGY .....</b>                     | <b>36</b>   |
| 3.1 INTRODUCTION.....                                    | 36          |
| 3.2 RESEARCH DESIGN PHASE.....                           | 37          |
| 3.2.1 <i>Literature Survey</i> .....                     | 40          |
| 3.2.2 <i>Experimentation</i> .....                       | 40          |
| 3.2.3 <i>Dataset Identification</i> .....                | 42          |
| 3.2.4 <i>The CICIDS2017 Dataset Features</i> .....       | 43          |
| 3.2.5 <i>Instance Distribution</i> .....                 | 45          |
| 3.2.6 <i>Dataset Pre-processing</i> .....                | 46          |
| 3.2.7 <i>Dataset Cleaning</i> .....                      | 47          |
| 3.2.8 <i>Feature Selection</i> .....                     | 48          |

|                                                             |                                                                  |            |
|-------------------------------------------------------------|------------------------------------------------------------------|------------|
| 3.2.9                                                       | <i>Mechanism Development</i> .....                               | 53         |
| 3.2.10                                                      | <i>Mechanism Training and Testing</i> .....                      | 54         |
| 3.2.11                                                      | <i>Mechanism Evaluation</i> .....                                | 54         |
| 3.3                                                         | MECHANISM DESIGN AND IMPLEMENTATION PHASE.....                   | 54         |
| 3.3.1                                                       | <i>Tools and Instruments</i> .....                               | 55         |
| 3.3.2                                                       | <i>Hybrid Deep Reinforcement Learning Mechanism Design</i> ..... | 56         |
| 3.3.3                                                       | <i>Reinforcement Learning Algorithm</i> .....                    | 56         |
| 3.3.4                                                       | <i>Deep Learning Algorithms</i> .....                            | 58         |
| 3.3.5                                                       | <i>Q-Learning Algorithm</i> .....                                | 59         |
| 3.3.6                                                       | <i>Convolution Neural Network</i> .....                          | 62         |
| 3.3.7                                                       | <i>Recurrent Neural Network (LSTM)</i> .....                     | 62         |
| 3.3.8                                                       | <i>Deep Neural Network (DNN)</i> .....                           | 62         |
| 3.3.9                                                       | <i>Implementation</i> .....                                      | 63         |
| 3.4                                                         | EVALUATION AND ANALYSIS PHASE .....                              | 64         |
| 3.5                                                         | SUMMARY .....                                                    | 67         |
| <b>CHAPTER 4 : EXPERIMENT AND ANALYSIS OF RESULTS .....</b> |                                                                  | <b>70</b>  |
| 4.1                                                         | INTRODUCTION.....                                                | 70         |
| 4.2                                                         | EXPERIMENTATION SETUP.....                                       | 71         |
| 4.2.1                                                       | <i>Hardware</i> .....                                            | 71         |
| 4.2.2                                                       | <i>Environment Setup</i> .....                                   | 71         |
| 4.2.3                                                       | <i>Reward Strategy</i> .....                                     | 72         |
| 4.2.4                                                       | <i>The Detection Algorithm</i> .....                             | 74         |
| 4.3                                                         | EXPERIMENTATION PROCEDURE.....                                   | 76         |
| 4.3.5                                                       | <i>Results Recording</i> .....                                   | 77         |
| 4.4                                                         | EVALUATION PARAMETERS .....                                      | 78         |
| 4.5                                                         | RESULTS.....                                                     | 79         |
| 4.6                                                         | FEATURE TECHNIQUE AND FEATURE SUBSET EVALUATION.....             | 79         |
| 4.6.6                                                       | <i>Accuracy</i> .....                                            | 79         |
| 4.6.7                                                       | <i>False Positive Rate</i> .....                                 | 82         |
| 4.6.8                                                       | <i>Summarised Results for Feature Subsets Evaluation</i> .....   | 92         |
| 4.7                                                         | HYBRID DEEP REINFORCEMENT LEARNING MECHANISM EVALUATION .....    | 93         |
| 4.7.9                                                       | <i>Accuracy</i> .....                                            | 94         |
| 4.7.10                                                      | <i>False Positive Rate</i> .....                                 | 95         |
| 4.7.11                                                      | <i>Average Reward</i> .....                                      | 101        |
| 4.7.12                                                      | <i>CPU usage</i> .....                                           | 102        |
| 4.7.13                                                      | <i>Memory Usage</i> .....                                        | 105        |
| 4.8                                                         | ANALYSIS BASED ON PARAMETERS .....                               | 106        |
| 4.9                                                         | VALIDATION.....                                                  | 107        |
| 4.10                                                        | SUMMARY .....                                                    | 108        |
| <b>CHAPTER 5 : CONCLUSION.....</b>                          |                                                                  | <b>109</b> |
| 5.1                                                         | INTRODUCTION.....                                                | 109        |
| 5.2                                                         | ACHIEVEMENTS OF STUDY OBJECTIVES .....                           | 111        |
| 5.3                                                         | CHALLENGES AND LIMITATION .....                                  | 114        |
| 5.4                                                         | FUTURE WORK.....                                                 | 114        |
| 5.5                                                         | CONCLUSION.....                                                  | 115        |
| <b>REFERENCES.....</b>                                      |                                                                  | <b>116</b> |
| APPENDIX A: EDITOR'S REPORT .....                           |                                                                  | 123        |

## List of Tables

|                                                                                           |     |
|-------------------------------------------------------------------------------------------|-----|
| Table 2.1: List of attack types in the KDD Dataset.....                                   | 27  |
| Table 2.2: CICIDS2017 dataset files description .....                                     | 30  |
| Table 2.3: Summary analysis of five commonly used Dataset for DDoS attack detection ..... | 32  |
| Table 3.1: CICIDS2017 Dataset Features.....                                               | 44  |
| Table 3.2: EMFFS candidate feature set .....                                              | 51  |
| Table 3.3: Final feature subset.....                                                      | 51  |
| Table 3.4: Details of feature subsets from literature.....                                | 51  |
| Table 3.5: Combined feature subsets from literature .....                                 | 52  |
| Table 3.6: Feature subset after majority voting .....                                     | 53  |
| Table 3.7: Feature subsets from Literature and Experimentation .....                      | 53  |
| Table 4.1: False positive rate results for feature subsets on LSTM+CNN .....              | 85  |
| Table 4.2: False positive rate results for feature subsets on LSTM+DNN.....               | 88  |
| Table 4.3: False positive rate results for feature subsets on CNN+DNN .....               | 91  |
| Table 4.4: Summarised feature subsets Accuracy .....                                      | 92  |
| Table 4.5: Summarised feature subsets false positive rate .....                           | 93  |
| Table 4.6: Hybrid Deep Reinforcement Learning, false positive rate computed results.....  | 101 |
| Table 4.7: Summarised HDRL CPU usage.....                                                 | 106 |
| Table 4.8: Summarised HDRL memory usage .....                                             | 106 |
| Table 4.9: Comparative validation results of the HDRLM.....                               | 107 |
| Table 5.1: Feature subsets from Literature and experimentation .....                      | 110 |

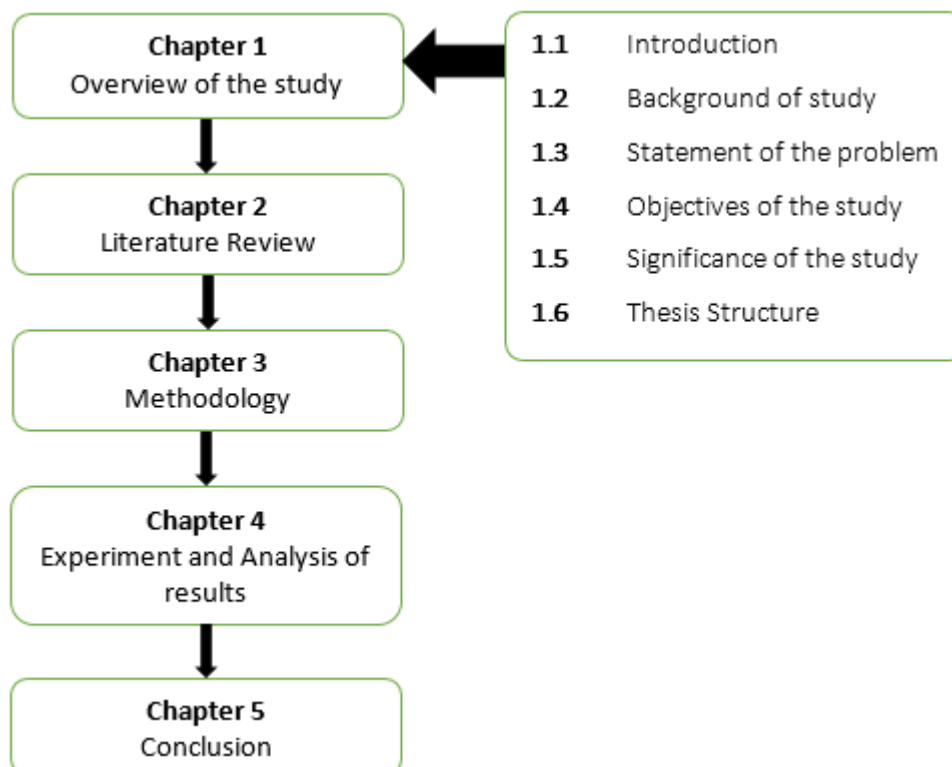
## List of Figures

|                                                                                                   |    |
|---------------------------------------------------------------------------------------------------|----|
| Figure 1.1: DDoS Attack .....                                                                     | 3  |
| Figure 2.1: Software Defined Network Architecture .....                                           | 11 |
| Figure 2.2: Intrusion Detection Systems categorisation.....                                       | 15 |
| Figure 2.3: Intrusion Detection System Architecture .....                                         | 20 |
| Figure 2.4: CICIDS2017 Test bed architecture. ....                                                | 34 |
| Figure 3.1: The Design Science Research methodology, Research process, and research methods. .... | 39 |
| Figure 3.2: DDoS attack Mechanism experimental design. ....                                       | 42 |
| Figure 3.3: CICIDS2017 instance distribution .....                                                | 46 |
| Figure 3.4: Feature selection Workflow.....                                                       | 49 |
| Figure 3.5: The Ensemble-based Multi-filter Feature Selection (EMFFS) method .....                | 50 |
| Figure 3.6: The agent-environment interaction in a Markov decision process .....                  | 57 |
| Figure 3.7: Deep Neural Network .....                                                             | 59 |
| Figure 3.8: Q-Learning Algorithm.....                                                             | 60 |
| Figure 3.9: Epsilon greedy algorithm trade-off.....                                               | 61 |

|                                                                                                     |     |
|-----------------------------------------------------------------------------------------------------|-----|
| Figure 3.10: Architecture design of Hybrid Deep Reinforcement Learning mechanism. ....              | 63  |
| Figure 3.11: Confusion Matrix .....                                                                 | 66  |
| Figure 3.12: DSRM application in Designing Efficient HDRL DDoS attack Detection for SDN .           | 69  |
| Figure 4.1: Feature subsets accuracy on LSTM+CNN.....                                               | 80  |
| Figure 4.2: Feature subsets accuracy on LSTM+DNN .....                                              | 81  |
| Figure 4.3: Feature subsets accuracy on CNN +DNN.....                                               | 81  |
| Figure 4.4: Confusion matrices for all feature subsets on HDRL mechanism with LSTM and<br>CNN ..... | 83  |
| Figure 4.5: Confusion matrices for all feature subsets on HDRL mechanism with LSTM and<br>DNN.....  | 86  |
| Figure 4.6: Confusion matrices for all feature subsets on HDRL mechanism with CNN and<br>DNN.....   | 90  |
| Figure 4.7: HDRLM Accuracy evaluated with FS3 .....                                                 | 94  |
| Figure 4.8: HDRLM Accuracy evaluated with FS4 .....                                                 | 95  |
| Figure 4.9: Confusion matrix for LSTM+CNN evaluated with FS3 .....                                  | 96  |
| Figure 4.10: Confusion matrix for LSTM+CNN evaluated with FS4 .....                                 | 96  |
| Figure 4.11: Confusion matrix for LSTM+DNN evaluated with FS3 .....                                 | 97  |
| Figure 4.12: Confusion matrix for LSTM+DNN evaluated with FS4 .....                                 | 98  |
| Figure 4.13: Confusion matrix for CNN+DNN evaluated with FS3.....                                   | 99  |
| Figure 4.14: Confusion matrix for CNN+DNN evaluated with FS4.....                                   | 99  |
| Figure 4.15: HDRL Average Reward .....                                                              | 102 |
| Figure 4.16: CPU status before the experiment .....                                                 | 103 |
| Figure 4.17: CPU usage during training with LSTM+CNN .....                                          | 103 |
| Figure 4.18: CPU usage during testing with LSTM+CNN .....                                           | 104 |
| Figure 4.19: CPU usage during training with LSTM+DNN .....                                          | 104 |
| Figure 4.20: CPU usage during testing with LSTM+DNN.....                                            | 104 |
| Figure 4.21: CPU usage during training with CNN+DNN.....                                            | 105 |
| Figure 4.22: CPU usage during testing with CNN+DNN .....                                            | 105 |
| Figure 4.23: Memory Usage before detection experimentation.....                                     | 106 |
| Figure 4.24: Memory Usage during detection experimentation .....                                    | 106 |

## Chapter 1 : Overview of the Study

This chapter presents the study topic, the background of the study, and what necessitates this study. Furthermore, the statement of the problem, the significance of the study, the main objective and sub-objectives of the study, and study questions are presented. The outline of the chapter is presented in a chapter map below.



### 1.1 Introduction

The computer network has never been more complex than it is now. This is due to numerous developments ranging from the Internet of things (IoT), Cloud computing, and corporate networks. One of the things that have made the traditional network more complex is the architectural design of vertical integration of bundling the control plane and the data plane inside the networking devices (Kreutz, Ramos, Veríssimo, Rothenberg, and Azodolmolky, 2015). According to Kreutz et al., (2015), the structural design of traditional networks reduces flexibility and hinders innovation and evolution of the networking

infrastructure and despite their continual usage, traditional networks are a nightmare to network managers.

The introduction of the Software Defined Network (SDN), which is a complete change on the model from the traditional network, presents a whole new network architecture that has allowed more versatility. According to Tang et al. (2018), the architectural arrangement of the SDN in which the control plane and data plane are no longer combined has made the network to be more flexibility, easy to maintain, and reusable in addition to giving network managers the ability to determine how the traffic flows within the network.

## 1.2 Background to the Study

Although SDN is enabling network capabilities, security has become a major worry because security is not yet incorporated into the architecture (Alsmadi and Xu, 2015). With the separation of the planes, each plane as well as the channels of communication between the planes has its security vulnerabilities. With its architectural design, the SDN is centralized where the controller is the central decision point responsible for establishing and terminating flows and paths. This creates a single point of failure in the network (Akhunzada et al., 2016). This means that an attack on the controller compromises the entire network.

Like any type of network, SDN is vulnerable to Distributed denial of service attacks (DDoS). The DDoS is an attack launched by multiple compromised computers as shown in Figure 1.1. The purpose of the attack is to send numerous requests to the target machine to use up its processing resources or compromise configuration thereby making its services unavailable to legitimate users (Behal and Kumar, 2016).

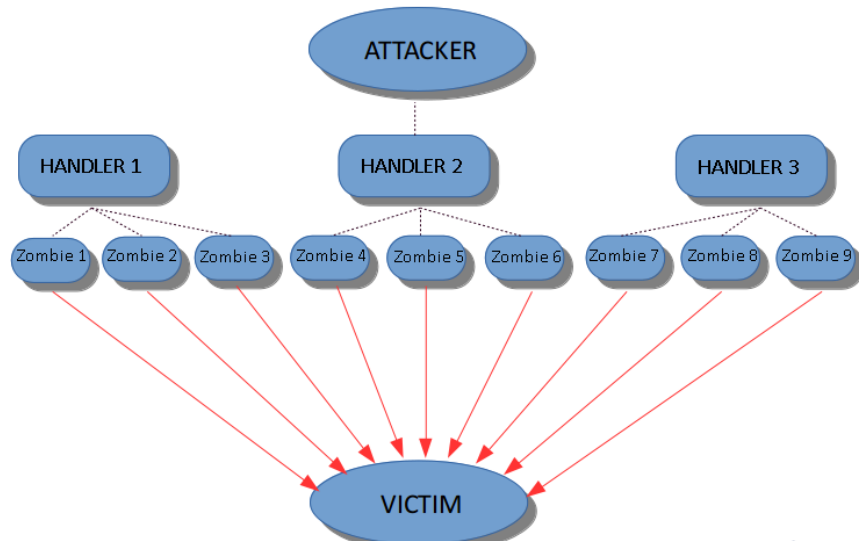


Figure 1.1: DDoS Attack (Source: Bhaya and Manaa, 2014)

The DDoS attack remains a major security concern for SDN. Therefore, one of the best ways to build a secure SDN environment is to integrate an intrusion detection system (IDS) into the design (Abubakar and Pranggono, 2017). Intrusion detection is a fundamental security measure to protect any network against attacks. An IDS is a security mechanism used to protect a network. The main functionality of the IDS is to identify any activity or behavior that is suspicious or intrusion from outside or within the network and then alert the network administrator so that necessary action is taken against the intrusion or the perpetrator of the intrusion. (Sengupta et al., 2013).

There is a need for continuous monitoring of network traffic in SDN to detect DDoS attacks as networks are used to transmit critical data. The amount of data being transmitted keeps on increasing each day and so does the demand for detection. To perform DDoS detection, there is a need for computational resources which come at a cost. Therefore, efficient use of computational resources in the detection of DDoS attacks is vital in the designing of an IDS (Arun Raj Kumar and Selvakumar, 2013).

Different machine learning approaches have been used to develop IDS for efficient DDoS attack detection such as support decision trees, artificial neural networks, k-nearest neighbour, vector machines, and clustering techniques (Sengupta et al., 2013). However, these techniques have

challenges to accommodate data dynamically for DDoS attack detection. Over the years more attention has been given to neural networks as a promising technique for DDoS detection.

Neural networks are a subset of machine learning modelled from the human brain and follow the pattern that the human brain uses to classify objects in real life (Karatas et al., 2019). The architecture of the neural network is made up of three basic layers namely, the input layer, the hidden layer and the output layer, the three layers are connected by nodes forming a network (Ozerdem et al., 2017). Over the years neural networks have evolved with the introduction of different types of variants such as Deep Neural Networks, Convolutional Neural Networks and Recurrent Neural Networks (Ketkar, 2017; Muhammad Diyan, Murad Khan, 2020).

According to Shone et al., (2018), deep learning has emerged as one of the most commonly used in neural network methods because deep learning algorithms have the capabilities to learn on their own by extracting features from unlabelled data. This self-learning capability is an example of unsupervised learning, which enables the deep learning algorithms to identify the relevant features from a dataset without requiring any human intervention and this has shown to be an important element in using deep learning in the design of intrusion detection systems (Alrawashdeh and Purdy, 2016).

Another machine learning method of interest is Reinforcement Learning. This is a machine learning method that when exposed to an environment, the algorithm learns by itself by taking try-and-error actions and builds an experience from the actions taken (Arulkumaran et al., 2017). The experience is then used to make the best decisions when exposed to the environment again. The capabilities of Reinforcement Learning have further been enhanced by combining it with other methods such as Deep Learning to introduce Deep Reinforcement Learning. Deep reinforcement learning has been used for the classification of images and playing games (Hausknecht & Stone, 2015). According to the work of Alansary et al., (2019), Deep Reinforcement Learning that used a DNN in its neural network was used to classify images and perform medical scanning. DRL has also been used in intrusion detection mechanisms by using CNN for the neural network component

(Hossen & Janagam, 2018). The use of one type of neural network comes with its limitations that can be overcome by combining two or more variants of neural networks.

This study designed a hybrid Deep Reinforcement Learning Mechanism (HDRLM) by combining two heterogeneous neural network classifiers to accurately detect DDoS attacks for SDN while using minimal computational resources. We propose an IDS that is placed on the control plane of the SDN architecture. By securing the control plane we reduce the risk of a single point of failure.

### 1.3 Statement of the Problem

The emergence of the SDN brings with it numerous possibilities to strengthen network security. However, SDN also brings a much more dangerous increase in potential threats (Tang et al., 2016). One of the threats is due to its architecture of centralised control, the design creates a single point of failure. With DDoS attacks still posing a major network security challenge, it is a major security concern for SDN.

Attackers continue to come up with new DDoS attack techniques and introduce DDoS attacks which are unknown to the existing mechanisms thereby lowering the detection accuracy of the mechanisms. Furthermore, the increase in the volume of network traffic over the years has increased the potential for DDoS attacks in SDN which in turn increases the cost of detection as detection requires computational resources.

This challenge necessitates the development of an efficient detection mechanism for accurate DDoS attack detection of known and unknown DDoS attacks in SDN to address this security concern. The purpose of this study, therefore, is to design an efficient detection mechanism for DDoS attacks in SDN. Software Defined Networking is probably one of the best developments in recent years in the field of networking. It offers many solutions to the future of the Internet and securing it will offer greater benefits.

## 1.4 Objectives of the Study

The main aim of the study was to design an efficient intrusion detection mechanism for the detection of known and unknown DDoS attacks for SDN. To achieve this objective, the following three sub-objectives were first achieved.

### 1.4.1 Research sub-Objectives

1. To identify and evaluate the best feature selection technique and best features for DDoS attack detection from a dataset to achieve high classification accuracy in Software Defined Network.
2. To determine efficient classifiers based on deep learning and reinforcement learning
3. To design an efficient hybrid DDoS attack detection mechanism that will use minimal computational resources while accurately detecting DDoS attacks in SDN.

### 1.4.2 Research Questions

To achieve the objective, the study had to answer the question of how to design an efficient intrusion detection system (IDS) for DDoS attacks detection for Software Defined Networks with high accuracy in detecting new attacks without the need for retraining.

1. What are the best feature selection techniques and best features that give high accuracy in the classification of DDoS attacks?
2. Which classifiers based on deep and reinforcement learning are efficient?
3. How can deep learning and reinforcement learning algorithms be combined to design a hybrid IDS for DDoS attack detection in SDN?

To address objective (1), a review of literature on available feature selection techniques was first undertaken. From the review, the best feature selection technique was identified and used in an experiment to select features from the CICIDS2017 dataset. Under the same objective, a review was done of previous works on which features were used in detecting DDoS attacks. The feature subsets

from the experiment together with the one identified from the previous works were evaluated to identify the best feature subset for evaluating the designed HDRLM for DDoS in SDN.

To accomplish objective (2) a literature review was undertaken of the different algorithms for deep learning and reinforcement learning. The algorithms were evaluated to establish the most efficient algorithms by subjecting them to an experiment. The best feature subset identified in the first objective was used to evaluate the performance of the algorithms.

Objective (3) was achieved through experimentation in which the algorithms identified in the second objective were combined seamlessly to create an HDRLM that is more efficient and effective. Three permutations were used in the experimentation and evaluated based on parameters to identify the best combination. The performance of the designed HDRLM was then validated by comparing it with other available state-of-the-art mechanisms.

### 1.5 Significance of the Study

This study produces an artifact that potentially adds value to the cyber security research and industry in dealing with DDoS attacks. As the industry embraces the Software Defined Networks the threat of DDoS attacks remains a major concern and has also increased the need for mitigation. To mitigate DDoS attacks the first step is to be able to identify the presence of the attack. Detection is thus a critical element and initial step in the process of dealing with DDoS. Therefore, this study presents a solution at the initial stage of dealing with DDoS attacks.

Some mechanisms have been designed in the past aimed at detecting DDoS attacks on the SDN which requires a significant investment to adopt. While efforts to deal with DDoS attacks are being put in place, there is a daily increase of traffic passing through networks every day which increases the presence of DDoS attacks. Therefore, the continuous increase of traffic has also increased the need for detection that requires computational power. So as the demand for detection increases the demand for computational power also increases and comes at a cost. It is expected that the traffic will continue to increase exponentially, therefore there is a need for an efficient mechanism to Detect DDoS attacks.

This study aimed at providing an efficient mechanism that uses less computational resources while accurately detecting DDoS attacks in SDN. The efficient mechanism reduces the cost of detection and if implemented it offers a better return on investment. The study also adds to the body of knowledge in approaches to develop a hybrid mechanism and the use of Deep Reinforcement Learning for DDoS detection in Software Defined Networks. It opens more research areas that improve the mechanism and improve the detection of other types of attacks.

## 1.6 Thesis Structure

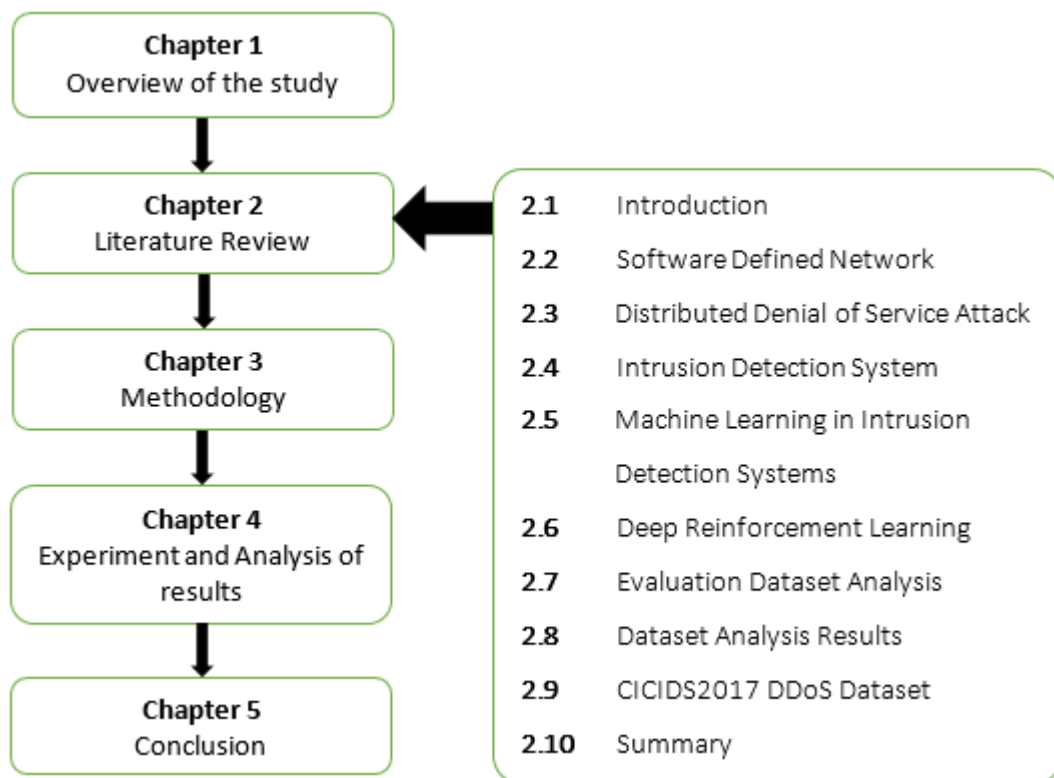
The presentation of the rest of the thesis is structured as follows. Chapter 1 – Introduction, presents an overview of the topic and the problem that the emergency of Software Defined Network presents in the face of the existence of DDoS attacks. Then it explains how this study addresses the problem. Chapter 2 which is Literature Review analyses available past research work that has been conducted in Software Defined Networks, detection mechanisms for DDoS attacks, Deep Reinforcement Learning, and dataset for evaluation of detection mechanisms through the published literature. Chapter 3- Methodology covers the approach used in the study in designing the HDRLM for DDoS detection in Software Defined Networks. The approach chosen is the use of an experiment and the tools. In Chapter 4-Evaluation Dataset, preparation of the dataset used for evaluating the designed HDRLM and the selection of features is presented while in Chapter 5- Experimentation, the step-by-step approach in the implementation of the HDRLM from setting up of the environment, the structure of the HDRLM, and evaluation of the HDRLM is presented. Further presented in Chapter 5 are the results and an analysis of the results. Chapter 6 - Conclusion summarises the study and makes recommendations for future work.

## Chapter 2 : Literature Review

### 2.1 Introduction

Since the introduction of the Software Defined Network (SDN) in the year 2005, research work has demonstrated how this new architecture comes with more gains compared to the traditional network. The gains include the programmability which allows for customised configuration of the network and services (Shi and Yang, 2017), the use of OpenFlow which removes vendor-based restrictions allowing the use of devices from different vendors, and ease of scalability (Kreutz et al., 2015). The gains come due to its structural design and capabilities. However, other researchers have raised security concerns the SDN faces.

This chapter reviews literature on SDN in terms of gains and challenges including the security concerns of DDoS attacks, Distributed Denial of Service (DDoS) attack, the use of Intrusion Detection Systems (IDS) to address the challenge of DDoS attacks in SDN, the Deep Reinforcement Learning algorithms used for detection mechanisms, and the available datasets used for the evaluation of intrusion detection mechanisms for DDoS attacks. The chapter further presents an analysis of the available evaluation dataset and the outcome of the analysis. The CICIDS2017 DDoS dataset is also discussed in detail This outline is presented in the chapter map below.



## 2.2 Software Defined Network

Since the introduction of the computer network, no other change to its architecture had taken place until the emergence of the SDN (Abubakar and Pranggono, 2017). The SDN is referred to as a paradigm shift in computer networking from the traditional network architecture where the decision making and the traffic forwarding are all bundled at the same level or device to a new architecture that unbundles the network architecture by introducing three distinct levels namely the application plane, the control plane and the data plane (Kreutz et al., 2015). Figure 2.1 presents the SDN architecture with the three separated planes (Dongsoo, 2015). Network Applications that are used to configure and manage the network environment that determines how the network will work are placed on the application plane. While the control plane consists of the SDN controller which is the brain of the network, and all decisions are made on this plane. The data plane is where the forwarding of traffic is performed by following the instructions from the control plane.

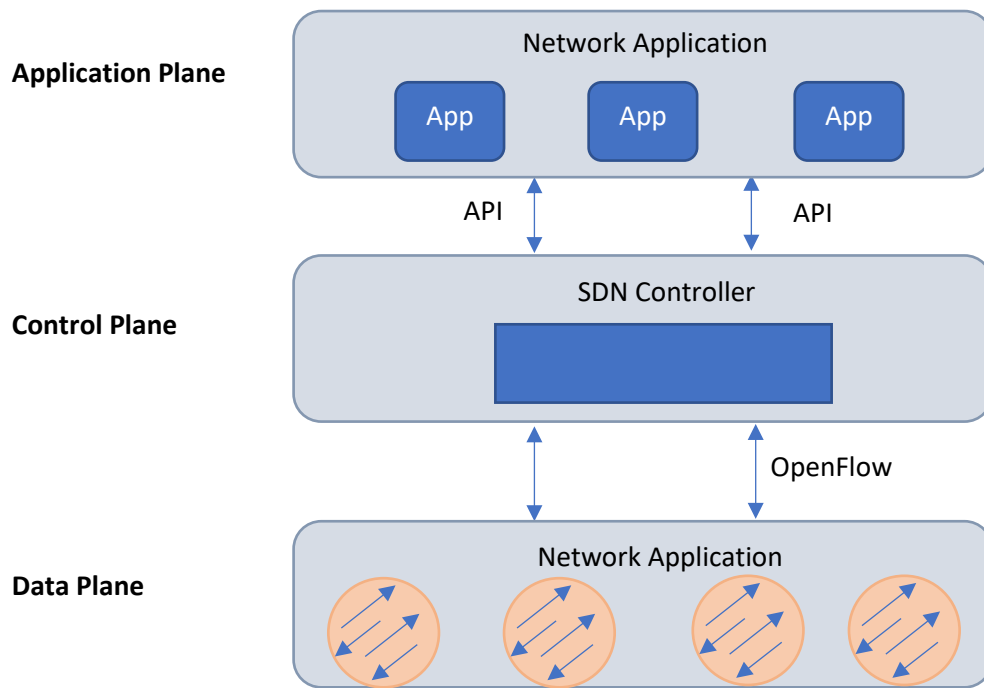


*Figure 2.1: Software Defined Network Architecture*

In Figure 2.1, the Application Programmable Interface (API) is used to communicate between the application plane and the control plane and the communication between the control plane and the data plane is through OpenFlow. OpenFlow is a standard protocol that determines how packets are handled by providing a communication channel between the SDN controller and the network devices on the data plane (Ibrahim, 2017)

The change brought about by the SDN has come with many benefits for network managers due to its architectural design as well as its capabilities. One of the capabilities available in SDN is programmability, according to Abdelaziz et al., (2017), programmability in SDN provides network managers with the flexibility and control that cannot be found in the traditional network. The programmability also provides for unlimited future development of abstract applications and services thereby making it more adaptable to the ever-changing needs in connectivity (Wang et al., 2015). As argued by Wang et al., (2015), the SDN presents the opportunity to build networks that are scalable and flexible as anyone can customise the network and control it to meet their needs. Furthermore, the architectural design provides the capability to control several networks from one

controller and to add more networks without a major change on the controller (Abdelaziz et al., 2017).

On the flip side, research has revealed security concerns of the SDN due to the SDN architecture and its capabilities. According to Akhunzada et al., (2016), the programmability of the SDN which allows anyone to create a program for the controller exposes the whole system to malicious code. This vulnerability is minimal in the traditional network as the network applications, the control and the data plane that are in one device are proprietary specific. A security concern on the architectural design of having the controller as the central point of decision making and that it poses a vulnerability of a single point of failure is discussed by (Akhunzada et al., 2016). While this architectural design is seen as a vulnerability, Ibrahim (2017) argues that the architectural design of the SDN presents more advantages to dealing with many attacks effectively. He argues that the SDN controller having overall control of the entire network can easily detect and deal with any abnormal traffic. And on programmability, Ibrahim (2017) argues that it presents the network operator with the capability to program controls and logic on how to deal with any type of attack without waiting for a software update as is the case with other intrusion detection mechanisms. The SDN architecture comes with both advantages and disadvantages as far as network security is concerned.

While the researchers acknowledge the gains achieved by the SDN, the vulnerability remains and with the prospect of implementing the SDN in enterprise networks such as Google, it is imperative to look at efforts to address the vulnerabilities. Some of the research aimed at addressing some of the vulnerabilities have been conducted by Ajaeiya et al (2017) who proposed an IDS that gathers and analyses data from the OpenFlow switches. Mousavi and St-Hilaire, (2015) propose the use of entropy variation of the destination IP address to detect DDoS attacks in SDN. Niyaz et al (2016) introduce a mechanism using Deep Learning to detect DDoS attacks on the control plane as well as the data plane. Despite all these efforts, the DDoS attack remains a major threat to the

Software Defined Networks like any other network. A discussion on DDoS attacks is presented in the subsequent section.

### 2.3 Distributed Denial of Service attack

Literature paints a picture that DDoS attacks remain a major challenge for SDN due to its architectural design of having the controller as the decision plane. A DDoS attack against the controllers paralyses the entire network. This is a major security concern as the rate of DDoS attacks continues to grow substantially. In 2016 Cisco predicted that by 2020, there will be 17 million DDoS attack events, a threefold increase from 2015 (“Cisco Visual Networking Index Predicts Near-Tripling of IP Traffic by 2020-Cisco”, 2016) making it more complicated because, as Niyaz et al., (2016), explains, the nature of attacks has also changed, with multi-vector flooding replacing single-vector flooding.

The DDoS attack is intended to interrupt the normal service of a network or server by flooding the target with fraudulent traffic that overwhelms the system, thereby rendering the service unavailable to legitimate users (Fouladi et al., 2016). The attack is done in a way that the attacker uses several hosts that are compromised and remotely commands them to send a flood of superfluous network requests to the target. In an attempt to address all the requests, the target uses up all its critical resources such as the CPU and Memory. In the end, the target is overwhelmed and cannot respond to genuine requests. The DDoS attack is the most sophisticated. As Fouladi et al., (2016) state, the traffic requests sent to the target are similar to normal requests. This makes it hard for the target to differentiate between an attack and normal traffic. This sophistication in DDoS attacks continues to be a nightmare for networks. DDoS attacks are in different types such as UDP, ICMP, TCP SYN flooding, NTP amplification, and ping of death (Aslansefat et al., 2020).

According to Kalkan et al., (2017), a DDoS attack on SDN can take place due to the SDN architecture, as well as on the communication channels. They outlined five scenarios in which a DDoS can take place:

1. The first scenario is the communication channel between the controller and the data plane device as a target. This attack takes advantage of the OpenFlow protocol in which any request from unknown IP has to be forwarded to the controller for deciding on the action to be taken, therefore the attacker can use spoofed IP addresses to launch an attack which will overwhelm the communication channel between the controller and the data plane device.
2. The second scenario is the link between devices and the data plane. According to Kalkan et al., (2017), devices in DSN data plane that is in the same network share information of the flow table. Therefore, an attack on one switch in the data plane forces it to keep updating its flow table with new entries and persistent sharing of the new information causes congestion in the communication between the switches.
3. The third scenario is where the switch is the target. Kalkan et al., (2017) state that in this attack the continuous demand to update the flow table with new entries demands the use of the switch memory leading to the depletion of the available resource.
4. The fourth scenario is when the controller is a target which results into flooding of the controller with requests. In trying to resolve all such requests, the controller is overwhelmed and cannot take care of legitimate requests (Kalkan et al., 2017).
5. The fifth scenario is when the controller is the target. The attacker uses different switches managed by the same controller to send the requests to the controller. This attack is the most sophisticated as it comes from different switches and appears as normal traffic. This type of attack is named blind DDoS attack (Kalkan et al., 2017).

The scenarios presented above are all concerned in securing the SDN network. However, an attack on the controller is the most worrisome as it collapses the entire network. Any efforts to deal

with DDoS attack starts by detecting the presence of the attack. Therefore, the use of an Intrusion Detection System is necessary

## 2.4 Intrusion Detection System

An intrusion detection system (IDS) was invented as one way to secure networks. An IDS is used to monitor the traffic passing through a network and is built with capabilities to distinguish normal traffic from malicious traffic and alert the network manager of any traffic classified to be malicious. Two main types of IDS are categorised based on how they function. The first type is called signature-based and the second type is called anomaly based. However, sometimes the IDSs are categorised from different perspectives such as where they are placed, their structure, how they react, and how frequently they are used (Hossen and Janagam, 2018). Figure 2.2 depicts how the categorisation of IDS is based.

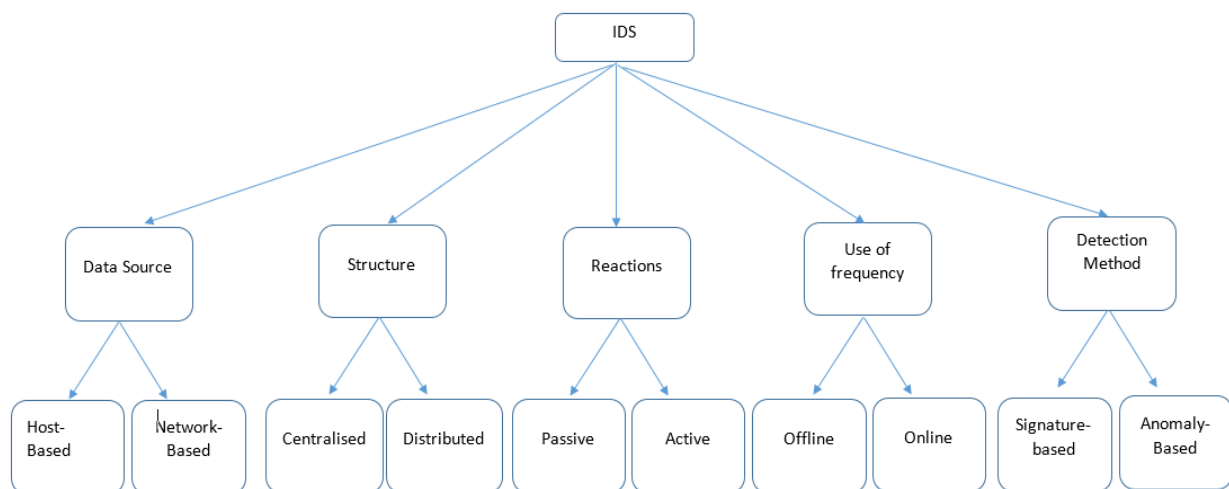


Figure 2.2: Intrusion Detection Systems categorisation (Source: Hossen and Janagam, 2018)

The IDSs categorised by data source are determined by where they are placed within a network. The two types based on data source are Network-based Intrusion Detection System (NIDS) which is used to monitor the entire network and the Host-based Intrusion Detection System (HIDS) which monitors traffic for a specific host within a network (Jain, 2018). The NIDS is applicable in a setup where there are more devices connected to the same network. The NIDS is, therefore,

attached to the network switch while the HIDS works well with individual workstations where detection is important at host. When the HIDS is configured on a workstation and uses computational resources, it may cause under-performance of the workstation.

Categorising IDSs based on the structure has one category as centralised IDS and another as distributed IDS. A centralised IDS is used on a node or point from which monitoring is done whereas a distributed IDS uses different nodes or points to monitor the network. The centralised IDS is utilised in a configuration where the network is divided into segments such as Demilitarised Zone (DMZ), corporate network, and public WIFI. Instead of having an IDS for each of these segments, only sensors are configured in the segments with the monitoring done at a central point. In the case of a distributed IDS, each of the segment has its own monitor. The distributed IDS is ideal for a large network that is segmented and covers a wide geographical area.

Using the reaction perspective, the IDSs are categorised into passive and active. A passive IDS is the one that just monitors the traffic but does nothing when there is malicious traffic while the active IDS does not only monitor but also takes specified action once malicious traffic is detected. Such action includes alerting the network manager or in a more advanced approach blocking the source of the attack. The active IDS is suitable for networks where there are no personnel available to continuously monitor the network logs and for networks whose unavailability does not cause serious damage or loss because the IDS shuts off the access once an intrusion is detected. Passive IDSs are best suited for networks that need to be accessible no matter the circumstance. For networks such as airlines and medical the passive IDS is mostly used.

With the use of frequency, IDSs are categorised into either Offline or Online. An offline IDS uses analyses traffic that has already passed through a network while online IDS is placed in the network and analyses data as it is passing through the network. The offline IDS is used to improve the configuration of the IDS as the data that is analysed might have caused the damage or loss by

the time it is being analysed. Online IDS is used for network that require instant action to resolve an intrusion attempt.

Last is the general categorisation that is based on the detection method: the signature-based IDS and the anomaly-based IDS. The signature-based IDS is also named misuse. It uses an existing database of known attacks to match with the signatures of the traffic passing through a network or a host (Moustafa and Slay, 2015). The downside of the signature-based IDS is that it cannot detect unknown attacks. That is where the anomaly-based IDS becomes more relevant. The anomaly-based IDS first establishes a profile of what is regarded to be normal network traffic. Then it monitors traffic in the network and compares it with what is established to be normal traffic. Any diversion from the normal is flagged to be an anomaly or an attack. Anomaly-based IDS are best suitable for detecting intruders from within the network such as copying of restricted data from a system whereas signature-based IDS is best suitable for detection of intruders from outside the network.

IDSs are used to detect any security breach occurring in a network from both external and internal users. An IDS can also be used to detect violations against security policies set by a network owner for example a restriction to copy data that has been classified as confidential from a system. The IDS is also used to gather details of security breaches by creating logs that are used for forensic investigations, and possibly evidence for prosecution (Jain, 2018).

The anomaly-based IDS system has been recommended for the detection of DDoS attacks due to the sophistication of the attack as attackers keep on developing new techniques of launching DDoS attacks.

Researchers have proposed mechanisms in the detection of DDoS attacks using different approaches. Kale and Choudhari (2014) propose that an ensemble of classifiers achieved better results by combining classification algorithms based on machine learning. However, Yuan, C. Li, and X. Li (2017) argue that the narrow representation of models limits traditional machine learning

techniques. Instead, they propose a deep learning-based DDoS attack detection approach called Deep Defense. Yuan, C. Li, and X. Li (2017 p.1) further claim that “Deep learning approach can automatically extract high-level features from low-level ones and gain powerful representation and inference”. Adding to this, Rukavitsyn, Borisenko, and Shorov (2017) propose a mechanism based on reinforcement learning due to its self-learning approach through an adaptive algorithm.

Taking a slightly different approach Barati et al., (2015), Cepheli et al., (2016), Jantila and Chaipah, (2016) propose a hybrid DDoS attack detection framework. Considering a hybrid approach used in Barati, Abdullah, Udzir, Mahmod, and Mustapha’s (2015) work, they achieved good results with an accuracy of 99.9%, Precision at 100%, Recall at 99.8, FPR at 0.002, and F-Measure 99.9 %. The hybrid approach has shown a lot of potential in accurately detecting DDoS attacks. This is supported in the work of Cepheli et al. (2016), Branch and Azad (2014), and Barati, Abdullah et al. (2015) with accuracy ranging from 98% to 99.9%. The overall performance of Hybrid detection mechanisms proves to achieve high accuracy between 98% and 99.9%. In using the hybrid approach, research leverages on the advantages of the combined approaches and advantages of one approach potentially compensates for the disadvantages of the other approach. However, when using the hybrid approach, it is important to also consider the shortfalls of each of the combined approaches, because it can potentially double the shortfall. Some of the shortfalls of hybrid detection mechanisms are that they become heavy and that they demand more computational resources to function.

In designing a detection mechanism, it is important to consider the efficiency of the detection mechanism apart from achieving high accuracy. An efficient detection mechanism uses minimal computational resources such as CPU and Memory to function and in addition performs the detection in minimal time possible. Therefore, CPU usage and Memory usage are some of the metrics to be taken into consideration when evaluating a detection mechanism for efficiency. One way to improve the efficiency of a mechanism, according to Chuan-long, Yue-fei, Jin-long, and Xin-

Zheng (2017), is to use a hybrid approach as demonstrated in their proposed mechanism that obtained an accuracy of 99.5% and 11444 seconds to train the mechanism. Similarly, in their work, Niyaz, Sun, and Javaid (2016) proposed the use of a stacked autoencoder by building three levels of feature selection. Using this approach, they achieved an accuracy of 95.65%, a training time of 524s and a detection time of .0835s in a Software Defined Network environment. According to Phan, Bao, and Park (2017) work, using Hybrid of Support Vector Machine (SVM) and the Self-Organizing Map (SOM) achieved a DDoS attack detection accuracy rate of 98.13%, a false positive rate of 3.85%, and CPU usage of 25% in an SDN environment.

The reviewed literature shows that hybrid detection mechanism has not only achieved high accuracy but enhanced efficiency in the detection mechanism as well. Therefore, the use of a hybrid approach has empirically been recommended.

## **2.5 Machine Learning in Intrusion Detection Systems**

The IDS is made up of four components namely the sensors, the analyser, the monitor and the management console. These components work collectively in the detection of any intrusion. Figure 2.3 presents the general architecture of the Intrusion Detection System in which the sensors are used to collect the traffic such as packets from a network. The analyser is used to classify the traffic as either malicious or benign while the monitor is there to send an alert once the analyser identifies traffic to be malicious. The management console is used to configure the IDS according to a given security policy (Jain, 2018).

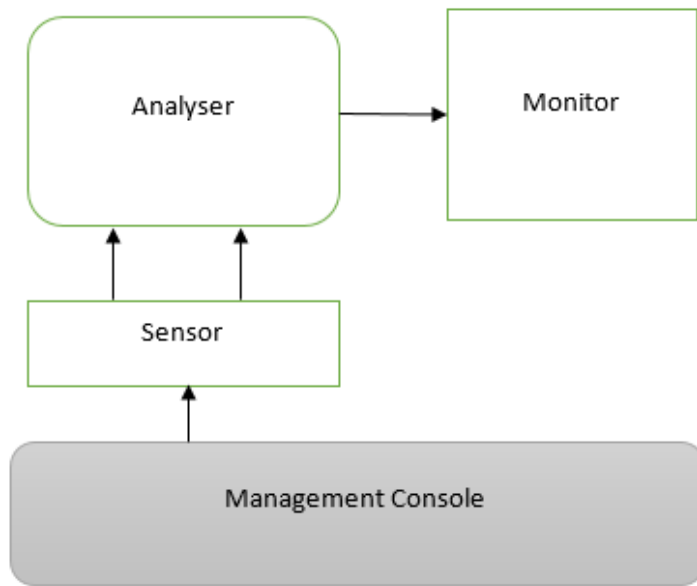


*Figure 2.3: Intrusion Detection System Architecture*

When the IDS was first introduced, the analyser was made of applications that required a network manager to go through the logs and isolate the classified traffic. This required that the network manager be available throughout the use of the network. However, with the growth of traffic going through networks, it has become impossible for a network manager to be present and look at the logs. This human intervention has been replaced with machine learning. Machine learning is a concept of using machines to perform tasks similar to human beings without the machine being explicitly programmed but through learning and improving the capability to perform a task by being trained using data (Batta, 2020). Since the introduction of Machine Learning in intrusion detection systems, the concept has gained ground through research and industrial implementation and has improved the performance of the IDS (Benmessahel et al., 2018). The three primary models of machine learning are supervised learning, unsupervised learning, and reinforcement learning. (Batta, 2020). According to Batta (2020), in Supervised Learning the machines learn a task from a function that maps input to an output forming an input-output pair and the algorithms are trained using data that is labelled and consists of training examples. Unsupervised Learning on the other hand, learns without labelled data and there is no correct answer in its predictions, the algorithm is left to learn the data by itself and create patterns from the data.

Unlike Supervised and Unsupervised Learning, which learn from data, reinforcement learning is self-learning by creating a model that is improved through retro-feeding it based on rewards received from interacting with the environment (Gou and Liu, 2019). From these three Machine Learning paradigms, many other algorithms such as Decision Trees (G. Kim et al., 2014), Random Forest (Alkasassbeh et al., 2016), Support Vector Machine (Phan et al., 2017) and Artificial Neural Networks (Slay and Moustafa, 2015) are developed. Other specialised algorithms have been developed to solve different types of problems, one such algorithm is Deep Reinforcement Learning. The subsequent sections discuss Deep Reinforcement Learning.

## 2.6 Deep Reinforcement Learning

Deep Reinforcement Learning is a subset of Machine Learning that has its roots in Reinforcement Learning and Deep Learning. Reinforcement learning has its foundation in the behavioural psychology of learning by interacting with the environment (Arulkumaran et al., 2017). The learning is achieved by performing several trial and error actions. Adopting the same approach in the field of Artificial intelligence, the RL is built on the principle of the Markov Decision Process (MDP) to solve sequential problems and is made up of five main components also referred to as the Markov properties namely, an agent, the environment, state, set of actions, and reward. The agent operates in an environment that presents states that the agent observes and based on the state, the agent takes a specific action and a reward is given to the agent based on the action taken as the environment moves to another state. The agent aims at attaining the highest possible number of rewards. Therefore, the reward drives the actions of the agent (Arulkumaran et al., 2017).

For the agent to take any action that will yield maximum reward, it is informed by a policy. A policy is a mapping of state and possible actions with corresponding rewards for every action (Arulkumaran et al., 2017). When an agent is first introduced in an environment the policy is non-

existent and is developed as the agent learns the environment by trial and error. The aim is to have an optimal policy that will maximize reward per given state.

According to Arulkumaran et al., (2017), there are two approaches that the agent uses to follow a policy to take any given action in any given state. The first one is value function and the other one is policy search. In a value function, any given state is mapped to a value or values of the state and an agent operating in an environment will take action in a given state and will observe the value resulting from the action taken. When the value is not available, the agent estimates the value it is mapped with the action taken. In case of a state which has a value mapped available, the agent will take the action and while observing the value and if the resulting value is better than the available value, the mapping is updated with the new value (Fan et al., 2019). This method is used by the Q-Learning algorithm, one of the most popular Reinforcement Learning algorithms which have the value function in a table referred to as the Q- values table (Fan et al., 2019). For policy search, any given state is mapped to a specificities action and when the agent is presented with a given state from the environment, the agent refers to the mapped action for the given state (Arulkumaran et al., 2017).

This approach, therefore, shows that an agent is as good as the policy it follows. Research on how to obtain an optimal policy and improve the value approximation has shown that this is best done by using Neural Networks. The Neural networks are said to have an inherent capability to learn low-dimensional feature representation and a powerful function for approximation (Arulkumaran et al., 2017). Therefore, neural network has been added as a component to the Reinforcement Learning for approximation of the optimal values and the optimal policy. This inclusion of the neural networks to the RL is the emergence of Deep Reinforcement Learning. According to Ghiassian et al., 2020, Deep Q Network (DQN) is a popular and reliable Deep Reinforcement Learning algorithm. In using the DQN, researchers mostly use the Convolution Neural Network (CNN) for value approximation. In their work, Hossen and Janagam, (2018), used DQN in designing a detection

mechanism for a range of attacks in which they used the CNN for value approximation. Similarly, Gou and Liu, (2019) used CNN for a proposed efficient DQN algorithm. Apart from CNN, Deep Neural Network has also been used for value approximation (Hu et al., 2020). A theoretical analysis of DQN presented by Fan et al (2019) outlines past research work that used DNN in DQN for value approximation.

## 2.7 DDoS Attack Benchmark Dataset

In designing a DDoS attack detection mechanism, the use of a dataset in place of live network traffic to evaluate the mechanism has been a more adopted approach (Sharafaldin et al., 2017). This is because evaluating proposed mechanisms on real network traffic remains a challenge due to network owners not wanting to compromise their network through exposure to cyber threats and to protect network users' privacy which also borders on ethical issues (Gharaibeh and Papadopoulos, 2009).

Since the concept of using a benchmark dataset in place of real network traffic, from 1998 until now, eleven datasets have been made available (Abdulraheem and Ibraheem, 2019) and researchers have used them in their work in Intrusion Detection Systems (IDS). However, selecting a benchmark dataset to be used in the validation of the DDoS attack detection mechanism is critical because a dataset affects the performance of the mechanism (Behal and Kumar, 2016). It was, therefore, pertinent for the study to use the appropriate dataset for evaluating the detection mechanism as the available dataset have inherent differences due to their configuration of the environment they are generated in, the technology used, and presentation format (Gharib et al., 2017).

To do this, the study analysed the five most-used datasets namely KDD CUP 99, NSL KDD, DARPA DDoS 2009, CAIDA DDoS, and the CICIDS2017 from the literature surveyed to determine the most appropriate dataset for evaluation of a DDoS attack detection mechanism in Software Defined Networks.

Using an evaluation framework for IDS and IPS datasets proposed by Gharib, Sharafaldin, Lashkari, and Ghorbani (2017), the study considered eight out of the eleven elements follows:

- **Complete Network Configuration:** An offline dataset must be generated from a network that is well and complete configuration with both hardware and software as a real network. The devices must include personal computers or hosts, servers, switches, routers and firewalls, and the appropriate operating systems. This essentially provides a more realistic behaviour of an attack in a real network.
- **Complete Traffic:** A dataset should be made of network traffic generated from a sequence of packets from a source to a destination. The traffic can be realistic, pseudo-realistic, or synthetic depending on the approach used to generate it. In this study, realistic traffic was considered as it gave data with true characters from a real-world network.
- **Labelled DDoS Dataset:** a well-labelled DDoS attack dataset is crucial to the evaluation of a DDoS attack detection mechanism, as a wrongly labelled dataset casts doubts on the performance of the mechanism when types of attacks are not distinctively identified and just labelled as attack and benign. Therefore, the performance of the mechanism would be generalised and not based on DDoS.
- **Complete Capture:** A dataset must capture all traffic in its totality. This is essential for the evaluation of a detection mechanism. A dataset that partially captures the traffic and removes non-functional or non-labelled data? does not give an accurate evaluation of a detection mechanism because the omitted traffic is important in calculating the false-positive rate of a detection mechanism.
- **Available Protocols:** With many types of network traffic flowing through the network and covering a wide range of protocols, a detection mechanism performs differently based on the traffic flowing through the network. A mechanism detecting TCP traffic protocol would be different from that under UDP traffic protocol. In a TCP channel, complete interaction

between the source and the destination and delivery of complete packets is expected while in a UDP the interaction between the source and destination is not guaranteed. In the same way, the timing of the delivery varies based on the protocol in use. Therefore, a detection mechanism that predicts based on the data sent and received and the time is taken would perform differently based on the protocol. Additionally, a mechanism is designed based on the domain they are to function. A detection mechanism for SDN can best be evaluated with a dataset containing flow-based protocol as it is the protocol used in the network architecture.

- **Feature set:** A dataset is generated to be used by researchers to evaluate detection mechanisms for different types of attacks. Therefore, the dataset must have a feature set for each of the attacks it is to be used for and must be well documented.
- **Heterogeneity:** A dataset can be generated from different sources in a network where logs are stored, ranging from hardware such as hosts, switches and routers, to software such as operating systems. It can be generated from a single log or by combining multiple logs from different sources. This has a direct impact on the evaluation of a detection mechanism as a dataset from multiple sources which is heterogeneous, would be appropriate to evaluate a detection mechanism that can be used for the entire network.
- **Availability of Metadata:** An accurate documentation of a dataset provides researchers with information that will make the researcher understand the environment, equipment, and the nature of attacks used in the generation of the dataset, this can be used to determine the usability of the dataset in evaluating a detection mechanism. Documentation for a dataset should be complete and must provides information on network configuration, attacker and victim operating systems, attack scenarios, and other critical information.

## 2.8 Evaluation Datasets Analysis

To obtain an accurate evaluation of the performance of a detection mechanism, the dataset used for evaluation must have more realistic scenarios of network traffic and be well normalised

without creating bias under a detection mechanism. Selecting an appropriate dataset for the DDoS detection mechanism remains a challenge as there is no proposed approach and criteria to follow. The study analysed five popular and widely used datasets in DDoS attack detection research namely; KDD Cup 99, NSL KDD, DARPA, CAIDA, and CICIDS2017. Considering the relevance of the dataset based on its inherent attributes for benchmarking DDoS attack detection mechanisms, we will analyse each of these datasets using the criteria presented in Section 2.6 as proposed by Gharib et al., 2017.

### 2.8.1 KDD CUP 99

Since 1999, researchers working on intrusion detection mechanisms including DDoS detection mechanisms have used a well-known dataset named KDD Cup 99. The KDD Cup 99 is a publicly available dataset created by the Massachusetts Institute of Technology (MIT) Lincoln labs through the Defense Advanced Research Projects Agency (DARPA) intrusion detection evaluation program (Arun Raj Kumar and Selvakumar, 2013). KDD Cup 99 was first used by the University of California in 1999 during the third International Knowledge Discovery and Data Mining Tools Competition ("KDD Cup 1999 Data - Information and Computer Science University of California, Irvine", 1999), from which the name KDD was derived. Since then the use of the KDD Cup 99 dataset has become extensive in anomaly detection research. Researchers working on DDoS attack detection mechanisms have used it as a benchmark dataset to evaluate their proposed approaches (Alrawashdeh and Purdy, 2016; Arun Raj Kumar and Selvakumar, 2013; Xiao, Qu, Qi, and Li, 2015).

The dataset has 41 features with each instance labelled as either normal or attack. The attacks are categorised into four classes namely DoS, Probe, User-to-Root (U2R), and Remote-to-Local (R2L) (Jeya, Gifty and Ravichandran, 2012). It is a labelled dataset with a more realistic representation of network traffic, and due to this it also became a basis on which other databases were created such as the NSL KDD and Kyoto 2006+ Dataset (Alyasiri, 2018). Despite the increased use in evaluating other intrusion detection mechanisms, the dataset has major limitations for use in

evaluating a DDoS attack mechanism for SDN for containing many redundant records and lacks the most recent attack instances (Alrawashdeh and Purdy, 2016). The redundant records result in poor performance of a predictor by bringing bias towards the unfairly redundant records. The lack of most recent instances of attacks is a major setback in using this dataset for a DDoS attack detection mechanism as DDoS attack has become more sophisticated in recent times with attackers coming up with multiple attack vectors (Zhang et al., 2019). The NSL KDD is discussed in the following section.

## 2.8.2 NSL KDD

The NSL KDD dataset is an improved dataset from KDD Cup 99 that was created as a way of addressing the challenges of the earlier dataset in the year. The challenge of redundant records is addressed in this dataset while it maintains 39 attack types categorised into DoS, Probe, U2R, and R2L as in the KDD Cup 99. It has been used in designing a DDoS detection mechanism by, among other researchers, Osanaiye et al., (2016) and Yusof, Udzir, Selamat, Hamdan, and Abdullah, (2018). It contains notable DoS attack types as shown in Table 2.1. This dataset provides a viable option to evaluate the performance of a DDoS attack detection mechanism as it has been used by many researchers with available performance that can be used to validate current work. However, the fact that it was created over two decades ago limits its use as it does not represent the true characteristics and behaviour of network traffic flowing in modern networks and attacks (Gharib et al., 2017). Another inherent insufficiency is the labelling of the instances as either normal or anomaly. Using this dataset for evaluating a DDoS attack detection mechanism poses a challenge as we will need to use the entire dataset that will adversely impact computational time and resources which are also critical in having an efficient mechanism. It is also hard to determine the performance of the mechanism for a specific type of attack where in this study the interest was in DDoS attack. As an alternative to this dataset, researchers have used the DARPA DDoS 2009 dataset.

*Table 2.1: List of attack types in the KDD Dataset*

| <b>Attack Class</b> | <b>Attack Type</b> |
|---------------------|--------------------|
|---------------------|--------------------|

|              |                                                                                                                                                        |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>DoS</b>   | Back, Land, Neptune, Pod, Smurf, Teardrop, Apache2, Udpstorm, Processtable, Worm                                                                       |
| <b>Probe</b> | Ipsweep, Portsweep, Nmap, Satan, Mscan, Saint                                                                                                          |
| <b>R2L</b>   | Ftp_write, Xlock, Xsnoop, Snpmpguess, Snpmpgetattack, Httpptunnel, Sendmail, Named, Guess_password, Imap, Multihop, Phf, Spy, Warezcclient, Warezmater |
| <b>U2R</b>   | Buffer_overflow, Loadmodule, Perl, Rootkit, Sqlattack, Xterm, Ps                                                                                       |

### 2.8.3 DARPA DDoS 2009

DARPA DDoS 2009 dataset was created by the Colorado State University for evaluation of network intrusion detectors from a previous version that was generated in 1998. The 2009 dataset was created with synthesised traffic that emulated traffic between a /16 subnet (172.28.0.0/16) and the Internet and was generated for 10 days between 3<sup>rd</sup> and the 12<sup>th</sup> of November 2009 (Gharaibeh and Papadopoulos, 2009). The dataset contains synthetic HTTP, SMTP, and DNS background data traffic. Despite being synthetic it has many types of attacks instances, including the DDoS attack (Slay and Moustafa, 2015). It is a huge dataset from which DDoS attack data that is well labelled can be extracted. At the time of its generation, it contained current and realistic DDoS attack characteristics unlike any other dataset (Behal and Kumar, 2016). Several researchers such as Slay and Moustafa (2015), Gu, Li, Guo, and Wang (2019), and Kokila, Thamarai Selvi, and Govindarajan (2015) have used the dataset in their work in designing DDoS attack detection mechanisms. Until recently it has been used for evaluating a DDoS detection mechanism.

However, the inherent limitations found in the dataset include the imbalance in the DDoS attack types available. The DDoS attack is dominated by a SYN-Flood attack type targeting a single destination IP address 172.28.4.7 on the HTTP port and a few instances of other types of DDoS attack (Gharaibeh and Papadopoulos, 2009). This limitation results in a biased detection of the dominating attack type in the mechanism. Due to the period between the time of its generation and now, the attack types included in the dataset are not up-to-date. This is a limitation in using it for the

evaluation of a DDoS attack detection mechanism as modern attackers use new techniques and multiple attack vectors to launch a DDoS attack. In using synthetic traffic, the attack has attributes different from the attack in a real network. A laboratory is used to generate a dataset from synthetic traffic, sometimes allowing for the artificial injection of traffic or attack types to meet the desired intention, while a dataset generated from a real network is captured live in a realistic network condition which includes the operating system, applications, and platform (Behal and Kumar, 2016). Therefore, datasets from synthetic traffic have inherent limitations for evaluating the performance of mechanisms that can work in a real-world network as such mechanisms have been revealed to perform poorly when put in a real network with routers, firewalls, and operating systems (Gharib et al., 2017).

#### 2.8.4 CAIDA DDoS

The DDoS dataset from the Centre for Applied Internet Data and Analysis (CAIDA) hosted at the University of California, contains traffic captured for one hour on 4<sup>th</sup> August 2007. The traffic has been anonymised. The one hour split into five minutes traces pcap files with a total size of 5.3GB. TCP SYN flood, ICMP flood. HTTP requests comprised the DDoS attack, which used up all bandwidth on the network connecting the targeted server to the internet, interrupting access to the server. In addition, the non-attack traffic has been deleted as much as possible from the traces, leaving just records of attacks to the victim and responses from the victim ("The CAIDA DDoS Attack 2007 Dataset – CAIDA", 2020). Researchers such as Gu, Li, Guo, and Wang, (2019) and Balkanli, Alves, and Zincir-Heywood,(2014) have used a mechanism for DDoS attack detection. This dataset provides a close view of DDoS attack instances by focusing on the actual attack in its generation from real traffic and provides a viable option for use in designing a DDoS attack detection mechanism. The attacks in this dataset are low-rate DDoS attacks. According to Bhuyan, Bhattacharyya, and Kalita (2015), a high-rate DDoS attack occurs when the network receives more than 10,000 packets per second. Thus, 1000 attack packets per second covering 60% of the attack traffic is considered low-rate attack

traffic. This limits the use of this dataset as a benchmark for evaluating mechanisms for high-rate DDoS attack research. In the following section, we analyse the CICIDS2017 as an alternative dataset.

### 2.8.5 CICIDS2017

The Canadian Institute of Cybersecurity at the New Brunswick University in 2017 generated the CICIDS2017 dataset that covers the latest trends of threats and a variety of attack features. The data was captured for a period of five full days Monday to Friday, from 9 a.m. on 3<sup>rd</sup> July 2017 until 5 p.m. on 7<sup>th</sup> July 2017. The entire dataset is available in eight separate files. It is made of data captured from naturalistic benign background traffic that profiled 25 users based on the HTTP, HTTPS, FTP, SSH, and email protocols (“Intrusion Detection Evaluation Dataset -CIC-IDS2017”, 2018). It covers the latest trends of threats and a variety of attack features that are not covered by previous datasets such like Brute Force FTP, Brute Force SSH, XSS, SQL Injection, DoS, Heartbleed, Web Attack, Infiltration, DDoS, Port scan, and Botnet which represents a more real network data (“Intrusion Detection Evaluation Dataset -CIC-IDS2017”, 2018).

Since its creation, the CICIDS2017 dataset has attracted much interest from researchers designing intrusion detection systems including DDoS attack detection mechanisms (Alyasiri, 2018; Gu et al., 2019; Haider et al., 2019). This is because of its versatility which covers multiple attack types making it usable for both binary and multiclass detection mechanisms. Based on its latest attack instances, Abdulrahman and Ibrahim (2018) argues that it is a more suitable dataset for the evaluation of intrusion detection techniques for future researchers.

*Table 2.2: CICIDS2017 dataset files description*

| File Number | Day Activity         | Attacks found                                                                |
|-------------|----------------------|------------------------------------------------------------------------------|
|             | Monday               | Benign                                                                       |
|             | Tuesday              | Benign, FTP-Patator, SSH-Patator                                             |
|             | Wednesday            | Benign, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS Slowloris, Heartbleed |
|             | Thursday (Morning)   | Benign, Web Attack – Brute Force, Web Attack Sql Injection, Web Attack –XSS  |
|             | Thursday (Afternoon) | Benign, Infiltration                                                         |

|                    |                  |
|--------------------|------------------|
| Friday (Morning)   | Benign, Bot      |
| Friday (Afternoon) | Benign, PortScan |
| Friday (Afternoon) | Benign, DDoS     |

Table 2.2, shows all the attack information during the five days the dataset was generated. According to Panigrahi and Borah (2018), Monday is a normal day with only benign traffic with no attack recorded. Tuesday, Wednesday, and Thursday morning data have most of the attacks recorded. The three days captured data is best for designing a multiclass detection model. Thursday working hour afternoon and Friday data are well suited for binary classification. When all the eight files are combined the total size is 51.1Gigabites ("Intrusion Detection Evaluation Dataset -CIC-IDS2017", 2018). This combined dataset becomes a huge file that poses a challenge in the design of the IDS as it will consume a lot of computation resources in processing.

According to Gharib, Sharafaldin, Lashkari, and Ghorbani (2017), the dataset meets all the criteria, requirements, and standards for an intrusion detection dataset. It is generated from a complete network configuration that includes PCs, routers, and firewalls and from realistic pseudo-realistic traffic which is partially real-world traffic. The dataset is fully labelled with the entire traffic captured without removing part of it as done in other datasets. The dataset also contains a diversity of attacks including the latest DDoS attack scenarios and all traffic protocols recorded. Unlike other datasets, the CICIDS2017 has no anonymity as both the IP and Payload are available (Gharib et al., 2017). This dataset generated from a heterogeneous system can be used at any point in the detection process. The availability of feature sets makes the dataset usable by researchers to evaluate diverse detection mechanisms. Finally, the well recorded metadata found on this dataset is one of the most missing elements in many datasets.

Despite the gains achieved in this dataset, it still comes with some challenges which are well documented by researchers who have evaluated it. The dataset is scattered over eight separate files and becomes huge when combined. The combined dataset results in 3119345 instances and a file size of 51.1Gigabites. An evaluation by Panigrahi and Borah, (2018) details that the CICIDS 2017

contains 203 missing values and 83.34% instances of benign, presenting a majority of one class that leads to a high-class imbalance. A class imbalance in a dataset is when there is a majority of one or more instances. When such a dataset is used in its raw state, the class imbalance misleads the classifier and its performance is biased towards the majority class.

## 2.9 Analysis Outcome

The outcome of the analysis for the five commonly used datasets in DDoS attack detection presented in Table 2.3 shows that CICIDS2017 is a more appropriate dataset for evaluating a DDoS attack detection mechanism in Software Defined Network. Despite its shortcomings, the dataset remains appropriate for use, especially in evaluating DDoS detection mechanisms due to the gains and the available solutions research has unearthed in resolving the shortcomings. One of the most adopted approaches is the pre-processing of the dataset. This is a process that uses techniques to prepare a dataset to meet specific requirements so that it can be used for evaluating an intrusion detection mechanism.

*Table 2.3: Summary analysis of five commonly used Dataset for DDoS attack detection*

|                    |                                | <b>KDD CUP<br/>99</b> | <b>NSL KDD</b> | <b>CAIDA</b> | <b>DARPA</b> | <b>CICIDS2017</b> |
|--------------------|--------------------------------|-----------------------|----------------|--------------|--------------|-------------------|
| <b>1.</b>          | Complete Network Configuration | YES                   | YES            | YES          | YES          | YES               |
| <b>2.</b>          | Complete Traffic               | NO                    | NO             | YES          | NO           | NO                |
| <b>3.</b>          | Complete Capture               | YES                   | YES            | NO           | YES          | YES               |
| <b>4.</b>          | Heterogeneity                  | NO                    | NO             | NO           | NO           | YES               |
| <b>5.</b>          | Feature set                    | YES                   | YES            | NO           | NO           | YES               |
| <b>6.</b>          | Diverse DDoS attack Types      | YES                   | YES            | NO           | YES          | YES               |
| <b>7.</b>          | Labelled DDoS attacks          | NO                    | NO             | YES          | YES          | YES               |
| <b>8.</b>          | Available Protocols            | YES                   | YES            | YES          | YES          | YES               |
| <b>9.</b>          | Available Metadata             | YES                   | YES            | NO           | YES          | YES               |
| <b>Total Score</b> |                                | <b>6/9</b>            | <b>6/9</b>     | <b>4/9</b>   | <b>6/9</b>   | <b>8/9</b>        |

Table 2.3 presents a summary analysis of the five most used datasets based on the characteristics which are relevant for an evaluation dataset for DDoS attack detection. From this analysis, CICIDS2017 has more inherent advantages than the other four datasets. Based on this we

propose the use of CICIDS2017 for the evaluation of a DDoS attack detection mechanism. The dataset provides a better option for evaluating the DDoS attack mechanism in Software Defined Network due to its state-of-the-art approach of generating and containing the most up-to-date attack vectors. The inherent structure of having a DDoS attack data on a separate file captured Friday afternoon contains data labelled benign and DDoS attack which provides a focused approach to research on DDoS attack detection mechanisms and addresses the fundamental challenge of the curse of dimensionality that arises from using the entire dataset. Arun Raj Kumar and Selvakumar (2013) employed this approach by singling out the subset that only contains DDoS attack type from the entire CAIDA dataset. Using the CICIDS2017 DDoS dataset reduces the number of instances from 3119345 to 225745, which subsequently reduces the computational resources required to process the dataset.

## 2.10 CICIDS2017 DDoS Dataset

The CICIDS2017 was generated in a testbed that was configured to meet the requirement of a dataset framework by having a complete network configuration (Gharib et al., 2017). The configuration had two separate networks as the attack network and the victim's network. Each of these networks had a more realistic configuration by including both hardware and software that are used on a real network. The equipment included personal computers, servers, routers, switches, and a firewall on the victim side while the attack network had a router, a switch, and PCs. The configuration included different versions of mostly used operating systems namely Linux, MacOS, and Windows with a domain controller and an active directory set up on the victim network. Figure 2.4 shows a setup of the two networks with public and private IP addresses of the devices on the victim network and public IP on the attack network.

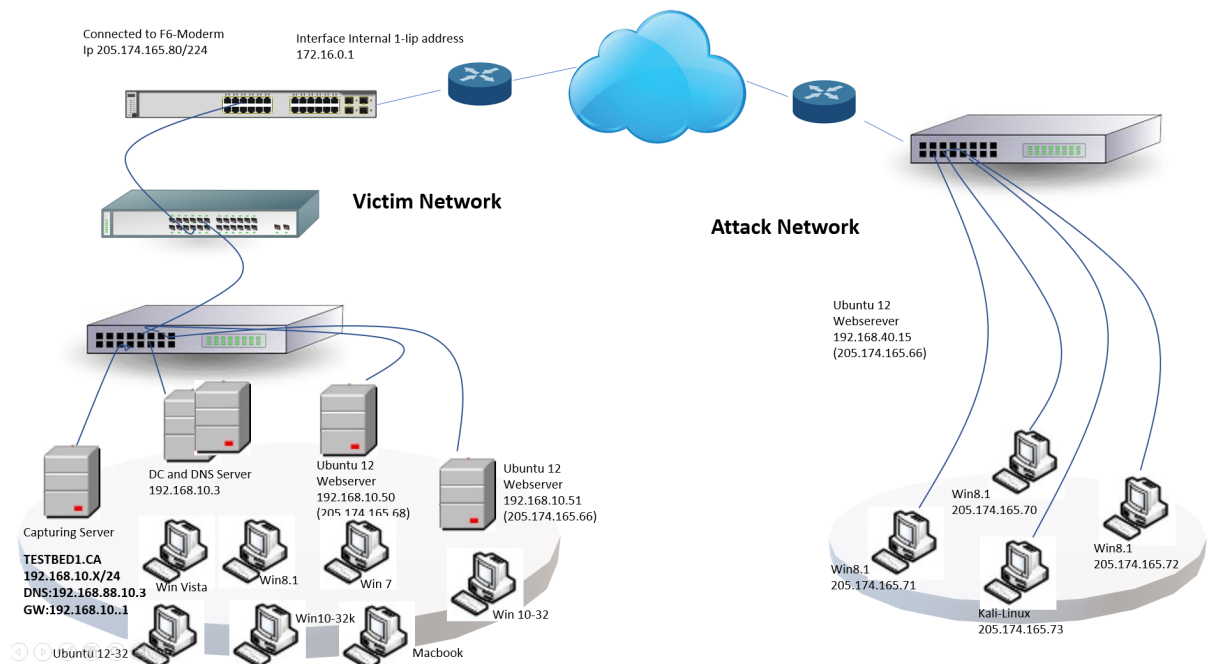


Figure 2.4: CICIDS2017 Test bed architecture. (Source: Sharafaldin et al., 2018)

The attack network used a DDoS attack tool namely the LOIC to launch the attack using all three PCs on the attack network sending UDP, TCP, or HTTP requests to the victim server on which a webserver is running Ubuntu 16.

Attackers: Three Win 8.1, 205.174.165.69 - 71

Victim: Ubuntu16, 205.174.165.68 (Local IP: 192.168.10.50)

NAT Process on Firewall:

Attackers: 205.174.165.69, 70, 71 -> 205.174.165.80 (IP Valid Firewall) -> 172.16.0.1

### Traffic Capturing

The traffic was captured using CICFlowMeter, an application that captures network traffic by generating bidirectional flows using the information presented in the first packet and also calculating

time related features such as flow duration and amount of idle time of a flow (Lashkari et al., 2017).

The captured traffic was converted and presented in the dataset as a CSV file.

## 2.11 Summary

Software Defined Network has changed the network architectural design from bundling all planes in one to a separate three planes design. It has also provided the opportunity to centralise network management and provision for scalability. In addition, it also comes with programmability that allows anyone to program how and what they want the network to operate. All these benefits come with their security challenges and the most significant is the centralised architecture of having the controller as the brain for the network. This centralised approach creates a single point of failure. With DDoS still, devastating networks, SDN faces this threat and there is a need to deal with it.

Reviewing past works has shown numerous efforts in the detection of DDoS attacks in networks with the use of Artificial Intelligence. Different techniques have been used to design intrusion detection mechanisms using AI such as Deep Learning, Reinforcement Learning, and recently, Deep Reinforcement Learning. An analysis of the DRL algorithms has shown that the Markov Decision Process can be used to learn and classify instances if deployed even in a network environment. The DRL would be improved by combining with other algorithms to create a hybrid DRL to be used as a detection mechanism for DDoS in SDN. To evaluate the performance of the HDRL detection mechanism, the use of a reliable dataset is important. Analysing available datasets, the CICIDS2017 dataset was the most suitable for evaluating a Hybrid Deep Reinforcement Learning detection mechanism.

## Chapter 3 : Methodology

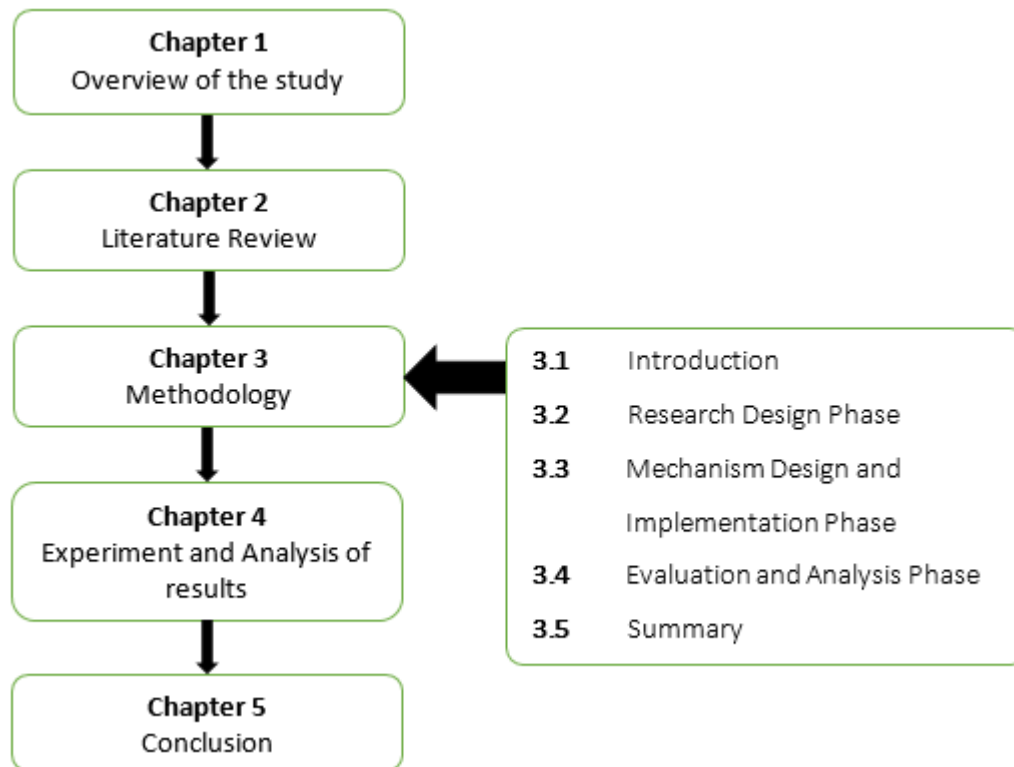
### 3.1 Introduction

This chapter presents the approach, methods, tools, and instruments employed by the study in the designing, implementation and evaluation of a Hybrid Deep Reinforcement Learning Mechanism (HDRLM) for DDoS attacks detection in Software Defined Network. This approach can be repeated and expected to yield similar results as produced by this study. The study's main objective was to design a hybrid mechanism to efficiently detect DDoS attacks in Software Defined Networks using Deep Reinforcement Learning. To achieve the objective, a systematic literature survey approach was first used to establish the best approach in the designing of the HDRLM, from which an experimental design was adopted and employed as the main approach. The literature survey was aimed at obtaining the best methods, tools, and strategies to design a detection mechanism to solve the research problem. The literature survey formed the basis on which the experiment was conducted to implement and evaluate the mechanism. By employing this approach, of combining literature survey and experiment (Peffer et al., 2007), the study benefited from the strength of each approach at the same time compensated for the shortcoming of each approach when used individually. The experimental design approach was adopted as it takes into consideration the time, resources, and skill set of the researcher at the same time it allows the researcher the greatest measure of control (Leavy,2017).

The approach is presented in three systematic and sequential phases used by the study as a structure as follows:

- **Research design phase**
- **Mechanism Design and Implementation phase**
- **Evaluation and analysis phase**

Section 3.2 presents the research methodology phase followed by the Mechanism design and Implementation phase in Section 3.3. The third and final phase of Evaluation and analysis and phase is presented in Section 3.5. The chapter outline is presented in the chapter map below.



### 3.2 Research Design Phase

The study used a design science research framework to design a HDRLM to detect DDoS attacks in Software Defined Networks. Design science is a model which focuses on producing artifacts, using a scientific approach, from ideas and innovations through analysis, design, implementation, management, and use to solve real business problems (Hevner et al., 2004). Peffers, Tuunanen, Rothenberger, and Chatterjee (2007) propose a six-step methodology for application for this research that aims to produce effective solutions to identified problems by creating artifacts.

Figure 3.1 shows how the processes undertaken by the study were aligned with the DSRFM and the methods used at each stage of the process. With the problem of the DDoS attack on SDN well defined, the focus of the study was on establishing the second stage, Definition of objective, an

appropriate entry point for the model as provided by the flexibility of the DSRM. The DSRM allows entry into the model at four different stages of the model based on the focus of the research (Peffer et al., 2007). Through literature, the research objectives as well as the methodology used by the study were defined. Literature in combination with experiment was used in the design and development of the artefact that was demonstrated in the implementation stage using the experiment. The same experiment was also used to evaluate the artefact. The outcome of the study was then communicated through a thesis.

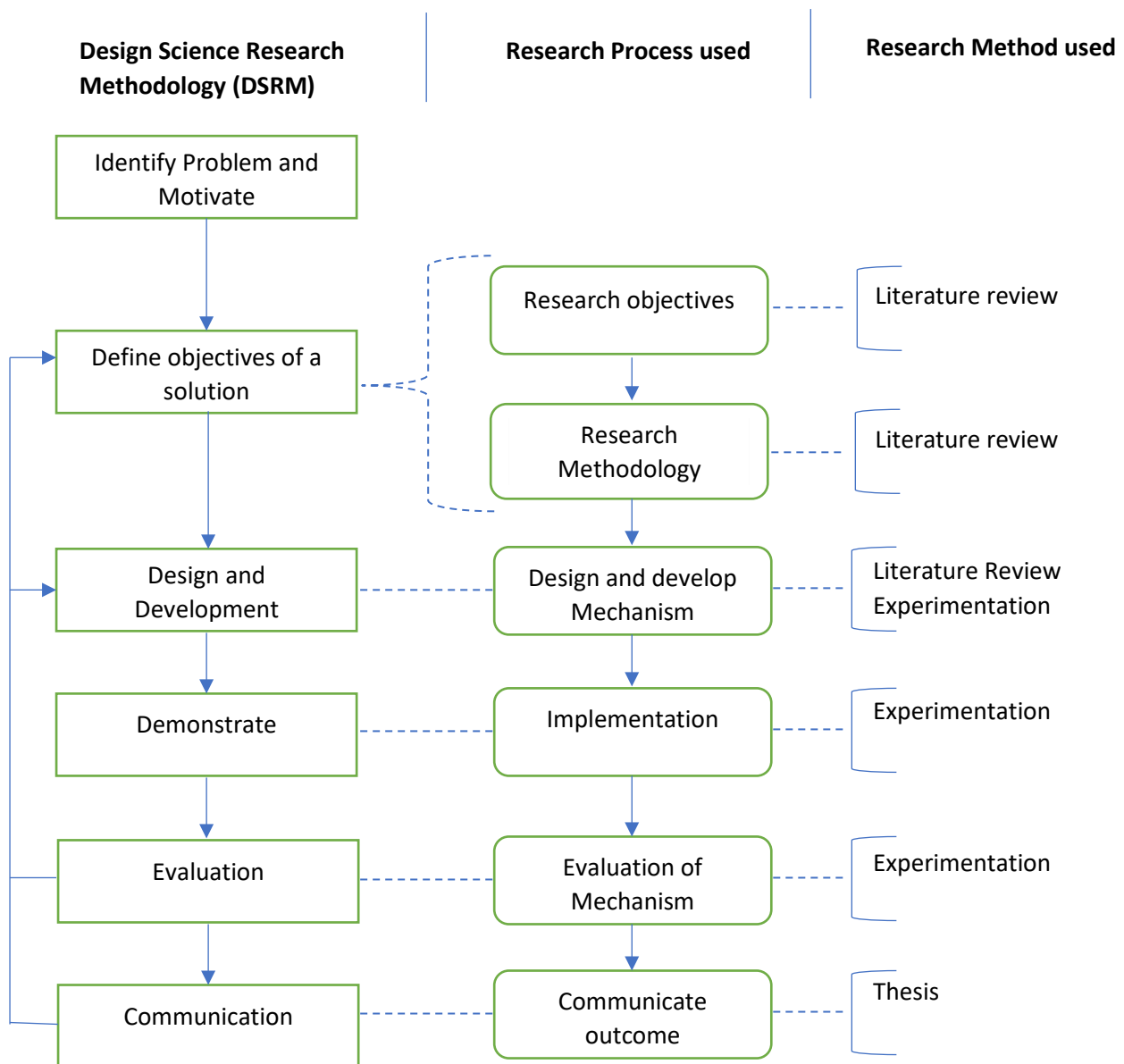


Figure 3.1: The Design Science Research methodology, Research process, and research methods.

A sequential triangulation of the qualitative-quantitative method, by combining a literature survey and an experiment was used (Clarke, 2005). The results from the literature survey are used to validate the results of the experiment. The literature survey was predominantly used as a method to achieve the first and second objectives, in identifying and evaluating a benchmark dataset for evaluating a DDoS attack mechanism for SDN, best feature selection technique, and features subsets, and in the second objective to determine efficient Deep Reinforcement Learning classifiers.

The third and main objective of designing the mechanism was achieved by the use of literature with an experimental approach as the main method. This experimental approach was employed to obtain a means to evaluate the quality of the design and measure the performance of the mechanism.

Section 3.2.1 explains how the literature survey was conducted, followed by the experimental design in Section 3.2.2.

### **3.2.1 Literature Survey**

The study used literature to understand techniques and approaches used in designing an intrusion detection system. The literature was collected based on criteria that were developed to obtain sufficient and relevant literature that added value to the study. The criteria considered the following: literature must be published within the past seven years; 2014 to 2022, results must be state of the art (between 85% to 99% in case of detection accuracy), number of citations must be more than 20, it must be authored by an expert in the area of study (if the paper is very recent and with less than 20 citations) and must be published in English.

Two specialised search engines, Google Scholar and Semantic Scholar, were used to search for the literature by using the following keywords: Intrusion detection, DDoS attack, Feature selection, Deep Reinforcement learning, Hybrid intrusion detection system. The literature was largely obtained from three databases namely: IEEE, Science Direct, and University Libraries. The literature that met the criteria was collated resulting in 166 sources that were further filtered by fitting them into three categories namely conference papers, peer-reviewed academic journals, and theses. Remaining at the end were 35 conference papers 50 peer-reviewed academic journals and eight theses that were used in the study.

### **3.2.2 Experimentation**

As empirically demonstrated by Yuan, Li, and Li (2017) and Barbhuiya, (2014) the use of an experiment in designing an anomaly detection mechanism is widely adopted as earlier argued by

Hevner et al. (2004) that experimental approach is one of the main methods used in design science research to produce quality solutions.

The design science research framework aims at producing an artefact that addresses critical business problems. The artefact is not only a concept through the activity of design and development, but it also seeks to demonstrate the utility of the proposed solution. To demonstrate the utility of an artefact, Peffers et al., (2007) argue for the use of experimentation as a method to demonstrate the designed artefact. The same experimentation is also used to evaluate if the artefact addresses the problem by achieving the objectives defined by the research.

Using the experimental approach, researchers have undertaken common steps in the process of designing anomaly detection mechanisms. However, there is no agreed sequence of the steps. Therefore, this study designed the experiment using the steps in a sequence that would present a credible experiment and best fits this research as follows:

- Dataset identification
- Pre-processing of dataset
- Feature selection
- Classifier development
- Training and testing of the classifier
- Mechanism evaluation

An experimental design, in Figure 3.2 represents the sequence of the steps as were carried out in the experimentation as each of the steps are further explained in detail as to how it works, why it is important in the process, and how it was used to achieve the objectives of this study.

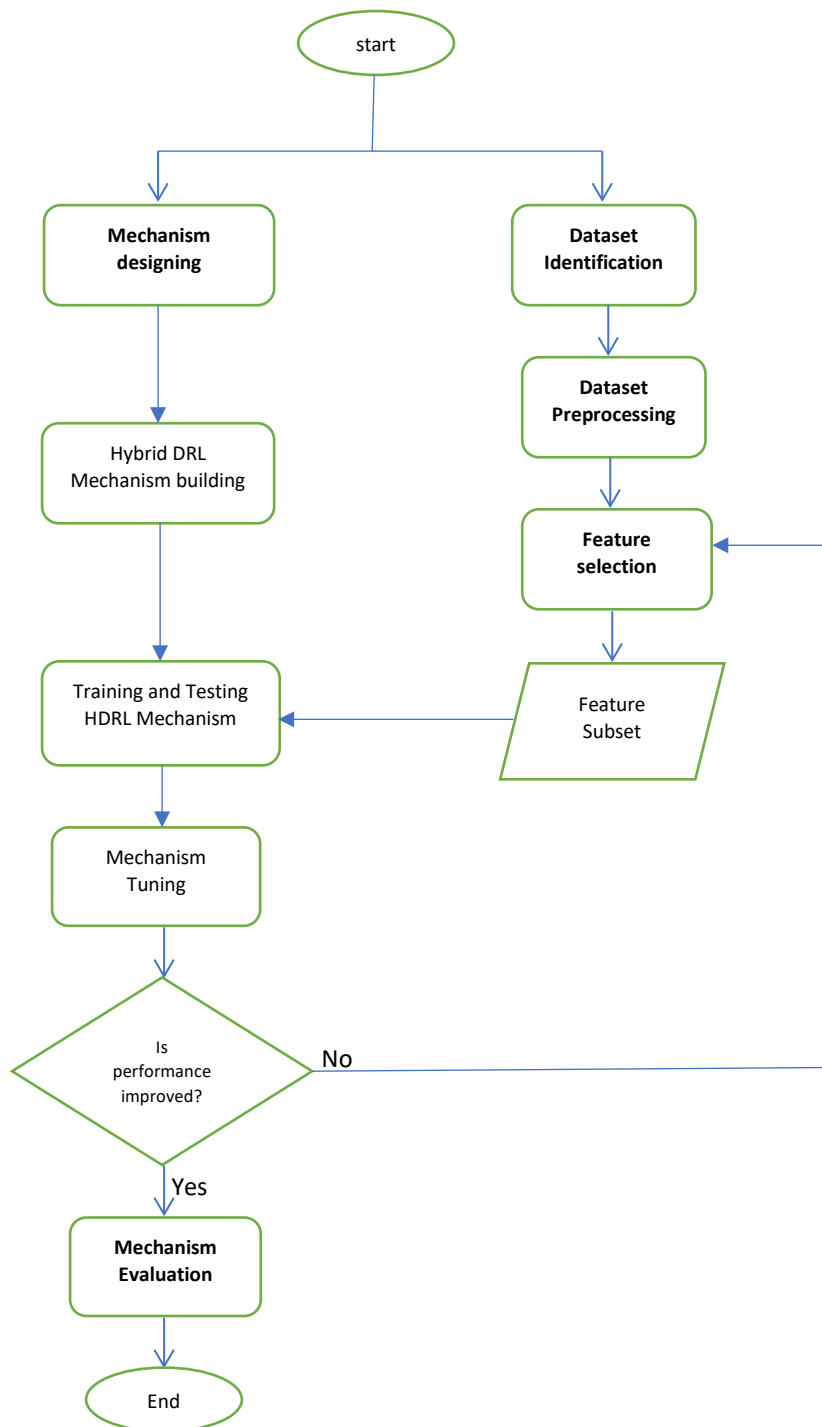


Figure 3.2: DDoS attack Mechanism experimental design.

### 3.2.3 Dataset Identification.

In designing a DDoS attack detection mechanism the use of a dataset, a collection of network traffic available offline, in place of live network traffic to evaluate the mechanism is a more adopted approach (Sharafaldin et al., 2017). Evaluation of proposed mechanisms on real network

traffic remains a challenge due to network owners not wanting to compromise their network through exposure to cyber threats and to protect network users' privacy which also borders on ethical issues (Gharaibeh and Papadopoulos, 2009).

Since the concept of using benchmark datasets in place of real network traffic, from 1998 until now, eleven datasets have been made available (according to Abdulraheem and Ibraheem (2019)) which researchers have used in evaluating proposed intrusion detection mechanisms. However, selecting a benchmark dataset to use in the evaluation of a DDoS attack detection mechanism is critical because a dataset affects the performance of the mechanism (Behal and Kumar, 2016). To address this, an analysis was done of the five most-used datasets, KDD CUP 99, KDD NSL, CAIDA, DARPA, and CICIDS2017 using a framework that takes into account eight inherent characteristics of a dataset (Gharib et al., 2017). This was conducted to determine and identify an appropriate dataset for evaluation of a DDoS attack detection mechanism in Software Defined Networks. The analysis is discussed in the proceeding chapter in which the CICIDS2017 dataset was identified to be appropriate for the study meeting seven of the eight criteria that were considered. The CICIDS2017 dataset has a range of attacks from traffic captured in five days, Monday to Friday, and is available on eight separate files in CSV format. The file containing traffic captured Friday afternoon has DDoS attacks and normal traffic only (<https://www.unb.ca/cic/datasets/ids-2017.html>). DDoS attack detection is the focus of this study. Therefore, from the CICIDS2017 dataset, only the Friday afternoon file with DDoS attack was used. The dataset was pre-processed to prepare for use in the experiment.

### 3.2.4 The CICIDS2017 Dataset Features

In its original format, the dataset contains more than 80 features including nominal and statistical features. For evaluating a mechanism, only statistical or numerical features are used. Table 3.1 lists the statistical features and their definitions with the features sorted alphabetically.

Table 3.1: CICIDS2017 Dataset Features

| Feature Label           | Feature description                                                                              |
|-------------------------|--------------------------------------------------------------------------------------------------|
| ACK Flag Count          | sum/nth of the acknowledge/established connections.                                              |
| Act_data_pkt_fwd        | Count of packets with at least 1 byte of TCP data payload in the forward direction               |
| Active Max              | Maximum time a flow was active before becoming idle                                              |
| Active Min              | Mean time a flow was active before becoming idle                                                 |
| Active Std              | Standard deviation time a flow was active before becoming idle                                   |
| Average Packet Size     | Average size of packet                                                                           |
| Avg Bwd Segment Size    | Average size observed in the backward direction                                                  |
| Avg Fwd Segment Size    | Average size observed in the forward direction                                                   |
| Bwd Avg Bytes/Bulk      | Average number of bytes bulk rate in the backward direction                                      |
| Bwd Avg Packets/Bulk    | Average number of packets bulk rate in the backward direction                                    |
| Bwd Header Length       | headers length in the backward direction                                                         |
| Bwd IAT Max             | Maximum time between two packets sent in the backward direction                                  |
| Bwd IAT Mean            | Mean time between two packets sent in the backward direction                                     |
| wd IAT Min              | Minimum time between two packets sent in the backward direction                                  |
| Bwd IAT Std             | Standard deviation time between two packets sent in the backward direction                       |
| Bwd Packet Length Mean  | Mean length of a flow in backward direction                                                      |
| Bwd Packet Length Min   | Minimum size of packet in backward direction                                                     |
| Bwd Packet Length Std   | Standard deviation length of a flow                                                              |
| Bwd Packets/s           | Number of backward packets per second                                                            |
| Bwd PSH Flags           | Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP) |
| Bwd URG Flags           | Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP) |
| CWE Flag Count          | Number of packets with CWE                                                                       |
| Destination Port        | Intended final arrival point of a data packet                                                    |
| Down/Up Ratio           | Download and upload ratio                                                                        |
| ECE Flag Count          | Number of packets with ECE                                                                       |
| Flow Duration           | Duration of the flow in microseconds                                                             |
| Flow IAT Max            | Maximum inter-arrival time of packet                                                             |
| Flow IAT Mean           | Mean inter-arrival time of packet                                                                |
| Flow IAT Min            | Minimum time of packet                                                                           |
| Flow IAT Std            | Standard depreciation time of packet                                                             |
| Flow Packets/s          | Number of flow packets per second                                                                |
| Fwd Avg Bulk Rate       | Average number of bulk rates in the forward direction                                            |
| Fwd Avg Packets/Bulk    | Average number of packets bulk rate in the forward direction                                     |
| Fwd Header Length       | The length of a header moving in a forward direction                                             |
| Fwd Header Length       | The length of a header moving in a forward direction                                             |
| Fwd IAT Max             | Maximum inter-arrival time of packet in forward direction                                        |
| Fwd IAT Mean            | Mean inter-arrival time of packet in forward direction                                           |
| Fwd IAT Min             | Minimum inter-arrival time of packet in forward direction                                        |
| Fwd IAT Std             | Maximum inter-arrival time of packet in forward direction                                        |
| Fwd Packet Length Max   | Standard deviation inter-arrival time of packet in forward direction                             |
| Fwd Packet Length Mean  | Mean length of a flow in forward direction                                                       |
| Fwd Packet Length Min   | Minimum length of a flow in forward direction                                                    |
| Fwd Packet Length Std   | Standard deviation length of a flow in forward direction                                         |
| Fwd URG Flags           | Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)  |
| Idle Max                | Maximum time a flow was idle before becoming active                                              |
| Idle Min                | Minimum time a flow was idle before becoming active                                              |
| Idle Std                | Standard deviation time a flow was idle before becoming active                                   |
| Init_Win_bytes_backward | The total number of bytes send in an initial window in the backward direction                    |
| Label                   | Type of flow (either benign or attack)                                                           |
| Max Packet Length       | Maximum length of a packet                                                                       |
| Min Packet Length       | Minimum length of a packet                                                                       |
| Min_seg_size_forward    | Minimum segment size observed in a forward direction                                             |
| Packet Length Mean      | Mean length of a packet                                                                          |
| Packet Length Std       | Standard length of a packet                                                                      |
| Packet Length Variance  | Variance length of a packet                                                                      |

| Feature Label               | Feature description                                                                             |
|-----------------------------|-------------------------------------------------------------------------------------------------|
| PSH Flag Count              | Number of packets with a PUSH                                                                   |
| RST Flag Count              | Number of packets with RST                                                                      |
| Subflow Bwd Bytes           | The average number of bytes in a subflow in the backward direction                              |
| Subflow Bwd Packets         | The average number of packets in a subflow in the backward direction                            |
| Subflow Fwd Bytes           | The average number of bytes in a subflow in the forward direction                               |
| SYN Flag Count              | Number of packets with SYN                                                                      |
| Total Backward Packets      | Total size of packet in backward direction                                                      |
| Total Fwd Packets           | Total size of packet in forward direction                                                       |
| Total Length of Bwd Packets | Total length size of packet in backward direction                                               |
| URG Flag Count              | Number of packets with URG flag                                                                 |
| Active Mean                 | Mean time a flow was active before becoming idle                                                |
| Bwd Avg Bulk Rate           | Average number of bulk rates in the backward direction                                          |
| Bwd IAT Total               | Total time between the two packets send in backward direction                                   |
| Bwd Packet Length Max       | Maximum length of a flow in backward direction.                                                 |
| FIN Flag Count              | Number of packets with FIN flags                                                                |
| Flow Bytes/s                | Number of flow bytes per second                                                                 |
| Fwd Avg Bytes/Bulk          | Average number of bytes bulk rate in the forward direction                                      |
| Fwd IAT Total               | Total time between the two packets send in forward direction                                    |
| Fwd Packets/s               | Number of packets per second                                                                    |
| Fwd PSH Flags               | Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP) |
| Idle Mean                   | Mean time a flow was idle before becoming active                                                |
| Init_Win_bytes_forward      | The total number of bytes send in an initial window in the forward direction                    |
| Subflow Fwd Packets         | The average number of packets in a subflow in the forward direction                             |
| Total Length of Fwd Packets | Total length of packet in forward direction                                                     |

### 3.2.5 Instance Distribution

There are a total of 225745 instances captured and available in the dataset. The instances are labelled as either Benign or DDoS. As shown in Figure 3.3, out of the instances, 97718 are benign which represents 43% and 128027 instances are labelled as DDoS attacks representing 57%. This is a fair distribution of the two classes that overcomes class imbalance that eventually reduces bias in the classification algorithm.

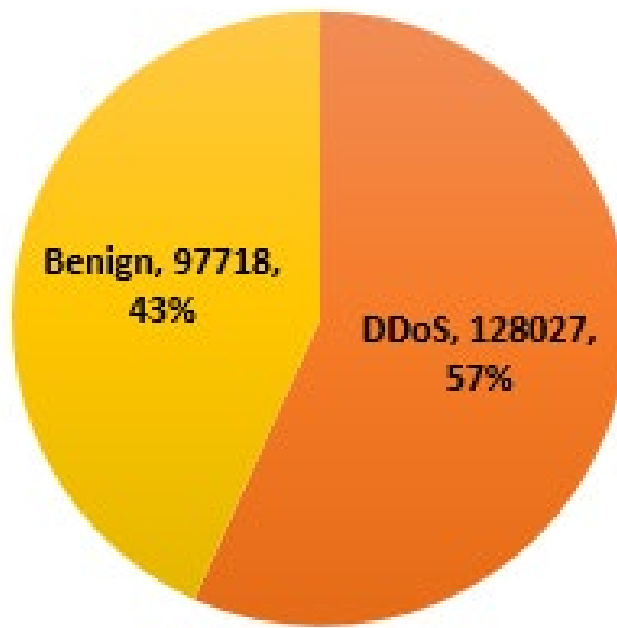


*Figure 3.3: CICIDS2017 instance distribution*

Like most datasets, the CICIDS2017 cannot be used in its raw state to evaluate a DDoS attack detection mechanism. The dataset requires to be cleaned and pre-processed to remove unwanted features or features that are not relevant and do not impact the performance of a mechanism.

### 3.2.6 Dataset Pre-processing

This is a process used to prepare the dataset by removing any unwanted or irrelevant elements as well as computing some attributes to obtain an appropriate form for the task. A dataset has inherent characteristics that if used in its raw state a detection mechanism might not present a realistic rate of performance. A major challenge with the dataset is the high number of features and instances that if used entirely, takes a long time to train the classifier and consume more computational resources. This challenge is overcome by this study by using the file with a DDoS attack and Benign only. During this process, all the missing data were removed (Panigrahi and Borah, 2018).

Another challenge with the dataset is that attributes in the dataset are made of values in different data types, some being integers and others float with different units of measurement. This variation necessitated transforming the values into a standardised representation. Several techniques are used to perform this standardisation including the min-max Z-score and decimal scaling normalisation (Abdulrahman and Ibrahim, 2019). The Z-score normalisation was used, and the values for the attributes were represented by the standard deviation using the Z-score formulae.

$$Z = \frac{X - \mu}{\sigma}$$

In which  $Z$  is the standard score,  $X$  is the observed value and  $\mu$  is the mean of all values and  $\sigma$  is the standard deviation of all values. At the end of this process, the dataset was clean for feature selection to be performed.

### 3.2.7 Dataset Cleaning

The dataset was first cleaned by undertaking the six steps. The first step checked for redundancies for both features and instances. The feature named “Fwd\_Header\_Length” was discovered to be in duplicate and was therefore removed. Additionally, 109 redundant instances were subsequently removed. The second step involved checking all missing values. These are values that might have been omitted or captured erroneously. All missing values were replaced with a zero value. In the third step, all the nominal features were removed. The nominal features are attributes that contain non-numeric values and are included in the dataset for information purposes. The dataset contains five nominal features which are FlowID, SourceIP, DestinationIP, External IP, Timestamp. These were manually removed. The fourth and final step was to identify instances that contained infinity values that were replaced by the mean of the feature value. At the end of these cleaning exercises, 79 features remained on the dataset and the dataset was clean and ready for feature selection.

### 3.2.8 Feature Selection

Feature selection is a process of selecting the only features from a dataset that are relevant to the task. This is performed by applying unique evaluation standards to improve the performance of a classifier while limiting the data's dimensionality by selecting a subset of the relevant characteristic from a vast number of fundamental attributes of a given dataset (Singh et al., 2019). This approach has been employed in different research disciplines for classification such as image recognition (Ewees and Eleraky, 2019) medical imaging (Han and Jiao, 2017), Fraud detection (Lima and Pereira, 2016), and DDoS attack detection (Gu et al., 2019).

This was one of the major components in this study that was used to achieve the first objective. The technique to select features subset and the features selected has a direct impact on the efficiency and accuracy of a mechanism. As Sahu, Sarangi, and Jena (2014) stated, feature selection has proven its ability to reduce computation demands, overfitting, model size and increase the accuracy.

An experiment together with a literature survey as presented in Figure 3.4 , were used to obtain five feature subsets. The literature survey was also used to inform the technique used in the experiment. A sample was conveniently identified from the literature collection of research where the CICIDS2017 dataset was used to evaluate a DDoS attack detection mechanism and the feature subsets used by the sampled literature were further analysed and selected based on their performance.

A feature selection process on the dataset obtained five feature subsets, one feature subset was obtained through the experiment and the other five were obtained through literature. The five feature subsets were further evaluated in an experiment to identify the most effective and efficient feature subset. The results are presented in [Section 4.7](#). The identified feature subset was used in the experiment to evaluate the performance of the mechanism.

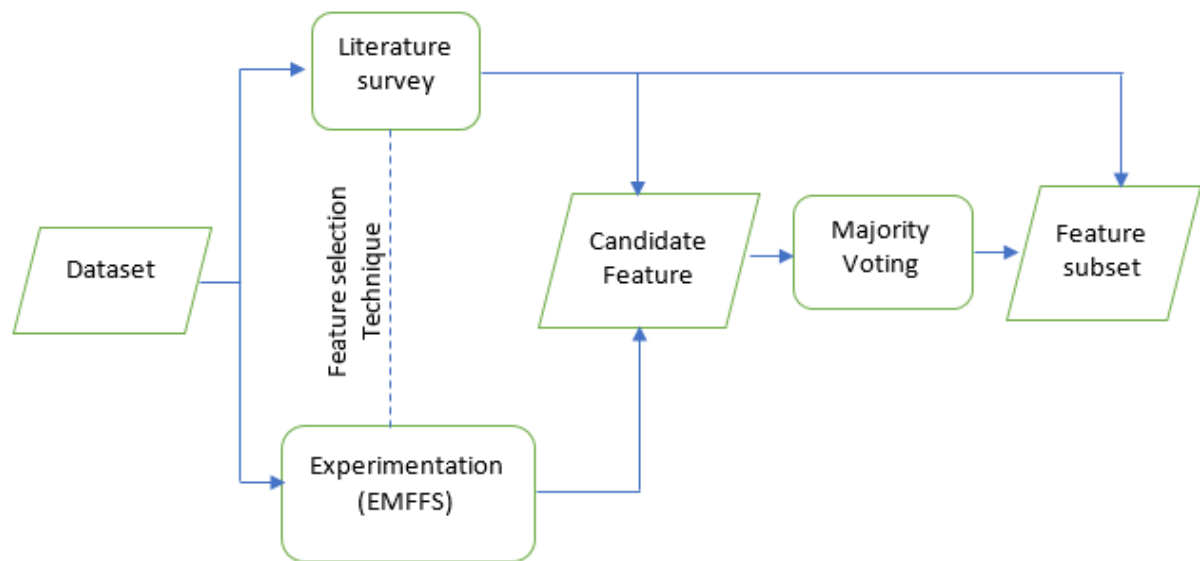


Figure 3.4: Feature selection Workflow

The fifth feature subset from literature was obtained by combining all the features from the sampled literature to come up with a candidate feature set to which a simple majority vote was applied to reduce the number of features to a small number in the final feature subset. The number of features was reduced to overcome challenges that arise in the classification such as overfitting. As Hamed, Dara, and Kremer (2018) state, reducing the number of features is an effective approach to overcome the challenge of overfitting and have relevant features. The voting methods such as simple Majority Voting, Weighted Majority Voting are useful when the number of features exceeds an appropriate threshold (Arun Raj Kumar and Selvakumar, 2013).

The first subset was obtained from the practical experiment in which an ensemble filter-based approach (EMFFS) was applied by combining four techniques namely: Information Gain, Chi-Square, Gain Ratio, and ReliefF (Osanaïye et al., 2016). This approach was employed, in contrast to using a single technique, so to gain the most from the techniques as combining them they collectively compensate for the shortfall of each other if used individually. Using the WEKA tool, the experiment was performed on the CICIDS2017 DDoS dataset and obtained a candidate dataset for each technique. A simple majority vote on the features was performed and obtained a feature subset.

In using an experimental approach, different studies have used techniques from the three feature selection methods independently, with recent studies having used new approaches through either an ensemble approach by combining different techniques from one method (Osanaïye et al., 2016) or a hybrid approach that combines one or more techniques from different methods (Gu et al., 2019). The feature selection approach has been revealed to have an impact on the performance of a mechanism as combining more than one technique improves the performance of a mechanism (Osanaïye et al., 2016).

This study used the approach that combines feature selection techniques and selected the one that achieves high accuracy and efficiency from those discussed in Section 3.2.2. The Ensemble-based Multi-filter Feature Selection (EMFFS) method (presented in Figure 3.5) which combines four techniques under the filter method namely, Information Gain, Chi-Square, Gain Ratio, and ReliefF (Osanaïye et al., 2016), was applied for feature selection on the CICIDS2017 dataset.

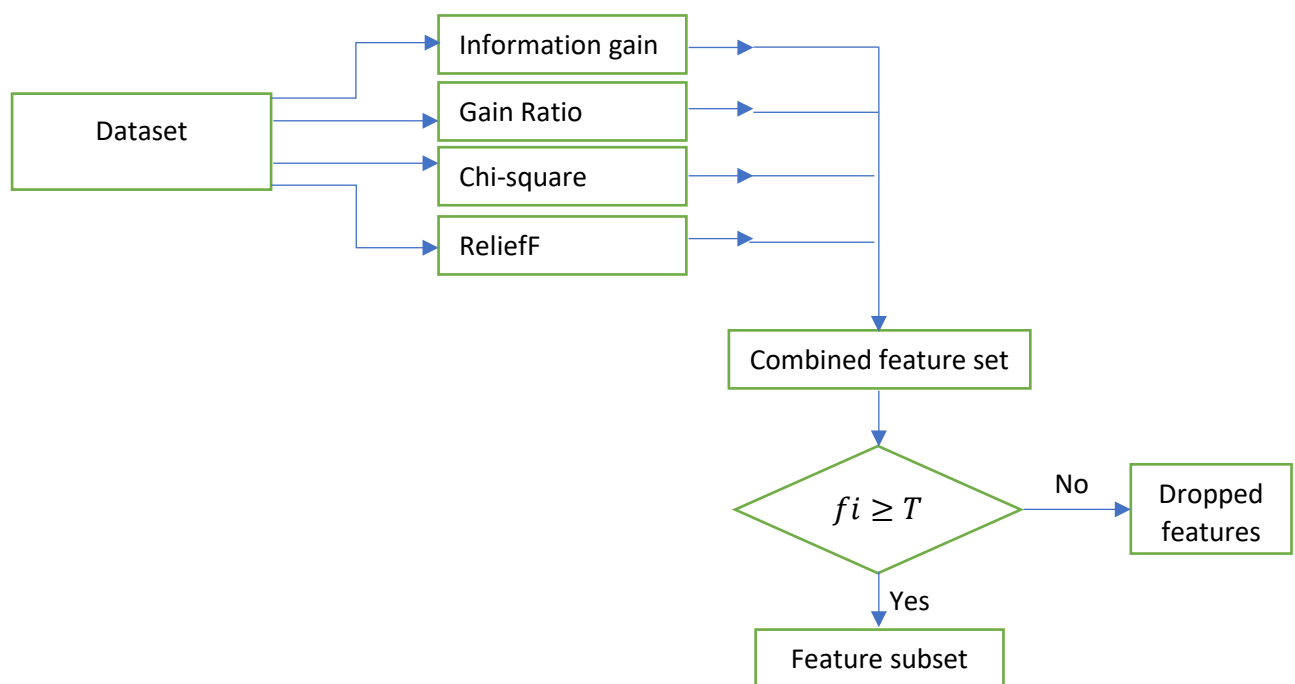


Figure 3.5: The Ensemble-based Multi-filter Feature Selection (EMFFS) method (Source: Osanaïye et al., 2016)

In the application of the EMFFS, each technique was applied to the dataset individually and obtained a candidate feature set represented in Table 3.2.

Table 3.2: EMFFS candidate feature set

| IG          | 5  | 63 | 53 | 6  | 65 | 55 | 13 | 35 | 1  | 11 | 66 | 54 | 9  | 7  |
|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Gain ratio  | 1  | 7  | 65 | 6  | 12 | 54 | 9  | 49 | 55 | 13 | 67 | 66 | 63 | 5  |
| Chi-squared | 63 | 5  | 53 | 1  | 35 | 65 | 6  | 55 | 13 | 11 | 66 | 36 | 9  | 54 |
| ReliefF     | 5  | 65 | 6  | 55 | 11 | 13 | 63 | 22 | 8  | 54 | 14 | 31 | 33 | 48 |

Thereafter, a simple majority vote was performed on the combined features from the subsets using frequency and ranking, from which the best six features were obtained. The features 5,6,13,55,63,65 presented in Table 3.3, emerged to be the best from the experiment using the EMFFS approach.

Table 3.3: Final feature subset

| Feature number | Feature name                |
|----------------|-----------------------------|
| 5              | Total Length of Fwd Packets |
| 6              | Total Length of Bwd Packets |
| 13             | Bwd Packets Length Mean     |
| 55             | Avg Bwd Segment Size        |
| 63             | Subflow Fwd Bytes           |
| 65             | Subflow Bwd Bytes           |

The three feature subsets from the literature were identified through their use in previous research work that used the CICIDS2017 dataset for evaluating DDoS detection mechanism and had produced satisfactory results of not less than 95% detection accuracy and the fifth set was drawn by putting together all the four feature subsets and applying majority vote that resulted in common features from the four feature subsets.

Table 3.4 shows the four feature subsets sampled from the literature with the technique that was applied by the researchers to come up with the features and the last column indicates the number of features in the subset.

Table 3.4: Details of feature subsets from literature

| Source            | Dataset             | Method and Technique       | Candidate Feature                 | Number of Features |
|-------------------|---------------------|----------------------------|-----------------------------------|--------------------|
| (Gu et al., 2019) | DARPA DDoS dataset, | Hybrid Filter and embedded | Destination port, Destination IP, | 2                  |

|                                 |                                                            |                                                            |                                                                                                                                                                               |    |
|---------------------------------|------------------------------------------------------------|------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
|                                 | CAIDA “DDoS attack 2007” dataset, CICIDS “DDoS attack 2017 | (Entropy and Sequential Forward Selection (SFS) algorithm) |                                                                                                                                                                               |    |
| (Abdulrahman and Ibrahim, 2019) | CICIDS2017                                                 | Filter (Information Gain)                                  | Fwd.IAT.Total, Flow.IAT.Max, Active.Max, Active.Min, Fwd.IAT.Max, Active.Mean, Idle.Min, Idle.Max, Flow.IAT.Std, Fwd.IAT.Mean                                                 | 10 |
| (Sharafaldin et al., 2018)      | CICIDS2017                                                 | Embedded (Random Forest regression)                        | Bwd packet length Std, Avg. packet size, flow duration, Flow IAT Std                                                                                                          | 4  |
| (Singh et al., 2019)            | CICIDS2017                                                 | Embedded (J48)                                             | Destination Port, Total Length of Fwd Packets, Total Length of Bwd Packets, Fwd packets Length Min, ACK flag Count, Init_Win_bytes_forward, Init_Win_bytes_backward, Idle Min | 8  |

The study considered three feature subsets by Abdulrahman and Ibrahim, (2019), Sharafaldin et al., (2018) and Singh et al., (2019). Feature subsets by Gu et al., (2019) was not considered due to its size of only two features with the other feature being a nominal feature. The fourth feature subset was, therefore, obtained by combining the features presented in Table 3.5 to obtain a candidate feature set.

*Table 3.5: Combined feature subsets from literature*

|                                                                                                                                                                                                                                                                                                                        |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Combined Features                                                                                                                                                                                                                                                                                                      |
| <b>Source ip, Destination port, Destination ip, Fwd.IAT.Total, Flow.IAT.Max, Active.Max, Active.Min, Fwd.IAT.Max, Active.Mean, Idle.Min, Idle.Max, Flow.IAT.Std, Fwd.IAT.Mean, Fwd.packetlengthStd, Avg.packetsize, Flowduration, Flow.IAT.Std, Destination Port, TotalLengthofFwdPackets, TotalLengthofBwdPackets</b> |

Thereafter the majority vote was applied to the candidate feature subset and resulted in three features shown in Table 3.6 as the common features appearing in all the studies. The three were, therefore, considered in the evaluation of the mechanism in this study.

Table 3.6: Feature subset after majority voting

|                                                 |
|-------------------------------------------------|
| Final feature set                               |
| <b>Destination port, Flow IAT Std, Idle Min</b> |

Table 3.7: Feature subsets from Literature and Experimentation

|    | Feature subset   | Features                                                                                                                                                                      |
|----|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. | Feature subset 1 | Total Length of Fwd Packets, Total Length of Bwd Packets, Bwd Packet Length Mean, Avg Bwd Segment Size, Subflow Fwd Bytes, Subflow Bwd Bytes                                  |
| 2. | Feature subset 2 | Fwd.IAT.Total, Flow.IAT.Max, Active.Max, Active.Min, Fwd.IAT.Max, Active.Mean, Idle.Min, Idle.Max, Flow.IAT.Std, Fwd.IAT.Mean                                                 |
| 3. | Feature subset 3 | Bwd packet length Std, Avg. packet size, flow duration, Flow IAT Std                                                                                                          |
| 4. | Feature subset 4 | Destination Port, Total Length of Fwd Packets, Total Length of Bwd Packets, Fwd packets Length Min, ACK flag Count, Init_Win_bytes_forward, Init_Win_bytes_backward, Idle Min |
| 5. | Feature subset 5 | Destination port, Flow IAT Std, Idle Min                                                                                                                                      |

Table 3.7 presents the five feature subsets obtained from literature and experimentation.

They were further subjected to an evaluation on different permutations of the HDRLM to identify the best feature subset. Feature selection technique is presented in Chapter 4 Section 4.7. The best feature subset was used to evaluate the HDRLM.

### 3.2.9 Mechanism Development

The basic and main functionality of an anomaly detection mechanism is to classify an attack from normal traffic. This classification is performed by classifiers that are algorithms that use logic and mathematical computation to read the attributes of traffic to detect if it is an attack or it is normal traffic. Therefore, a classifier was one of the major components of this study on which the second objective was to be achieved that would eventually produce an efficient DDoS attack detection mechanism.

To build classifiers that will use less computational resources while achieving high accuracy in detecting DDoS attacks in Software Defined Networks, the study built a hybrid mechanism using Deep Reinforcement Learning. According to Mazini, Shirazi, and Mahdavi (2018), the accuracy and detection rate of a hybrid IDS improves in comparison with legendary methods of using a single agent approach.

A literature review was first conducted on the design and evaluation of Deep Learning algorithms in DDoS attack detection and how Reinforcement Learning algorithms have been used in developing a detection mechanism. Further review was also done on how the two algorithms can be combined was. The techniques from the literature were employed in an experiment to build the Hybrid Deep Reinforcement Learning classifier to detect DDoS attack detection in SDN.

### **3.2.10 Mechanism Training and Testing**

For the mechanism to perform detection, it must learn the patterns of what is regarded as normal so that any diversion from that is regarded as an attack. The learning is achieved by first feeding the mechanism with training data sampled from the data that the mechanism is expected to perform the detection (Kokila et al., 2015). Once training is completed, another set of data is fed into the mechanism to test if it has learned and can detect whether traffic is normal or an attack (Kokila et al., 2015). Using the dataset obtained from the stage of dataset identification the study trained and tested the mechanism. The dataset was first split into train data and test data with a 40/60 split, using 40% to train the classifier and 60% to test the mechanism. The performance of the mechanism at this step is what is used to evaluate it.

### **3.2.11 Mechanism Evaluation**

Evaluation is a crucial element in the design and development of a detection mechanism. It determines if the machine does what it is meant to do. This is done by comparing metrics to determine if the mechanism solves the problem it is designed to solve (Peffer et al., 2007). The details on how the mechanism was evaluated are discussed in Section 4.9.

## **3.3 Mechanism Design and Implementation Phase**

This section details the design of the mechanism and the approach used in the implementation of the design. Section 3.7.8 3.2.1 lists the tools and instruments used, following the Hybrid Deep Reinforcement agent explained in Section 3.2.2, followed by the implementation in Section 3.2.3.

### 3.3.1 Tools and Instruments.

To build the environment, the model, and test the model, several tools were used and put together such as installing and configuring the environment for the experiment and development of the algorithms. From the many tools available to implement this study, the following were the tools selected:

- **Weka:** A software that is built to perform data mining tasks that include data pre-processing, attribute selection (Srivastava, 2014). The WEKA tool was used for the pre-processing of the dataset; data cleaning, normalisation., and feature selection.
- **Tensorflow:** A Machine learning based a tool that allows researchers to experiment with novel optimisations, algorithms and supports the training of deep neural networks and other applications. (Abadi, et al. 2016). The study made use of this capability to build and train hidden layers for computation in the model.
- **Keras:** This is a powerful library that is predominantly used to build deep learning Networks using abstract building blocks (Ketkar, 2017). It has an API that easily supports Tensorflow and is very useful in the Reinforcement Learning algorithm.
- **Gym:** A toolkit that is specifically developed for Reinforcement Learning (Brockman et al., 2016). It is made up of Reinforcement Learning libraries that support the development of a Reinforcement Learning model and the training of the agent.
- **Jupyter notebook:** This is an interactive web-based code editor that provides a development environment for live code and visualisation. This tool was used to develop the code and run the simulation, modelling, and data visualisation.
- **Scikit-learn:** Machine learning module, an effective tool for data mining, and data analysis. It was used for analysing the data.

- **Python:** It is a high-level, general-purpose programming language that can do structured or procedural programming, object-oriented programming, and functional programming. Python was used to develop the model (“Python - Overview of the Programming Language”, 2019)

### 3.3.2 Hybrid Deep Reinforcement Learning Mechanism Design.

The detection mechanism proposed in this study is a HDRLM that was built by using the Deep Reinforcement Learning algorithm with a combination of two heterogeneous neural networks. An evaluation of deep learning and reinforcement learning was undertaken with the criteria of efficiency and accuracy as determinants for the choice algorithms. The Deep Q Network (DQN) algorithm was identified and employed as it meets the requirement of a reinforcement algorithm and a deep learning algorithm combined. The DQN algorithm is made up of a Q-learning algorithm which is a classical Reinforcement Learning algorithm and a Deep Neural Network a Deep Learning Algorithm (Fan et al., 2019). Hu, Wu, and Yang (2020) support the idea that, the DQN algorithm with a component of Reinforcement Learning and Deep Learning has attributes that can be modified to attain efficiency

### 3.3.3 Reinforcement Learning Algorithm

The framework of reinforcement learning that forms the DQN algorithm, was built on the concept of an interaction between an agent and an environment. It starts with an agent that observes the state of the environment and in response takes an action, and thereafter the agent receives a reward based on the action it took. This sequential trajectory as shown in Figure 3.6 follows the Markov Decision Processing (Arunraj and Umarani, 2017).

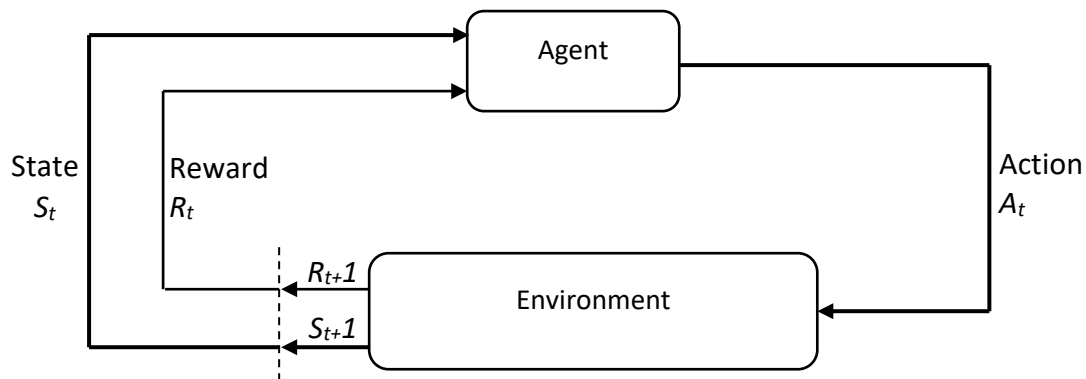


Figure 3.6: The agent-environment interaction in a Markov decision process (Source: Weidong Kuang, 2021)

In reinforcement learning algorithms, the actions of an agent are informed by a Policy. A policy is a matrix of probabilities of actions against any given state that the agent uses to take action. This means that when an agent is guided by policy  $\pi$  at time  $t$ , it follows that  $\pi(a|s)$  is the probability that  $A_t=a$  if  $S_t=s$ . Therefore, at time  $t$ , under policy  $\pi$ , the likelihood of taking action in state  $s$  is  $\pi(a|s)$  (Arunraj and Umarani, 2017). For the agent to decide the best action to take in a given state it relies on value function. Value functions are functions of state or state action pairs that estimate how good it is for an agent to be in a particular state or how good it is for an agent to perform a particular action in a given state (Servin and Kudenko, 2008).

For any state presented by the environment the determining of the best action for that particular state, is based on the expected reward while the target is to receive the highest reward. Therefore, actions taken by the agent in relation to the state is what defines the value functions. Since the agent takes action with reference to the policy which it turn determines the value function, a policy is thus used to specify value functions. ("Policies and Value Functions - Good Actions for a Reinforcement Learning Agent – Deeplizard", 2018). There are two types of value functions namely, state value function and action value function.

According to Deeplizard (2018), the state-value function for policy  $\pi$  denoted as  $V_\pi$  tell how good any given state is for an agent following policy  $\pi$ . In other words, it gives us the value of the state under  $\pi$ . Formally the value of state  $S$  under policy  $\pi$  is the expected return from starting from

state  $s$  at the time  $t$  and following policy  $\pi$  thereafter. This is denoted  $V_{\pi}(s)$  represented in the equation

$$V_{\pi}(s)=E[G_t|S_t=s]$$

Similarly, the action-value function for policy  $\pi$  denotes as  $q_{\pi}$ , tells us how good it is for the agent to take any given action from a given state while following policy  $\pi$ . In other words, it gives us the value of an action under  $\pi$ . Formally, the value of action  $a$  in state  $s$  under policy  $\pi$  is the expected return from state  $s$  at time  $t$ , taking action  $a$ , and following policy  $\pi$  thereafter. This is denoted  $q_{\pi}(s,a)$  in the equation

$$q_{\pi}(s,a)=E[G_t|S_t=s, A_t=a]$$

The left side of the equation is called the Q function referring to the quality of the decision made based on the value it obtained which is indicated on the right side of the equation and referred to as Q value. The Q-values are stored in a Q-table which is a matrix of actions and states as columns and rows respectively. The DQN uses a Deep Learning algorithm which is the deep neural network component to Q-values in the Q-table.

### 3.3.4 Deep Learning Algorithms

Deep learning is a subset/branch of machine learning which is in turn a subset of Artificial Intelligence. Deep learning algorithms are inspired by the pattern of the human brain. It is made up of layers of neurons that are interconnected forming a network. The layers are in three parts namely: the input layer, hidden layers, and output layer as illustrated in Figure 3.7. At the centre of these algorithms is the concept of learning from data. The data is captured by the input layer and passed through to the hidden layers where mathematical computations are performed and then a prediction or a decision is presented by the output layer.

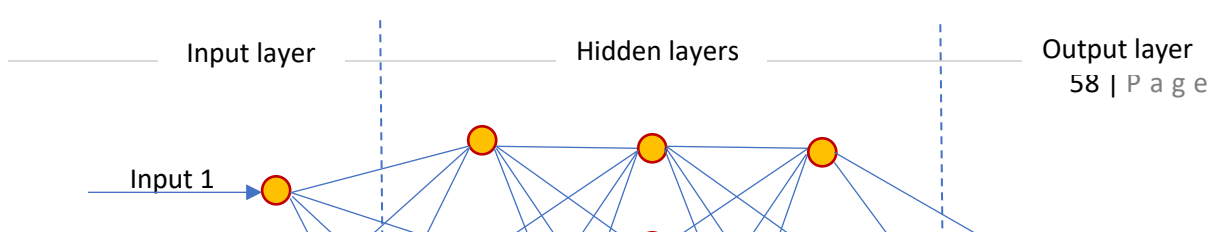


*Figure 3.7: Deep Neural Network*

There are several deep learning algorithms available but convolution Neural Network and Deep Neural Network are two of the most successful algorithms (Ketkar, 2017). Therefore, this study used a hybrid of CNN and DNN algorithms for the neural network component and Q-Learning for the reinforcement learning component of the DQN to build a classifier, as an agent to achieve the objective of building an efficient mechanism.

### **3.3.5 Q-Learning Algorithm**

The Q-learning is a reinforcement learning algorithm that uses Q-values from the Q-table, in which all Q-values are initiated to zero, and are updated during the learning process as depicted in Figure 3.8, in which the agent performs reiteration episodes by taking actions aiming at learning the environment (Hu et al., 2020). An agent takes as many episodes as possible until all the values in the Q-table are filled with a value greater or less than zeros.

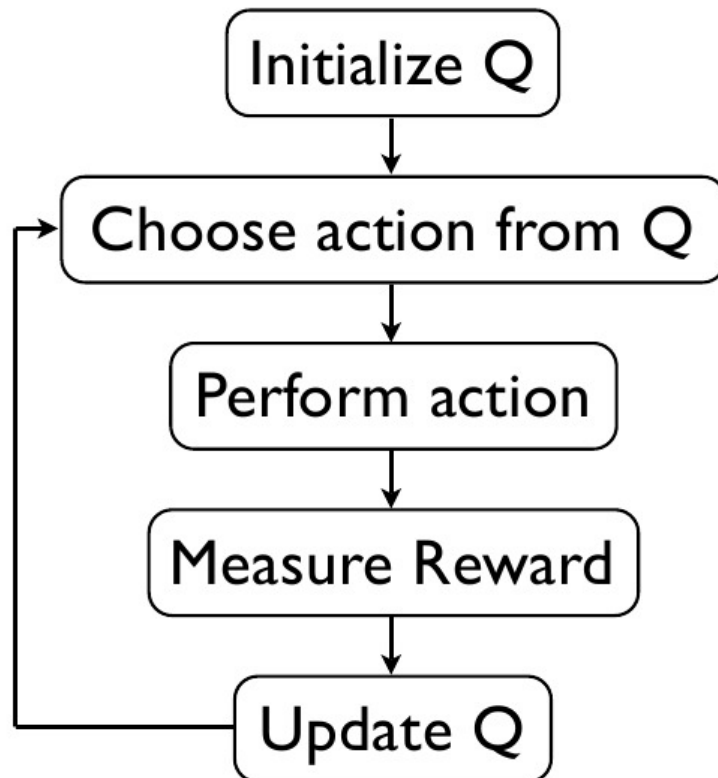


Figure 3.8: Q-Learning Algorithm (Source: Hossen and Janagam, 2018)

The agent's goal is to do actions that will result in the most points. An agent in state  $s$  takes action  $a$  because it receives the Q-value of action  $a$  in state  $s$  in the maximum possible point.

However, this is not applicable when the agent is to take the first action in the first episode when all the Q-values in the Q-table are zero. Therefore, in deciding which first action an agent is to take when the Q value is at Zero, a trade-off technique is employed using either exploration or exploitation. Exploration is when an agent act to discover the environment to find out more information about it, whereas exploitation is when the agent acts using the information it already knows to maximise its points (Gou and Liu, 2019).

This technique is employed in all the states and is not limited to the first action in the first episode. To decide whether to explore or exploit an environment a strategy called the epsilon-greedy strategy was used (Hossen and Janagam, 2018).

Using the episode greedy strategy, the agent was able to pick the best option between exploring and exploiting the environment in any current state. Servin and Kudenko (2008) define epsilon-greedy as a semi-uniform random exploration technique that chooses an action based on a modest value as a base probability. The ultimate purpose of this strategy is to select the optimal action in a current state. To achieve optimality, an exploration rate ( $r$ ), called epsilon, is first defined with the value of 1, making exploration (**E**) the first option for the agent and it takes a random action. This makes the agent greedy to always want to explore. As the agent takes subsequent episodes in the environment, the exploration rate declines by a predefined learning rate, thereby reducing the likelihood of performing an exploration infinitely. A random number between zero and one is generated, and if this number is greater than the exploration rate, the agent will choose its action based on exploitation as Figure 3.9 shows. This trade-off was employed in this study to increase the possibility of obtaining the best result.

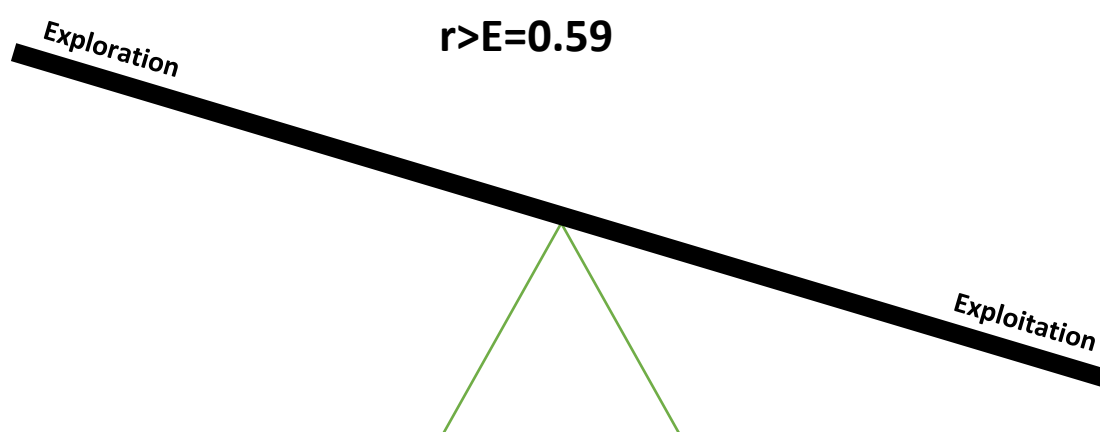


Figure 3.9: Epsilon greedy algorithm trade-off.

### 3.3.6 Convolution Neural Network

A Convolution Neural Network (CNN) is an artificial neural network that is largely used in image analysis such as face recognition (Chilimbi et al., 2014). CNN has also been used for other data analysis and classification problems including DDoS attack detection and has produced tremendous results (Haider, Akhunzada, Ahmed, and Raza, 2019). The basis of a CNN is hidden layers called convolution layers that have the specialised capability to pick out patterns and make sense out of them. This pattern detection ability made CNN useful for this study in building a DDoS attack detection mechanism.

### 3.3.7 Recurrent Neural Network (LSTM)

Recurrent Neural Network is an artificial neural network made of neuron-like nodes that are systematically structured into successive layers in which each node in a given layer is connected directly to every other node in the next successive layer (Salehinejad et al., 2017). RNN is best used for sequential and time-based classification problems and has shown to yield great results. It has two other neural networks built under it called Gated Recurrent Neural Network (GRNN) and the Long Short-Term Memory (LSTM)(Haq et al., 2015). Both have been applied in mechanisms to detect DDoS attacks and LSTM overcome a major challenge in GRNN of vanishing gradient using a memory cell to present the previous, adding to its ability to train faster than other neural networks. (Yuan et al., 2017).

### 3.3.8 Deep Neural Network (DNN)

A Deep Neural Network (DNN) is an algorithm that emerged from the neural network as an advancement of the Neural Network from having a single hidden layer to having more than hidden layers that are hierarchically cascaded between its input and output. Recent research has shown the improved performance of DNN in various fields of applications such as medicine, image classification, and intrusion detection (Potluri and Diedrich, 2016). DNN has shown capability in

classification problems that involve complex and huge data. This capability has made DNN applicable in the classification of a huge amount of network traffic (J. Kim et al., 2017).

Therefore, the study used the CNN and combining with the DNN on the DQN algorithm thereby creating a hybrid Deep Reinforcement Learning mechanism illustrated in Figure 3.10.

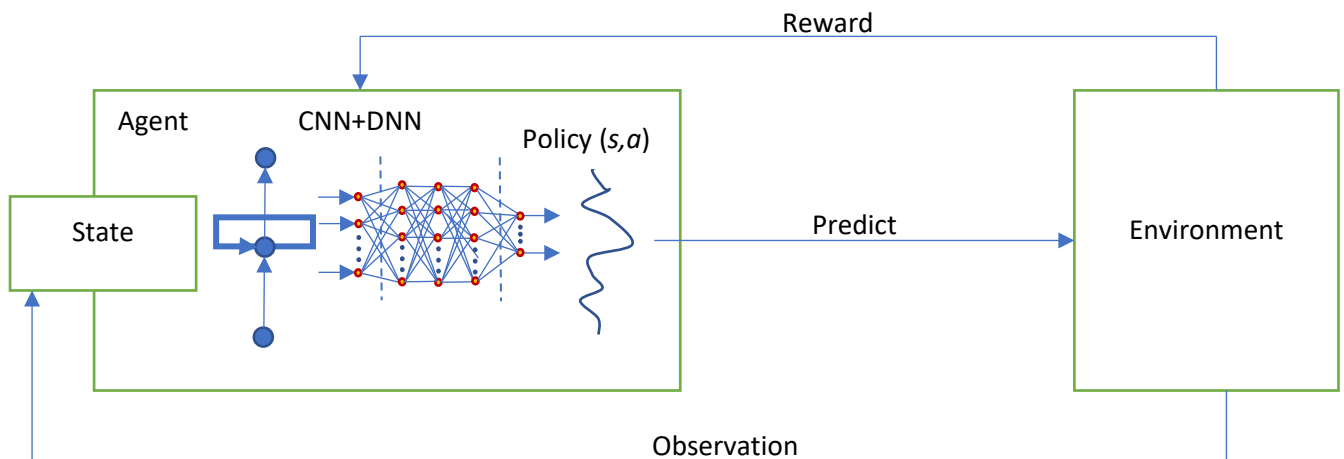


Figure 3.10: Architecture design of Hybrid Deep Reinforcement Learning mechanism.

### 3.3.9 Implementation

The implementation of this study was conducted in a simulated environment. A simulated environment is a virtual workspace that is developed to mimic the real world. Implementing an experiment in a simulated environment is customary when experimenting on a real network is dangerous, costly, and untimely or inconvenient (Dooley, 2002). As network owners would not allow the training and testing of a mechanism to be performed on their live network traffic due to security, legal imprecations, and user privacy, Mishra and Shukla (2018) implemented a DDoS attack detection mechanism in a simulated environment proposed for Internet Service providers.

Of the three simulation types namely; Discrete event, System dynamics (or continuous), and agent-based (Maidstone, 2012), this study used the agent-based simulation in which an agent interacts with the world around it and other agents by observing the changes to learn and adapt accordingly. In this study, the agent in-built using a hybrid deep reinforcement algorithm observes

the environment in the form of features and was able to classify if it is a DDoS attack or normal traffic.

In implementing the design in a simulated environment, the following steps were taken

- i. Build a Custom environment
  - a. Specify an observation space
  - b. Specify an action space
- ii. Specify hybrid Neural Network Architecture
- iii. Build a Neural Network (CNN and DNN)
- iv. Based on the estimated probabilities to select a random action
- v. Set up the training of the Hybrid Neural Network Using Policy Gradient
  - a. Number of training iterations 10-250
  - b. Max steps per episode 100 -1000
  - c. train the policy every 10 episodes
  - d. 0.99 discount rate
- vi. Configure Tensorboard
- vii. Executing the graph in Tensorboard
- viii. Evaluation of the results

### 3.4 Evaluation and Analysis Phase

In this phase, the detection mechanism is evaluated to establish if it addresses the problem identified in line with the objectives set by the research.

The core function of an anomaly detection mechanism is to detect the nature of traffic passing through a network if it is normal traffic or an attack. While performing the detection, the detection mechanism consumes system resources such as CPU cycles and memory which also affects the overall performance of the mechanism. This research sought to design an efficient detection

mechanism for DDoS attacks in SDN that perform a core function using minimum resources. In pursuant to establish whether the designed hybrid DRL mechanism achieved this, the mechanism was evaluated based on the following five parameters:

- Accuracy
- False Positive Rate
- Average Reward
- CPU usage
- Main memory usage

Accuracy is the rate at which the mechanism correctly predicts the nature of instances over the total number of instances. This is calculated as a ratio of the sum total of all correct detection against the sum total of correct and incorrect predictions (Kumar Ahuja and Kumar, 2014). Accuracy is calculated to evaluate the effectiveness of a detection mechanism and to calculate this, the following formula was used:

$$Accuracy = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{True Negative} + \text{False Negative} + \text{False Positive}}$$

False Positive Rate is the ratio of normal instances but the mechanism detected as an attack over the sum total of instances that are normal also referred to as the error rate (Kumar Ahuja and Kumar, 2014). This matrix is also used to measure the effectiveness of the mechanism. To calculate this, the following formula was used:

$$False\ positive\ Rate = \frac{\text{False Positive}}{\text{True Negative} + \text{False Positive}}$$

To obtain values from the experiment, that were used in calculating the accuracy and the false, a confusion matrix tool was used. The confusion matrix represents the classification

results from a mechanism as either true or false (Kumar Ahuja & Kumar, 2014) and are presented in Figure 3.11 as a matrix of True Positive, True Negative, False Positive, and False Negative.

|              | Positive (1) | Negative (0) |
|--------------|--------------|--------------|
| Positive (1) | TP           | FP           |
| Negative (0) | FN           | TN           |

Figure 3.11: Confusion Matrix

- **True Positive (TP):** Its value represents the number of attacks that have been correctly classified as attacks.
- **False Positive (FP):** Its value represents the number of normal traffic that has been misclassified as attacks.
- **False Negative (FN):** Its value represents the number of attacks that have been misclassified as normal traffic.
- **True Negative (TN):** Its value represents the number of normal traffic that has been correctly classified as normal traffic.

The efficiency of the mechanism was measured by evaluating three metrics: Average Reward, CPU usage and memory usage. Average Reward measures the period taken by the mechanism from start up to the point it learns a policy that starts to accumulation more reward as a result of accurate detection. Average Reward is calculated using the number of accumulated rewards over the number of steps taken per given time (Arulkumaran et al., 2017). The results were obtained by plotting the

maximum reward against steps taken over a given period for different combinations of neural networks.

CPU usage measures the number of processing circles allocated to the mechanism for it to perform the detection (Mousavi and Affairs, 2014). This is measured in the proportion of the maximum processing circles available in the system. This metric was obtained from the operating system inbuilt monitor that records the processing resources allocation against each application running.

Memory Usage measures the amount of spaces taken by the application in the main memory against the maximum available in the system, calculated as a proportion of the maximum available memory (Mousavi and Affairs, 2014). This metric was also obtained from the operating system inbuilt monitor that records the memory consumption for each application running.

Using the obtained evaluation metrics, the performance of the proposed mechanism was validated by comparing it with available state-of-the-art DDoS attack detection mechanisms and the findings are present in Section 4.9

### 3.5 Summary

This chapter outlined the approach employed in the design of an efficient DDoS attack detection mechanism in a Software Defined Network using Hybrid Deep Reinforcement Learning. Using the Design Science Research framework, the study focused on achieving the objectives of an efficient mechanism designed through a triangulation of literature and experimentation. The literature was used to inform the design of the study and the methods used in the design and carrying out of the study including the experimentation. Several Deep Reinforcement Learning Algorithms were analysed through both literature and experimentation and the Deep Q-Network was identified to be more suitable for the design and was thus adopted.

The study introduced the HDRLM architectural design that utilised generic Deep Reinforcement Learning algorithms and incorporated two heterogeneous Deep Learning algorithms thereby creating the HDRLM for DDoS attack detection in SDN. The combination of the Deep Learning algorithms was introduced to enhance the efficiency in the utilisation of computational resources while achieving high accuracy in the detection of DDoS attacks in SDN. The architectural design of the proposed HDRLM was first introduced in Section 3.3.13. Figure 3.10 illustrates the architectural design and how the various components of the HDRLM fits in the design and interacts with each other. The design was further implemented in a simulated environment to demonstrate and evaluate its utility.

Evaluation being a critical element in the DSRM, a rigorous evaluation of the mechanism was performed using the CICIDS2017 DDoS dataset. The CICIDS2017 dataset was identified after analysing the CAIDA DDoS, KDD Cup 99, NSL KDD, CICIDS2017, and the DARPA being the five most used datasets for evaluating DDoS attack detection mechanisms. The CICIDS2017 dataset could not be used in its raw state which necessitates a dataset pre-processing. The pre-processing aimed at presenting the dataset into a usable state by eliminating redundancies, and irrelevant attributes and instances such as missing values and infinity values.

Thereafter, a feature selection was performed on the clean dataset through an experiment and literature. The process of feature selection aims at obtaining a subset of all the available attributes of the dataset that are relevant to the task at hand.

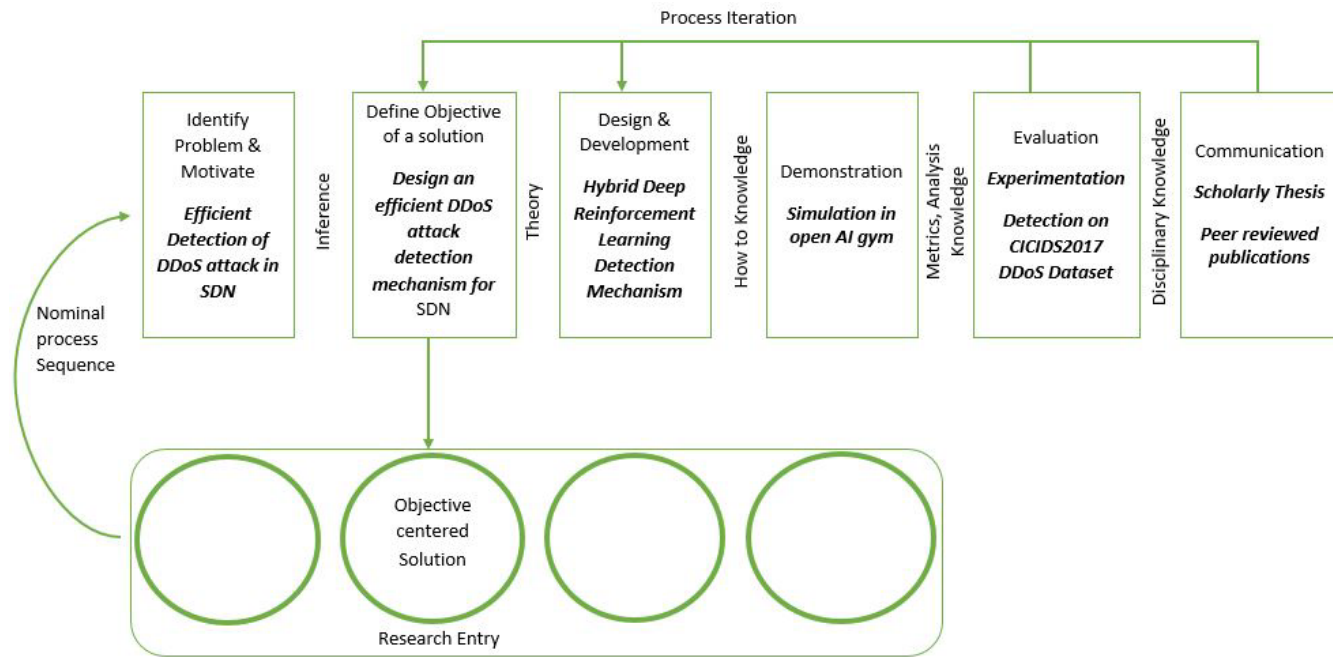


Figure 3.12: DSRM application in Designing Efficient HDRL DDoS attack Detection for SDN

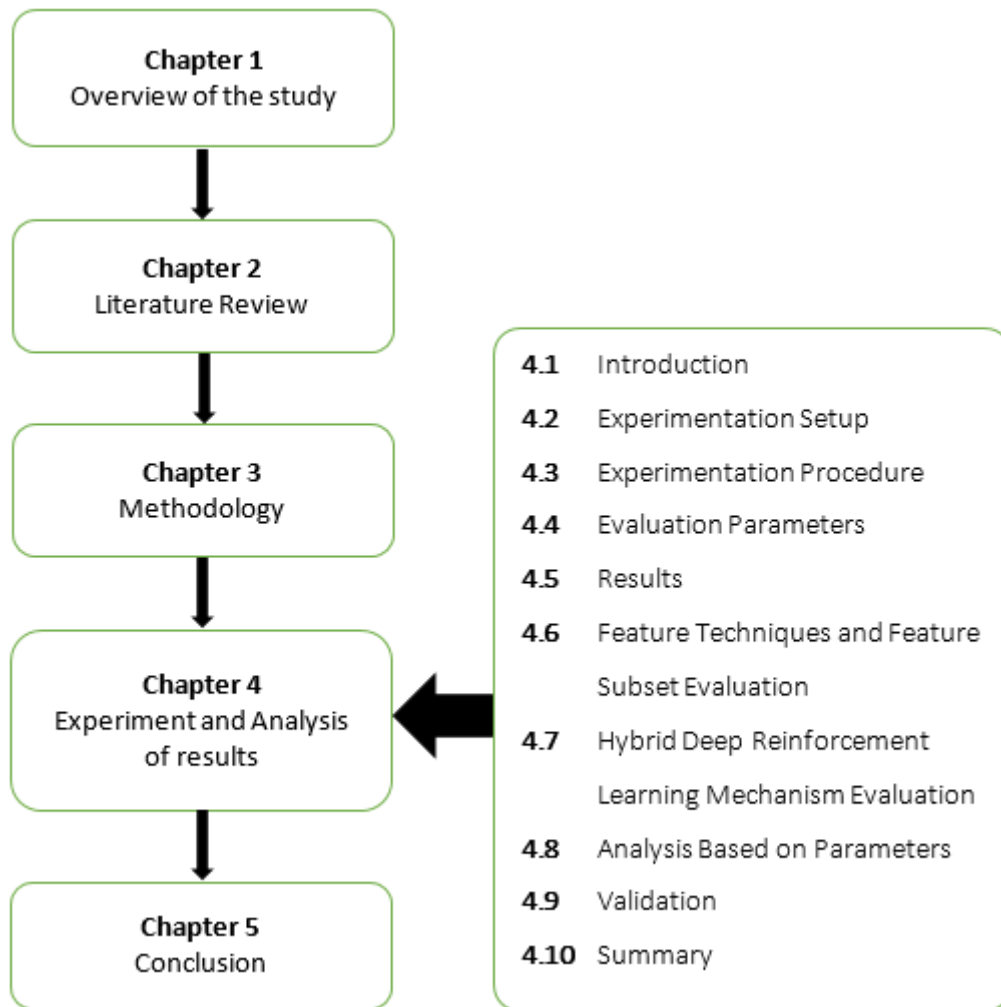
Using two approaches of an experiment and literature review, five feature subsets were obtained. The experiment used an approach of combining four techniques under the filter method to come up with an Ensemble-based Multi-filter Feature Selection (EMFFS) method presented in Figure 3.10.

To establish the performance of the mechanism in terms of its effectiveness and efficiency, five metrics were used which were: Accuracy, False positive Rate, Training Time, CPU usage, and Memory Usage. The values obtained from these matrices were then compared to available state-of-the-art DDoS attack detection mechanisms in SDN to establish the extent to which the study achieved the objectives and addressed the problem. Figure 3.12 shows how the DSRM was applied by the study to design the efficient Hybrid Deep Reinforcement Learning mechanism for DDoS attack detection in Software Defined Network.

## Chapter 4 : Experiment and Analysis of Results

### 4.1 Introduction

This chapter explains the experimentation approach used to evaluate the performance of the designed Hybrid Deep Reinforcement Learning Detection Mechanism (HDRLM) for DDoS attacks in Software Defined Networks (SDN) presented in Section 3.4 and the experiment results. The results used Accuracy, False Positive rate, Average reward, CPU usage and Memory Usage as parameters to establish the performance of the HDRLM. First, the experiment is explained by outlining the setup of the experiment in Section 4.2, followed by experimentation procedure in in the evaluation of the HDRLM in Section 4.3. The parameters used to evaluate the feature selection subsets and the performance of the HDRLM are presented in Section 4.4. Section 4.5 outlines the results of an evaluation of the feature subsets and the evaluation of the HDRLM. The evaluation of the feature subset is presented in Section 4.6 followed by the evaluation of the HDRLM in Section 4.7. Section 4.8 is the analysis of the results based on the parameters. The validation of the results are presented in Section 4.9. This outline is presented in the chapter map below.



## 4.2 Experimentation Setup

### 4.2.1 Hardware

The experiment was performed on an 8<sup>th</sup> generation computer running Windows 10 operating system. The processor was a Core i7-8550U CPU @ 1.80GHz, 1.99GHz, and 8Gigabytes of Random Access Memory. The tools and software applications used to perform the experiment are explained in detail in Section 3.3.1

### 4.2.2 Environment Setup

The simulation was performed using the OpenAI Gym. OpenAI Gym is a toolkit made up of Reinforcement learning libraries, designed for developing reinforcement learning models and

the training of the agent (gym.openai.com). Two key components in the Gym are the observation space and action space. These components were used in the experimentation.

The observation space specifies the structure of elements that the agent observes as the state of the environment and takes an action. The structure of the elements in the observation space is defined as either Discrete or Box. The Discrete space allows non-negative numbers, and the corresponding actions of an agent in discrete space are either 0 or 1. The box space defines an n-dimensional box and is presented as an array of four numbers. The gym has many environments available for research works with a majority of them focused on game controls, some on stock market predictions and a few on intrusion detection. The available gym environment on intrusion detection is built with a range of attacks, therefore they could not be used for this study with a focus on DDoS attacks only. Using the predefined structure of a gym environment, a custom environment was created for this experiment. The observation space for the created custom environment was discrete that was presented by the feature subset is presented in Section 3.2.8.

The action space defined the number of possible actions the agent could take for any given state presented by the environment. In this study, there were only two possible actions the agent could take, either the observed state, which was an instance, was a DDoS attack or it was normal traffic. Therefore, the action space was given a value of 2 (two), translated into either a 1 or a 0 with 1 denoting a DDoS attack and 0 benign. For every action the agent took, it received a reward. The next section explains the reward strategy that was used.

#### **4.2.3 Reward Strategy**

The reward is a crucial element of the Deep Reinforcement Learning algorithm that is meant to encourage the agent to learn by experience. The reward strategy is designed to motivate the agent to take the best action possible by giving it a positive incentive when it takes the right action and punishing it when it takes a wrong action (Sengupta et al., 2013). In this

study, the reward strategy was used in which the agent or classifier received positive points for every correct classification of an instance and negative points when it incorrectly classified an instance. To determine whether the action taken by the agent was correct or not, the labels from the dataset were used as the reference point. When the agent was observing the state of the environment the labels for the instance were excluded and the agent had to take any action without prior training it with labels. After the agent had taken an action based on the observed feature instance, the action was then compared against the respective label of the instance observed to determine the reward the agent would receive. This is illustrated in Use Case 1, Use Case2, Use Case 3, and Use Case 4 as follows:

**Use case 1**

Action =1

Label=1

Reward = Positive.

**Detection result: True Positive**

**Use case 2**

Action =1

Label=0

Reward = Negative.

**Detection result: False Positive**

**Use case 3**

Action =0

Label=1

Reward = Negative.

**Detection result: False Negative**

**Use case 4**

Action =0

Label=0

Reward = Positive.

**Detection result: True Negative**

#### 4.2.4 The Detection Algorithm

The Hybrid detection algorithm was built to detect DDoS attacks by classifying instances from the feature subset to be either benign or a DDoS attack. This detection classifier formed the agent component of the Deep Reinforcement Learning mechanisms. The classification was being performed as the agent actions based on the state observed from the environment. The agents aimed at taking the correct action for every instance to receive more positive rewards. To achieve, this the agent needed to be trained first for it to learn the environment. The agents first had to learn by going through the environment and taking actions either randomly or based on saved memory and receiving either a reward or a punishment. By the agent aiming to get as many rewards as possible, it improved the detection performance, therefore, increased detection accuracy.

The agent or classifier was built with four main components:

1. Epsilon greedy strategy: A greedy algorithm was a part of the agent and was used for the agent to decide between exploring and exploiting the environment for it to have the best result. In exploring, the agent took a random action from the state in the environment intending to gain experience that will result in more positive rewards later, while exploiting the agent used the already discovered knowledge to take an action.

2. Experience Replay and Replay memory: Experience replay is a dataset that was used to store the experiences of the agent for all the episodes the agent took. For each episode, several instances from the feature subset are presented as a state of the environment, are randomly selected and the agent takes actions for each instance and then gets a reward. This

experience is then stored in the Experience replay as a tuple containing the state, the action by the agent, the reward, and the next state. The maximum number of experiences that were stored in the replay memory was set by the experience replay size attribute to be 500k. The replay memory is an important hyperparameter in the Deep Reinforcement Learning algorithm as it has an effect on the performance of the agent (Fedus et al., 2020). However, the increased size of the replay memory consumes a lot of physical memory. It was therefore tuned to 500k for optimal performance for the study. An increased size of the reply memory improves the performance of the agent but requires a lot of physical memory. The dataset in the experience replay was then randomly sampled and used to train the agent. This approach was employed to increase learning efficiency and to deal with a common DQN problem of correlation that arises when sequential instances are used to train an agent (Hu et al., 2020).

3. The neural network: This was used for value approximation as explained in Chapter 3 Section 3.3.2. An ordinary DQN makes use of a single network to approximate a value function. The study used a hybrid neural network for a function approximation to determine the optimal Q-values for each state. The hybrid DQN was created by combining two heterogeneous deep neural networks into one sequential network as the policy network. The choice of algorithms to use is discussed in Section 3.3.2.

One of the well-known challenges with DQN is that they are unstable (Alansary et al., 2019). Therefore, to deal with this challenge, another network which was a replica of the policy network was added as a target network that was eventually used as a stable network (Fan et al., 2019). During an iteration of an episode, the policy network was used and used its model weights to update the target network model weights after a given number of episodes.

4. Loss function: The loss is the prediction error and is calculated as the squared difference between the predicted values from the actual value labels. According to Hu et al. (2020), the loss function improves the performance of an agent in a DQN by increasing the efficiency in

training the neural networks. The neural network requires to have its weights adjusted to increase accuracy, whereas the agent just learns through interacting with the environment without knowing the labels. Therefore, the loss function was used to adjust the weights of the neural network for optimal results. To achieve this, at the end of each episode the loss was calculated by comparing the output value of the local network with an output value of the target network for the given instance.

The parameters were optimised as follows for the experiment.

**Learning rate**

The learning rate was set to 0.01

**Number of iterations**

The total number of training iterations is 10000.

**Maximum steps**

Every episode will take a maximum of 100 steps to predict the outcome.

**Policy update**

It will train the policy every 10 episodes

**Discount rate**

The discount rate is 0.95

### 4.3 Experimentation Procedure

The experiment was initialised by loading the feature subset in the environment. The agent was then presented with an initial instance that was randomly selected from the environment as the current state. Being the initial state, the agent took a random action using the epsilon greedy strategy and received a reward accordingly. This followed many episodes of the agent observing the state and taking action and receiving a reward. Throughout all these initial episodes, the experience of the agent was stored in the experience replay as a tuple until the tuple reached its minimum size of 500.

Once the experience replay had stored enough experiences, the stored experiences were now used to train the agent. A sample of the experiences defined in the batch size of the stored experiences to be at 500, containing a tuple of state, action, reward, and next state was randomly selected. The tuple as the current state was then fed into the policy network to query for Q values and the subsequent tuple as the next state was fed into the target network to query future Q values. The loss was then calculated by the squared difference between target Q-value and predicted Q-value from the local network using the formula below:

$$\text{Loss}=(y-f(x))^2$$

When it is not the last instance which is a terminal state, a computation of the product of the highest value of the queried Q values from the future state is added to the reward to obtain a new Q value but for a terminal state, the reward value is used as the Q value for the respective state. The state was the input value to the neural network denoted as X and the new Q value was the actual value expected from a prediction denoted as Y. The values of X and Y were fed into the local Neural Network to train it. At this point, a gradient descent was applied to adjust the weights of the local network. This was repeated and at every 50th iteration, the target network was updated with the adjusted weights of the local network. These steps were repeated for 10,000 episodes.

#### 4.3.5 Results Recording

A Tensorboard was used to record the performance of the detection mechanism and capture results during the experiment. All the values are captured in the Tensorboard and presented in a confusion matrix and a graph from the initial to the end of the process. The graphs in the Tensorboard are smoothed to 0.95 to enhance visibility.

#### 4.4 Evaluation Parameters

Using the experiment, the study identified the best feature selection technique and feature subset as presented in Section 3.2.8 and evaluated the performance of the Hybrid Deep Reinforcement Learning mechanism. Accuracy and False Positive Rate (FPR) were used to identify the best-performing features subset on the Hybrid Deep Reinforcement Learning mechanism as shown in Section 3.2.11. The best feature subset was then used to evaluate the performance of the HDRLM.

The performance of the Deep Reinforcement Learning mechanism was evaluated based on five parameters namely Accuracy, False Positive Rate, Average Reward, CPU usage, and Memory usage to ascertain its efficiency and effectiveness. The main objective of the study was to design an efficient mechanism, but the effectiveness was also taken into account in evaluating the performance. Efficiency and effectiveness cannot completely be divorced from each other in evaluating the performance of a detection mechanism because the hybrid DLR detection mechanism is designed to effectively detect DDoS attacks at the same time it uses up computational resources and takes a certain amount of time when performing the detection. According to Kumar Ahuja and Kumar (2014) efficiency is achieved when the detection is performed with minimum available resources while effectiveness is achieved when the design successfully performs the detection accurately which it was built, although most research in designing detection mechanisms focus on the effectiveness with no or little attention on efficiency. Therefore, despite the main focus for this research being efficiency, the effectiveness was also evaluated to ensure the capability of the HDRLM.

There are many parameters available that have been used before to measure the effectiveness of a mechanism. According to a survey by Tavallaee (2011), the most used are accuracy and False Positive Rate (FPR). The two parameters together with a third parameter namely average reward (Andrew, 1998; Gou and Liu, 2019; Arulkumaran et al., 2017), were

therefore used to evaluate the effectiveness of the HDRLM. In evaluating the efficiency of the deep reinforcement learning mechanism, the study used CPU usage and memory usage as parameters for evaluation (Mousavi and Affairs, 2014 ; Scarfone et al., 2007).

## 4.5 Results

The three Deep neural networks Long Short-Term Memory (LSTM), Convolution Neural Network (CNN), and Deep Neural Network (DNN) were used in the experiment by a permutation of two of the three Neural Networks to identify the most efficient and effective combination in the creation of the hybrid DQN. First, the five feature subsets were evaluated to achieve the second objective of the study, to identify the best feature selection technique and the best feature subset for the Deep Reinforcement Learning detection mechanism evaluation, by subjecting each feature subset at a time for each permutation. Secondly, the feature subset identified to be the best was used to evaluate the Deep Reinforcement Learning detection mechanism. Adjustments of the parameters were performed at each experiment to optimise the agent. At each instance of the experiment, the evaluation parameters were recorded and are presented in the Sections 4.6 through Section 4.7.

## 4.6 Feature Technique and Feature Subset Evaluation

The five feature subsets identified in Section 3.2.8 to be suitable for the evaluation were compared in an experiment to establish the most effective feature subset. Using the HDRLM, the five feature subsets were evaluated using the accuracy achieved when used in each of the three permutations of the HDRLM with the value approximator in the order LSTM+CNN, LSTM+DNN, and CNN+DNN.

### 4.6.6 Accuracy

In the experimentation with LSTM and CNN, feature subset 4 performed better than feature subsets 1, 2, 3, and 5 by achieving a 97.7% accuracy, followed by feature subset 3 with an accuracy

of 96.4 %. Feature subset 1 performed the lowest at 77.4%. Figure 4.1, presents the performance of the feature subsets under the mechanism with LSTM and CNN.

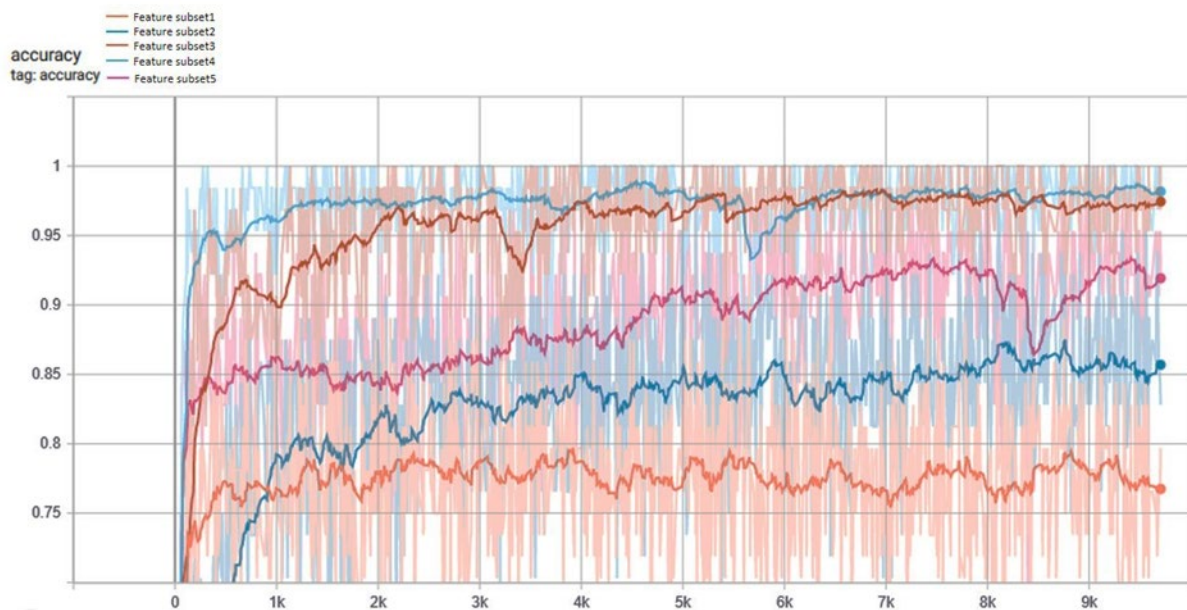


Figure 4.1: Feature subsets accuracy on LSTM+CNN

The five feature subsets were again used on a combination of LSTM and DNN. Figure 4.2 illustrates the performance of the five feature subsets. The feature subset 4 initially performed well but after episode 7,000 it suddenly dropped. The sudden drop could not be explained but could be a loss of memory. Feature subset 3 performed below feature subset 4 at the beginning and remained stable and improving to achieve a 97.5% accuracy. This was followed by feature subset 1 which achieved 96.6% but the performance dropped after 6000 episodes and started to pick again after 7000 episodes. Feature subset 2 did not perform impressively out of the five. It achieved an accuracy of 87% which is the lowest.

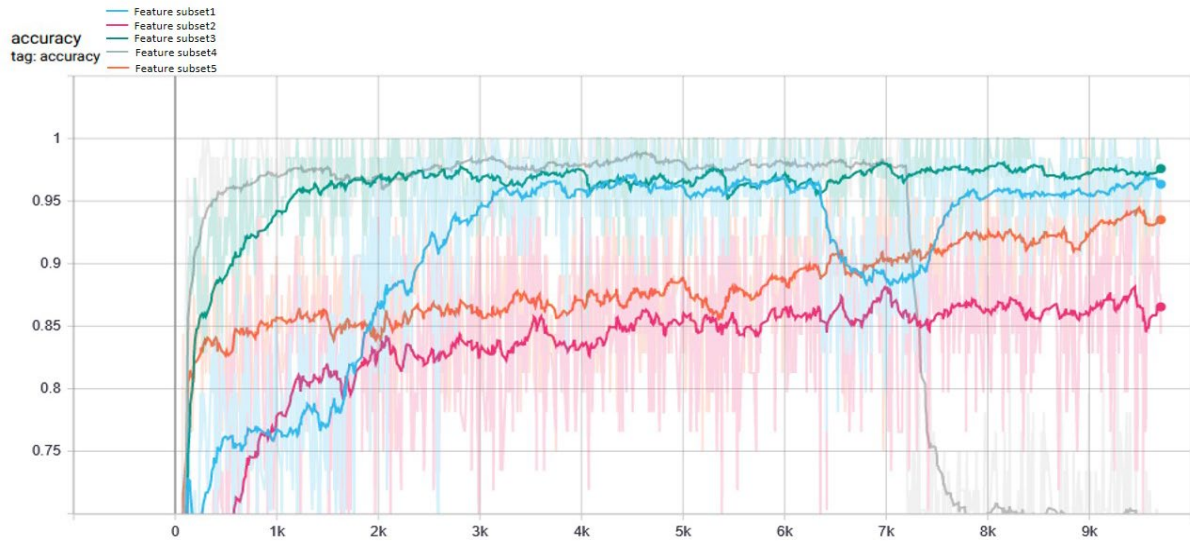


Figure 4.2: Feature subsets accuracy on LSTM+DNN

A third permutation used a combination of CNN and DNN and compared the performance of the five feature subsets. In the results presented in Figure 4.3, feature subset 4 emerged as the best performing feature subset achieving a 98.8% accuracy followed closely by feature subset 3 at an accuracy of 98%.

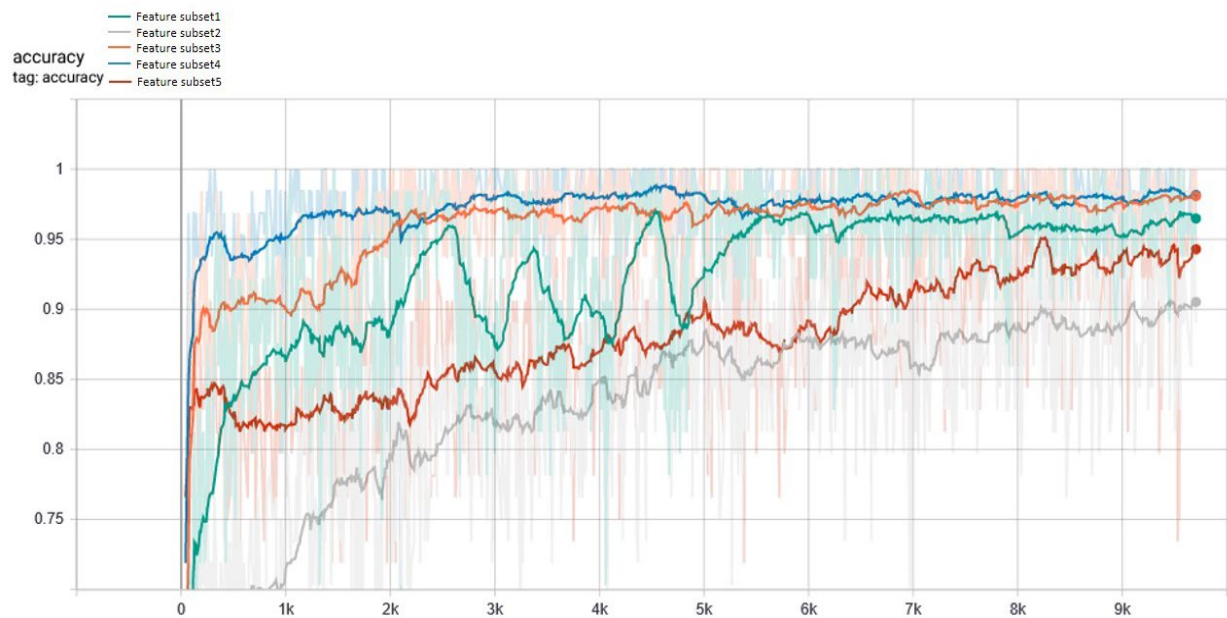


Figure 4.3: Feature subsets accuracy on CNN +DNN

#### 4.6.7 False Positive Rate

A false positive rate for each of the five feature subsets was obtained from the computation of matrices from the confusion matrix for each of the three permutations of the Hybrid Deep Reinforcement Learning detection mechanism. The formulae used to compute the false positive rate for each feature subset was;

$$\textbf{False Positive Rate} = \frac{\textit{False Positive}}{\textit{True Negative} + \textit{False Positive}}$$

Figure 4.4 presents a set of confusion matrixes for all the feature subsets, (a) feature subset 1, (b) feature subset 2, (c) feature subset 3, (d) feature subset 4, and (e) feature subset 5. The results were obtained when the HDRL mechanism used a combination of LSTM and CNN for value approximation.

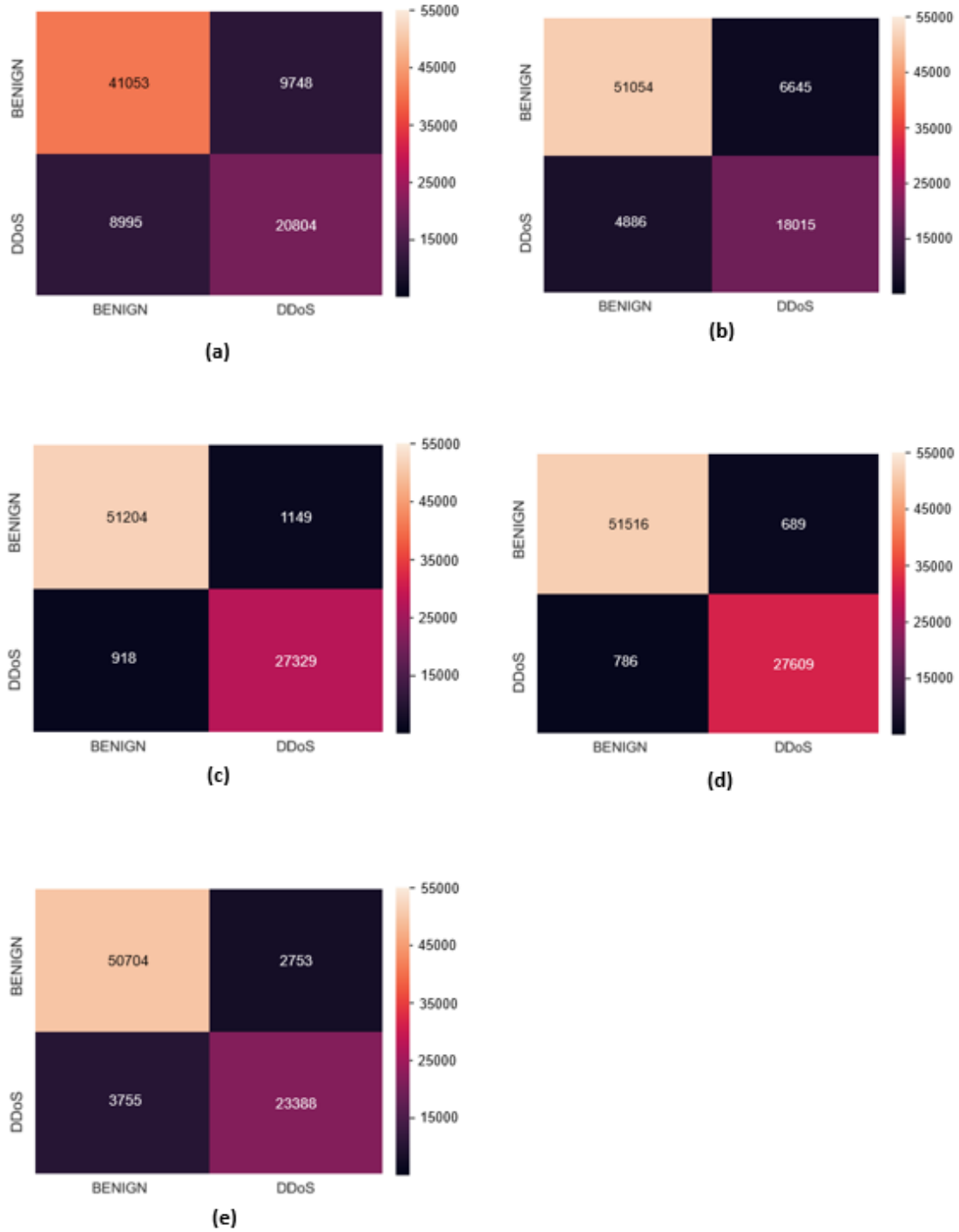


Figure 4.4: Confusion matrices for all feature subsets on HDRL mechanism with LSTM and CNN

Figure 4.4 (a) shows the general performance of feature subset 1 to have correctly classified 20804 as DDoS attacks and 41053 instances correctly classified as benign whereas 9748 were incorrectly classified as DDoS attacks when they were benign and 8995 instances were incorrectly classified as benign when in fact they were DDoS attacks. Figure 4.4 (b) shows results for feature

subset 2 where 18015 instances were correctly classified as DDoS attacks and 51054 instances correctly classified as benign. However, 6645 instances were incorrectly classified to be DDoS attacks which were benign and 8995 instances were incorrectly classified as benign when in fact they were DDoS attacks. The result for feature subset3 is presented in Figure 4.4 (c) where 27329 instances were correctly classified as DDoS attacks and 51204 instances correctly classified as benign whereas 918 instances which were DDoS attacks were incorrectly classified as benign and 1149 benign instances were incorrectly classified as DDoS attacks. Figure 4.4 (d) shows the classification of instances by the HDRLM with an algorithm combining LSTM and CNN and when feature subset 4 is used. The results show that 27609 were correctly classified as DDoS attacks and 51516 instances correctly classified as benign. Whereas, 689 were incorrectly classified to be DDoS attacks when they were benign and 786 instances were incorrectly classified as benign when in fact they were DDoS attacks. Feature subset 5 results presented in Figure 4.4 (e) shows 23388 instances correctly classified as DDoS attacks and 50704 instances correctly classified as benign while 2753 DDoS attack instances were incorrectly classified as benign and 3755 benign instances were incorrectly classified as DDoS attacks.

From the results in Figure 4.4, the calculated FPR for each feature subset is presented as follows:

$$\text{Feature subset 1 (a)} \quad \text{False Positive Rate} = \frac{9748}{20804+9748}$$

$$FPR = 0.32$$

$$\text{Feature subset 2 (b)} \quad \text{False Positive Rate} = \frac{6645}{18015+6645}$$

$$FPR = 0.27$$

$$\text{Feature subset 3 (c)} \quad \text{False Positive Rate} = \frac{1149}{27329+1149}$$

$$FPR = 0.04$$

$$\text{Feature subset 4 (d)} \quad \text{False Positive Rate} = \frac{689}{27609+689}$$

$$FPR = 0.02$$

$$\text{Feature subset 5 (e)} \quad \text{False Positive Rate} = \frac{2753}{23388+2753}$$

$$FPR = 0.11$$

Table 4.1 presents the summary of calculated FPR for each of the feature subsets under the combination of LSTM and CNN. Feature subset 4 had the lowest false positive rate at 0.02 and features subset 3 with a false positive rate of 0.04 was the second-lowest while feature subset 1 had the highest rate of 0.32.

Table 4.1: False positive rate results for feature subsets on LSTM+CNN

| Feature           | False Positive Rate |
|-------------------|---------------------|
| Feature Subset 1  | 0.32                |
| Feature Subsett 2 | 0.27                |
| Feature Subset 3  | 0.04                |
| Feature Subset 4  | 0.02                |
| Feature Subset 5  | 0.11                |

From the results, it shows that when the feature subset 4 is used on the HDRLM which had a combination of LSTM and CNN, there is a 2% probability of classifying an instance as a DDoS attack when it is benign whereas when feature subset 1 is used the probability is 32%. The next permutation of the HDRLM had a combination of LSTM and DNN.

Figure 4.5 presents a set of a confusion matrix for all the feature subsets, (a) is for feature subset 1, (b) is for feature subset 2, (c) is for feature subset 3, (d) is for feature subset 4, and (e) is for feature subset 5. The confusion matrix results were obtained when the HDRL mechanism used a combination of LSTM and DNN.

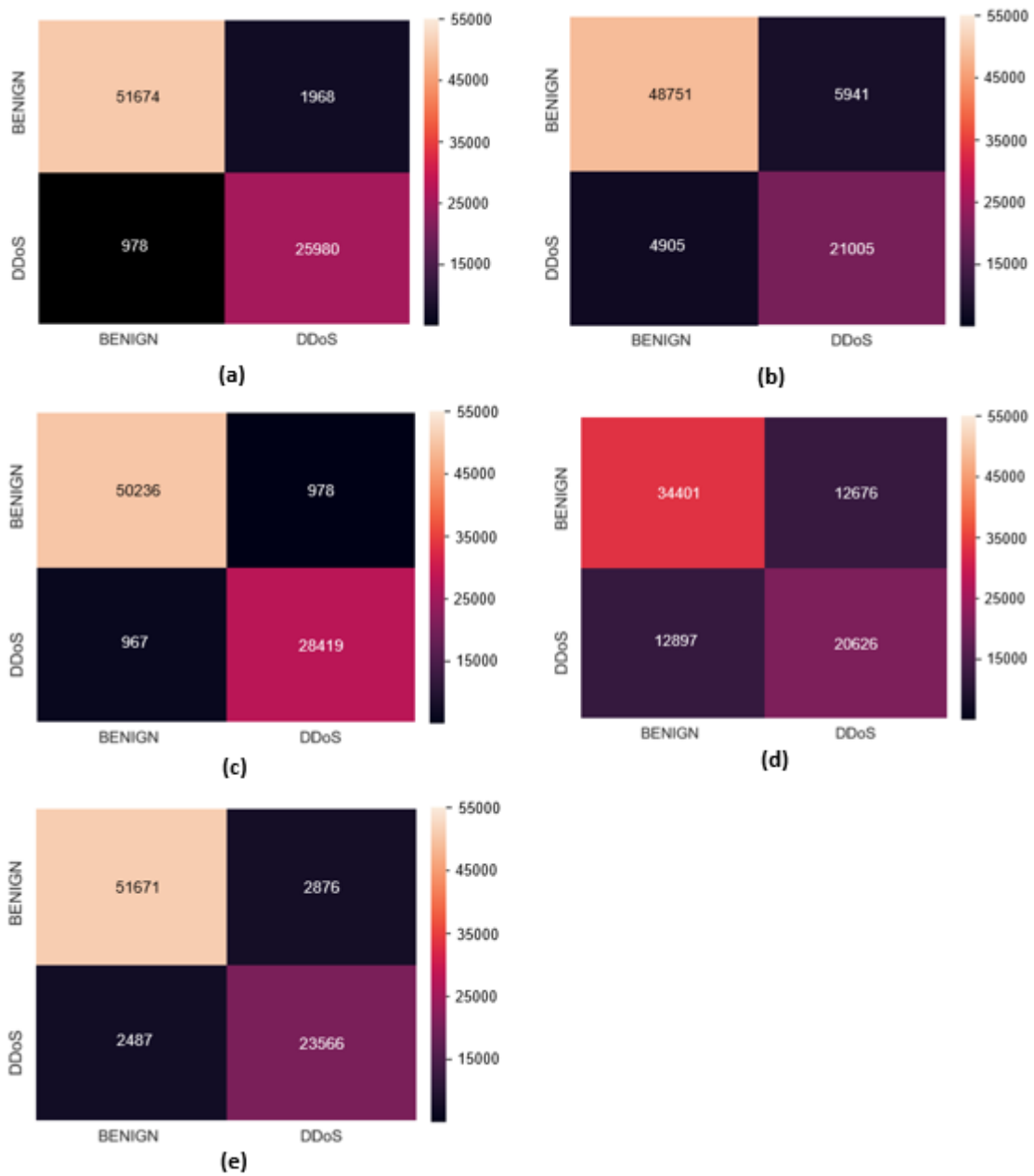


Figure 4.5: Confusion matrices for all feature subsets on HDRL mechanism with LSTM and DNN

Figure 4.5 (a) presents results for feature subset 1 where 25900 instances were correctly classified as DDoS attacks and 51674 instances correctly classified as benign. However, 1968 benign instances were incorrectly classified to be DDoS attacks and 978 instances were incorrectly classified as benign when in fact they were DDoS attacks. Figure 4.5 (b) shows results for feature subset 2 where 21005 instances were correctly classified to be DDoS attacks and 48751 instances correctly

classified as benign. However, 5941 instances were incorrectly classified to be DDoS attacks which were benign and 4905 instances were incorrectly classified as benign when in fact they were DDoS attacks. The result for feature subset 3 is presented in Figure 4.5 (c) where 28419 instances were correctly classified to be DDoS attacks and 50236 instances correctly classified as benign whereas 978 instances which were DDoS attacks were incorrectly classified as benign and 967 benign instances were incorrectly classified as DDoS attacks. Figure 4.5 (d) shows the classification of instances by the HDRLM with an algorithm combining LSTM and DNN and when feature subset 4 is used. The results show that 20626 were correctly classified as DDoS attacks and 34401 instances correctly classified as benign. Whereas, 12676 were incorrectly classified to be DDoS attacks when they were benign and 12897 instances were incorrectly classified as benign when in fact they were DDoS attacks. Feature subset 5 results presented in Figure 4.5 (e) show 23566 instances correctly classified as DDoS attacks and 51671 instances correctly classified as benign while 2876 DDoS attack instances were incorrectly classified as benign and 2487 benign instances were incorrectly classified as DDoS attacks.

The calculated FPR for each of the feature subsets from the results shown in Figure 4.5 were as follows:

$$\text{Feature subset 1(a)} \quad \text{False Positive Rate} = \frac{1968}{25980+1968}$$

$$FPR = 0.07$$

$$\text{Feature subset 2 (b)} \quad \text{False Positive Rate} = \frac{5941}{21005+5941}$$

$$FPR = 0.22$$

$$\text{Feature subset 3 (c)} \quad \text{False Positive Rate} = \frac{978}{28419+978}$$

$$FPR = 0.03$$

$$\text{Feature subset 4 (d)} \quad \text{False Positive Rate} = \frac{12676}{20626+12676}$$

$$FPR = 0.38$$

$$\text{Feature subset 5 (e)} \quad \text{False Positive Rate} = \frac{2876}{23566+2876}$$

$$FPR = 0.11$$

Table 4.2 shows the computed false positive rates against each feature subset when the HDRLM had a combination of LSTM and DNN. Feature subset 3 had the lowest rate of 0.03 followed by feature subset 1 at 0.07. Feature subset 4 performed the worst in terms of false positive rate with the highest rate of 0.38. The results indicated that when feature subset 3 was used on the HDRLM which had a combination of LSTM and DNN, the probability of misclassification was 3% which was the lowest. Following feature subset 3 was feature subset 1 with a probability of misclassification of 7% and the highest probability was 38% that was obtained from feature subset 4. The next permutation of the HDRLM had the combination of CNN and DNN.

*Table 4.2: False positive rate results for feature subsets on LSTM+DNN*

| Feature          | False Positive Rate |
|------------------|---------------------|
| Feature Subset 1 | 0.07                |
| Feature Subset 2 | 0.22                |
| Feature Subset 3 | 0.03                |
| Feature Subset 4 | 0.38                |
| Feature Subset 5 | 0.11                |

When the HDRL mechanism used the combination of CNN and DNN for value approximation, the performance of the feature subsets is presented as a confusion matrix in Figure 4.6, (a) feature subset 1, (b) feature subset 2, (c) feature subset 3, (d) feature subset 4, and (e) feature subset 5. The confusion matrix results were obtained when the HDRL mechanism used a combination of LSTM and DNN.

Figure 4.6 (a) presents results for feature subset 1 where 51406 instances were correctly classified as benign and 26359 instances correctly classified as DDoS attacks, whereas, 1397 benign instances were incorrectly classified to be DDoS attacks and 1438 instances were incorrectly

classified as benign when in fact they were DDoS attacks. Figure 4.6 (b) presents results for feature subset 2 where 22348 instances were correctly classified to be DDoS attacks and 50614 instances correctly classified as benign, while 3784 instances were incorrectly classified to be DDoS attacks which were benign and 3854 instances were incorrectly classified as benign when in fact they were DDoS attacks. The result for feature subset 3 is presented in Figure 4.6 (c) where 27809 instances were correctly classified to be DDoS attacks and 51245 instances correctly classified as benign whereas 807 instances which were DDoS attacks were incorrectly classified as benign and 739 benign instances were incorrectly classified as DDoS attacks. Figure 4.6 (d) shows the classification of instances by the HDRLM with an algorithm combining CNN and DNN and when feature subset 4 was used. From the results, 27599 were correctly classified as DDoS attacks, 51518 instances were correctly classified as benign, 687 were incorrectly classified to be DDoS attacks when in fact they were benign and 796 instances were incorrectly classified as benign when in fact they were DoS attacks. Feature subset 5 results presented in Figure 4.6 (e) shows 25059 instances correctly classified as DDoS attacks and 50135 instances correctly classified as benign while 1849 DDoS attack instances were incorrectly classified as benign and 2757 benign instances were incorrectly classified as DDoS attacks.

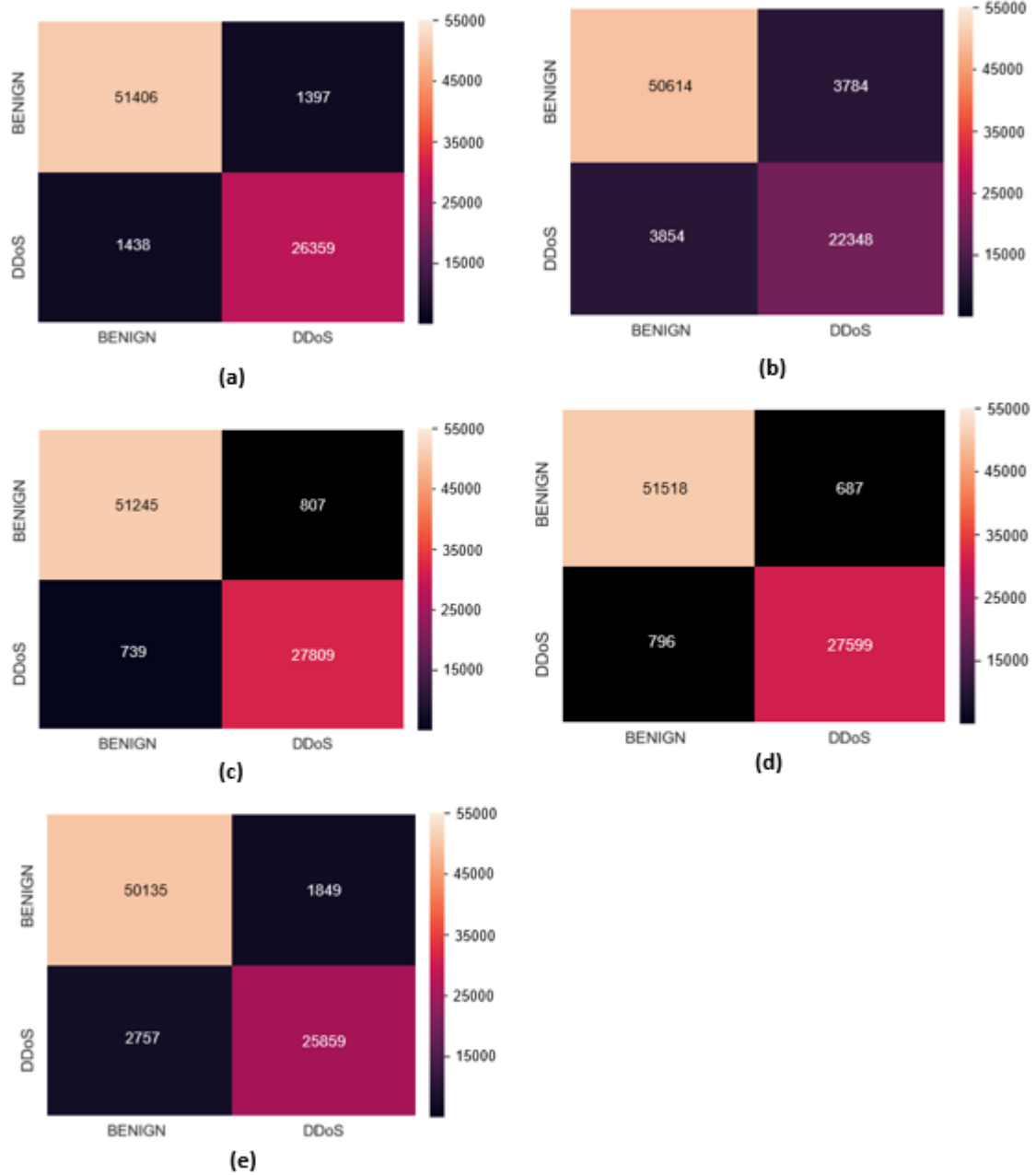


Figure 4.6: Confusion matrices for all feature subsets on HDRL mechanism with CNN and DNN

The calculated false positive rate was as follows;

Feature Subset 1 (a)  $False\ Positive\ Rate = \frac{1397}{26359+1397}$

$FPR = 0.05$

Feature Subset 2 (b)  $False\ Positive\ Rate = \frac{3784}{22348+3784}$

$$FPR = 0.14$$

$$\text{Feature Subset 3 (c)} \quad \text{False Positive Rate} = \frac{807}{27809+807}$$

$$FPR = 0.03$$

$$\text{Feature Subset 4 (d)} \quad \text{False Positive Rate} = \frac{687}{27599+687}$$

$$FPR = 0.02$$

$$\text{Feature Subset 5 (e)} \quad \text{False Positive Rate} = \frac{1849}{25859+1849}$$

$$FPR = 0.07$$

Table 4.3 presents the summary of the calculated false positive rate for each of the feature subsets under the combination of CNN and DNN. Feature subset 4 had a FPR of 0.02 which was the best among the rest, followed by feature subset 3 at 0.03 and the highest was feature subset 2 at 0.14.

Table 4.3: False positive rate results for feature subsets on CNN+DNN

| Feature          | False Positive Rate |
|------------------|---------------------|
| Feature Subset 1 | 0.05                |
| Feature Subset 2 | 0.14                |
| Feature Subset 3 | 0.03                |
| Feature Subset 4 | 0.02                |
| Feature Subset 5 | 0.07                |

Feature subset 4 comes out to be more effective on the HDRLM which has a combination of CNN and DNN as the probability of DDoS attacks instances being misclassified to be benign is the lowest at 2%. The second lowest was when feature subset 3 is used, the probability of misclassification was 3%. The highest rate of misclassifying DDoS attacks was 14% when feature subset 2 is used on the HDRLM which has a CNN and DNN combination.

#### 4.6.8 Summarised Results for Feature Subsets Evaluation

Table 4.4 shows the overall results in terms of accuracy for the five feature subsets on all three permutations of the DQN Hybrid Detection mechanism for DDoS attacks. Feature subset 3 and feature subset 4 emerged with the highest accuracy. Feature subset 4 achieved the highest accuracy on LSTM+CNN at 98.17% and 98.16% on CNN+DNN despite the lowest attained accuracy of 68.27% on LSTM+DNN.

*Table 4.4: Summarised feature subsets Accuracy*

|                         | <b>LSTM+CNN</b> | <b>LSTM+DNN</b> | <b>CNN+DNN</b> |
|-------------------------|-----------------|-----------------|----------------|
| <b>Feature Subset 1</b> | 76.75           | 96.34           | 96.48          |
| <b>Feature Subset 2</b> | 85.69           | 86.54           | 90.52          |
| <b>Feature Subset 3</b> | 97.44           | 97.59           | 98.08          |
| <b>Feature Subset 4</b> | 98.17           | 68.27           | 98.16          |
| <b>Feature Subset 5</b> | 91.93           | 93.35           | 94.29          |

The False Positive rate for the five feature subsets is presented in Table 4.5. Feature subset 4 showed the lowest False Positive Rate at 0.02 from feature subset 4 on two of the permutations which were LSTM+CNN and CNN+DNN representing that when it is used the probability of misclassifying DDoS attacks as benign is the lowest at 2%. However, the FPR 0.38 from feature subset 4 on LSTM+DNN was the highest representing a probability of 38% on misclassification of DDoS attacks as benign. This poor performance of feature subset 4 on LSTM+DNN was attributed to Reinforcement learning's inherent behaviour of being unstable (Andrew, 1998; Hossen and Janagam, 2018).

Feature subset 3 performed consistently on the HDRLM which had a combination of LSTM and CNN and the other permutation with the HDRLM having a combination of CNN and DNN with an FPR of 0.03 representing a 3% probability of misclassifying DDoS attacks as benign. When the HDRLM had a combination of LSTM+CNN the FPR slightly rose to 0.04 which was a 4% likelihood of missing a DDoS attack.

Table 4.5: Summarised feature subsets false positive rate

|                         | LSTM+CNN | LSTM+DNN | CNN+DNN |
|-------------------------|----------|----------|---------|
| <b>Feature Subset 1</b> | 0.32     | 0.07     | 0.05    |
| <b>Feature Subset 2</b> | 0.27     | 0.22     | 0.14    |
| <b>Feature Subset 3</b> | 0.04     | 0.03     | 0.03    |
| <b>Feature Subset 4</b> | 0.02     | 0.38     | 0.02    |
| <b>Feature Subset 5</b> | 0.11     | 0.11     | 0.07    |

From the results, feature subset 4 and feature subset 3 performed well compared to feature subset 1, feature subset 2 and feature subset 5. Therefore, feature subset 3 and feature subset 4 were used in the study to evaluate the performance of the HDRLM for detecting DDoS attacks in SDN before conclusively determining the best feature subset between the two. The performance of the deep reinforcement learning was then evaluated and presented in the following section.

#### 4.7 Hybrid Deep Reinforcement Learning Mechanism Evaluation

The designed HDRLM for detecting DDoS attack was evaluated on effectiveness and efficiency. The five parameters used to evaluate the mechanism were:

- Accuracy
- False Positive Rate
- Average Reward
- CPU usage
- Main memory usage

Effectiveness was evaluated using the first three parameters of accuracy and False Positive Rate. Accuracy measures the correctness of the prediction by the mechanism and is calculated as a ratio of the total of all correct detection against the total of correct and incorrect predictions (Kumar Ahuja and Kumar, 2014). False Positive Rate is the ratio of normal instances that the mechanism detected as an attack over the total of normal instances. It is also referred to as the error rate (Kumar Ahuja and Kumar, 2014). The Average reward is the number of positive results? the

mechanism obtained over several actions taken and its learning ability of the mechanism as a Deep Reinforcement Learning agent.

To evaluate the efficiency, the study used the CPU usage and Memory usage as parameters. The CPU and Memory are the main and critical resources that are used for a computer to perform any task such as detection. The results for the evaluation based on each parameter are presented in the following sections.

#### 4.7.9 Accuracy

Accuracy is the rate at which the mechanism correctly predicts the nature of instances over the total number of instances. When the mechanism was evaluated with feature subset 3, the CNN+DNN obtained the highest accuracy of 98.08% while LSTM+DNN achieved a 97.59% and a 97.44% by LSTM+CNN as shown in Figure 4.7. Figure 4.8 shows the performance when feature subset 4 was used to evaluate the mechanism. LSTM+CNN attained 98.17% and CNN+DNN attained a 98.16% accuracy. An inconsistent performance was observed on LSTM+DNN where the accuracy suddenly dropped after episode 7000 and resulted in the lowest accuracy of 68.27%.

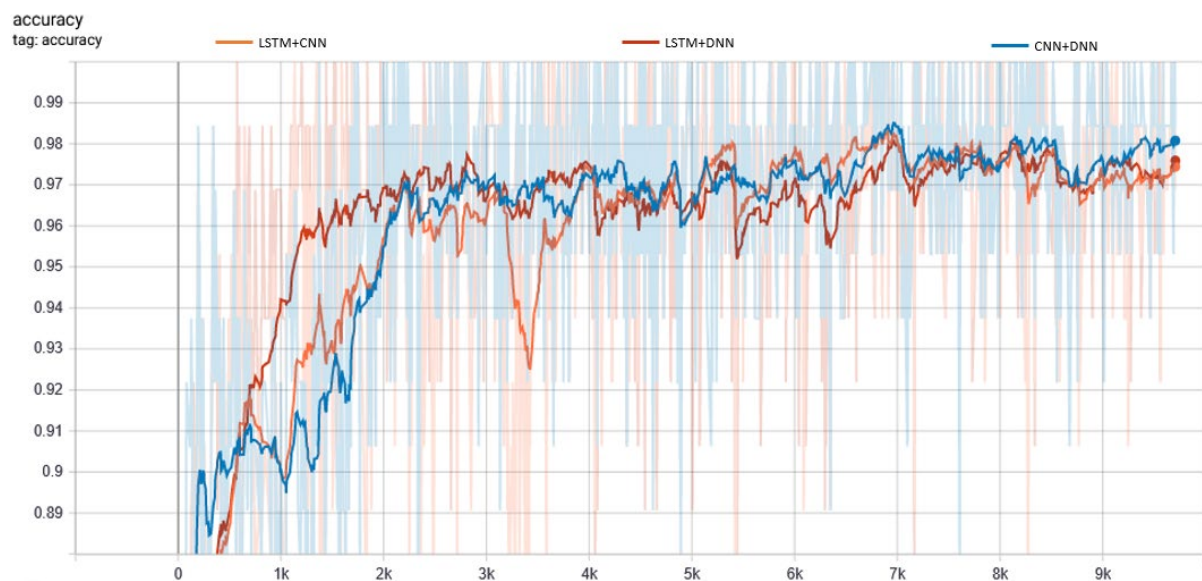


Figure 4.7: HDRLM Accuracy evaluated with FS3

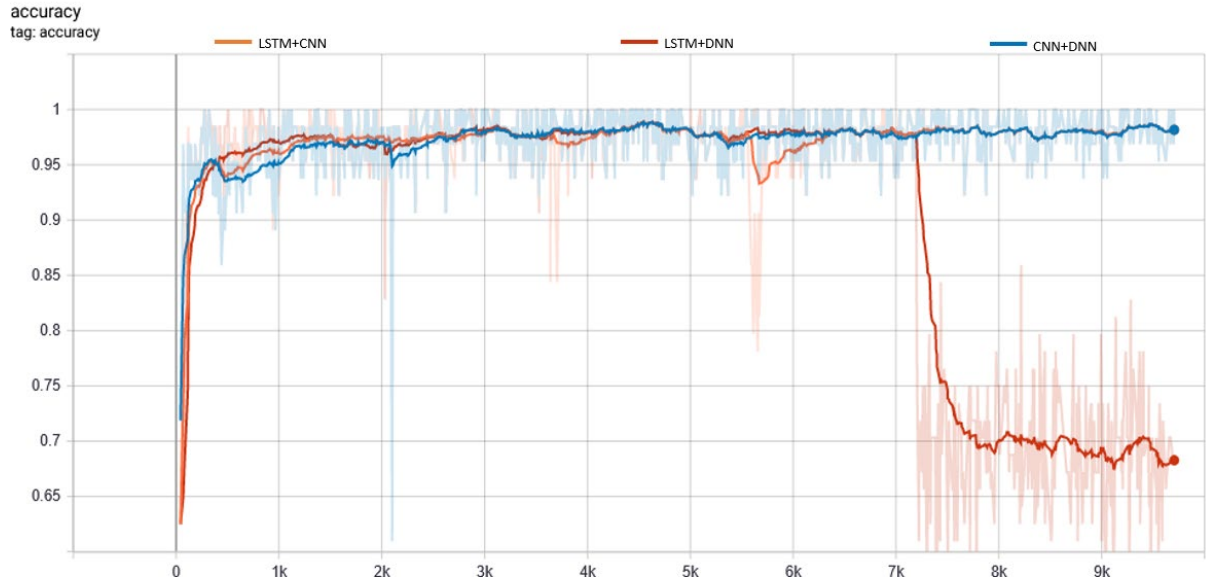


Figure 4.8: HDRLM Accuracy evaluated with FS4

#### 4.7.10 False Positive Rate

The False Positive Rate was also used to evaluate the HDRLM. False positive rate is the ratio of normal instances but the mechanism detected as an attack over the total of normal instances, is also referred to as the error rate (Kumar Ahuja and Kumar, 2014). Using the formula for FPR

$$False\ Positive\ Rate = \frac{False\ Positive}{True\ Negative + False\ Positive}$$

the best result on FPR for each feature subset 3 and feature subset 4 on all the three permutations of LSTM and CNN, LSTM and DNN, and CNN and DNN were compared to establish the best performing feature subset between the two in terms of HDRLM detecting DDoS attacks.

Figure 4.9 presents a confusion matrix for feature subset 3 on LSTM and CNN, in which 28478 instances of DDoS attacks, 27329 instances were correctly classified to be DDoS attacks while 1149 were classified to be benign when they were DDoS attacks and out of 51022 benign instances 918 instances classified as DDoS attack when they were in fact benign and 51204 were correctly classified as benign. On the same HDRLM with LSTM and CNN the confusion matrix presented in Figure 4.18 for feature subset 4 shows that out of 28298 DDoS attack instances 27601 were correctly

classified to be DDoS attacks and 689 were misclassified as benign whereas out of 52302 benign instances 51516 were correctly classified as benign but 786 were misclassified as DDoS attack.

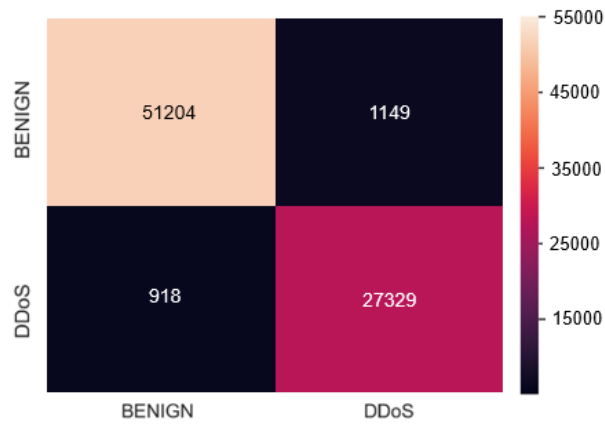


Figure 4.9: Confusion matrix for LSTM+CNN evaluated with FS3

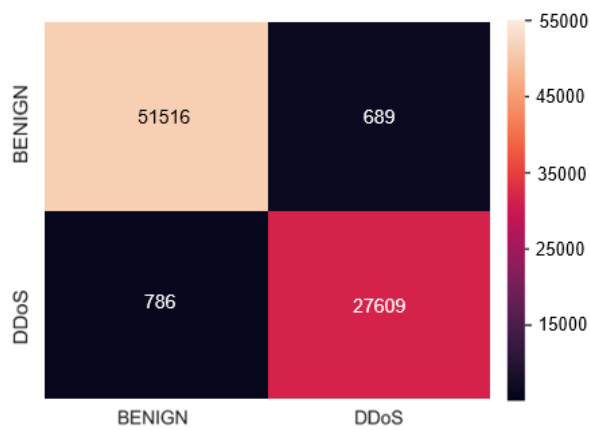


Figure 4.10: Confusion matrix for LSTM+CNN evaluated with FS4

Based on the confusion matrix for feature subset 3 and feature subset 4 on evaluating the HDRLM with a combination of LSTM and CNN for value approximation, the calculated False Positive Rate is presented as follows;

Feature subset 3

$$FPR = \frac{1149}{27329 + 1149}$$

$$FPR = 0.04$$

Feature subset 4

$$FPR = \frac{689}{27609 + 689}$$

$$FPR = 0.02$$

The results indicate that using feature subset 3 to evaluate it the false positive rate was 0.04 while when feature subset 4 was used the false positive rate was at 0.02.

In the second permutation using LSTM and DNN for value approximation, the HDRLM was evaluated with feature subset 3 and obtained results presented in Figure 4.19 is the confusion matrix in which out of 29397 instances of DDoS attack, 28419 were correctly classified to be DDoS attacks and 978 were incorrectly classified as benign. Out of 51203 instances that were benign 50236 were correctly classified as benign and 967 benign instances were misclassified as DDoS attacks. On the same permutation of LSTM and DNN feature subset 4 was used to evaluate the HDRLM and Figure 4.12 presents the confusion matrix obtained. The result show that 36252 DDoS attack instances that were sampled from the CICIDS2017 dataset, 20626 were correctly classified as DDoS attack with 12676 misclassified as benign. From a total of 47298 benign instances 34401 were correctly classified as benign while 12897 were misclassified as DDoS attack.

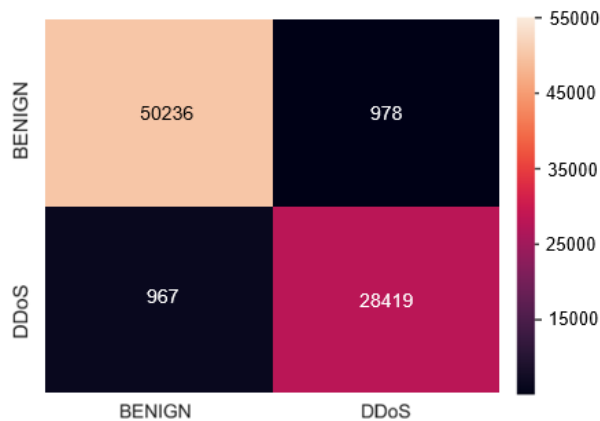


Figure 4.11: Confusion matrix for LSTM+DNN evaluated with FS3

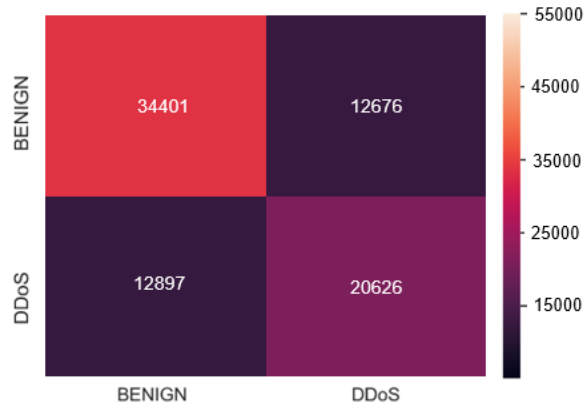


Figure 4.12: Confusion matrix for LSTM+DNN evaluated with FS4

The results obtained in the confusion matrix for feature subset 3 and feature subset 4 were applied to the FPOR formulae and the calculated FPR were as follows:

Feature subset 3:

$$FPR = \frac{978}{28419 + 978}$$

$$FPR = 0.03$$

Feature subset 4

$$FPR = \frac{12676}{20626 + 12676}$$

$$FPR = 0.38$$

When the HDRLM was evaluated with the LSTM and DNN for value approximation, the calculated false positive rate for feature subset 3 was 0.03 and 0.4 for feature subset 4.

The third permutation, the HDRL mechanism had CNN and DNN as a neural network for value approximation and the performance was first evaluated with feature subset 3. Figure 4.13 presents the confusion matrix from evaluating with feature subset 3 which shows that out of 28616 DDoS attack instances 27809 were classified to be DDoS attack which was a correct classification while 807 were misclassified to be benign when they were DDoS attacks. The mechanism also

correctly classified 51245 correctly to be benign out of a total of 51984 instances that were benign and the 739 were misclassified as DDoS attack. Using feature subset 4 to evaluate the HDRLM on the same permutation, the results (Figure 4.14) show that the mechanism correctly classified 27599 DDoS attacks from possible 28286 DDoS attack instances with the 687 DDoS attack instances misclassified as benign whereas out of 52314 benign instances, 51518 instances were correctly classified as benign with 796 benign instances misclassified as DDoS attack.

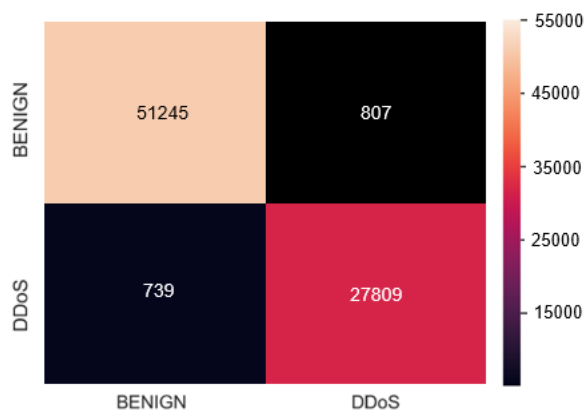


Figure 4.13: Confusion matrix for CNN+DNN evaluated with FS3

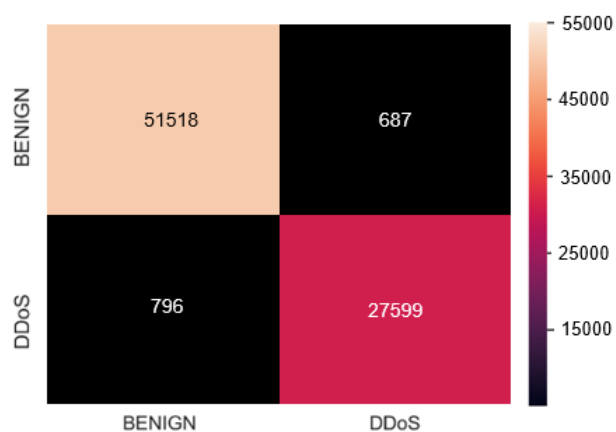


Figure 4.14: Confusion matrix for CNN+DNN evaluated with FS4

From the results obtained when evaluating the HDRLM with feature subset 3 and feature subset 4 with the value approximation component having a combination of CNN and DNN, the calculated false positive rates from the confusion matrix are as follows:

Feature subset 3:

$$FPR = \frac{807}{27809 + 807}$$

$$FPR = 0.03$$

Feature Subset 4:

$$FPR = \frac{687}{27599 + 687}$$

$$FPR = 0.02$$

The calculated false positive rate obtained for feature subset 3 was 0.03 and false positive rate for feature subset 4 was 0.02.

Table 4.6 summarises the results for the three permutations with feature subset 3 and feature subset 4 used to evaluate the performance of the HDRL mechanism based on the false positive rate parameter. From the results obtained, the HDRL mechanism performed well when the combination of CNN and DNN was used compared to the LSTM and CNN and LSTM and DNN. When the mechanism had a combination of CNN and DNN and was evaluated using feature subset 3 the FPR obtained was 0.03. Even when the HDRL used a combination of LSTM and DNN and was also evaluated using feature subset 3, FPR was also at 0.03. However, when feature subset 4 was used, the mechanism obtained an FPR of 0.02 with a combination of CNN and DNN against a false positive rate of 0.38 with a combination of LSTM and DNN.

Table 4.6: Hybrid Deep Reinforcement Learning, false positive rate computed results

|          | Feature subset 3 | Feature Subset 4 |
|----------|------------------|------------------|
| LSTM+CNN | 0.04             | 0.02             |
| LSTM+DNN | 0.03             | 0.38             |
| CNN+DNN  | 0.03             | 0.02             |

The results also show that the HDRLM performed well when a combination of LSTM and CNN was used and obtained a false positive rate of 0.02 when evaluated with feature subset 4 which is the same value obtained with a combination of CNN and DNN. However, when feature subset 3 was used to evaluate the mechanism, the performance of the HDRL mechanism was good with a combination of CNN and DNN obtaining a false positive rate of 0.03 compared to a false positive rate of 0.04 obtained when the combination of LSTM and CNN was used for value approximation in the mechanism.

#### 4.7.11 Average Reward

The average reward is an integral element in evaluating a reinforcement learning mechanism, it proves if there is any learning taking place by calculating the number of rewards obtained against the number of steps an agent takes in the environment (Andrew, 1998). The Average reward was used to establish if the mechanism performed as a Reinforcement Learning agent where it was supposed to improve on the rewards gained over time. This was achieved through learning the environment. This is the calculated total number of positive rewards received over the number of episodes taken. Figure 5.27 shows a positive trajectory of the Average reward signifying that the mechanism was able to learn and improve its performance. Figure 4.15 shows that as the number of episodes increase the average reward was also increasing.

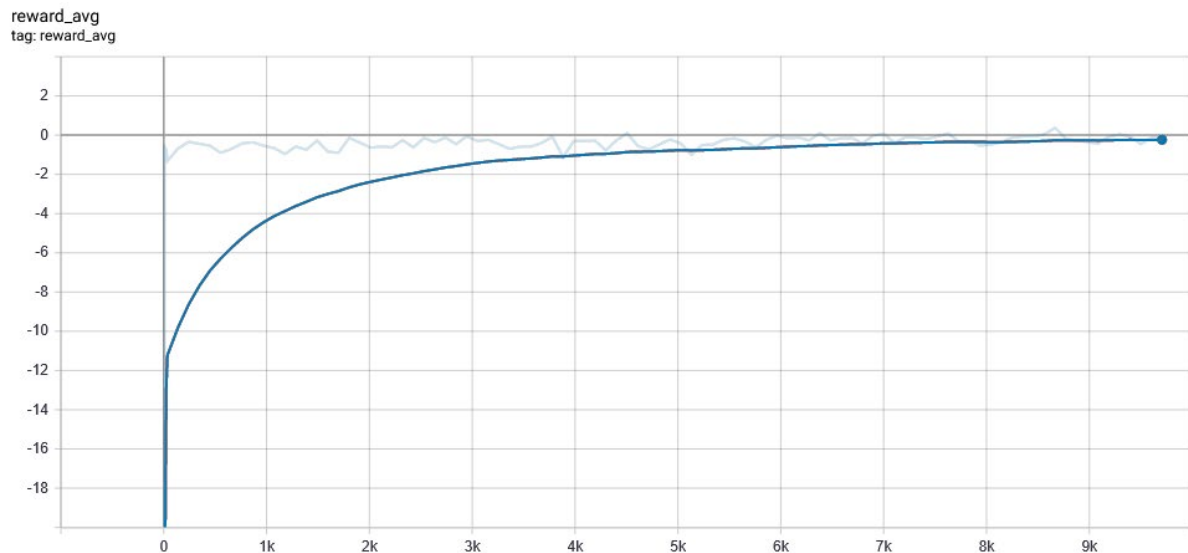


Figure 4.15: HDRL Average Reward

#### 4.7.12 CPU usage

For each of the three permutations, the CPU usage was monitored and it was noted that there was no change when either feature subset 3 or feature subset 4 was used in the evaluation. The parameters were recorded before running the process and during the process. The windows interface or dashboard on CPU and memory usage were used to monitor and record the results on the CPU usage. It was noted that the windows machine already has system applications and other non-system applications which are always running and do consume CPU and memory. This necessitated the need to have only key system applications running and kill all non-essential applications so that the CPU is at baseline before the running of the experiment. However, there is no literature available on Windows system application baseline. Therefore the baseline was subject to the knowledge and skill of the researcher which is at an intermediary level on the windows systems.

All other applications and processes were killed except Python, Firefox web browser on which the Jupyter runs and the windows resource monitoring tool. All network connections were also disabled. The system was at a baseline state with a CPU usage of 2%. Figure 4.16 shows the recorded status of the CPU for 60 seconds before the running of the experiment.

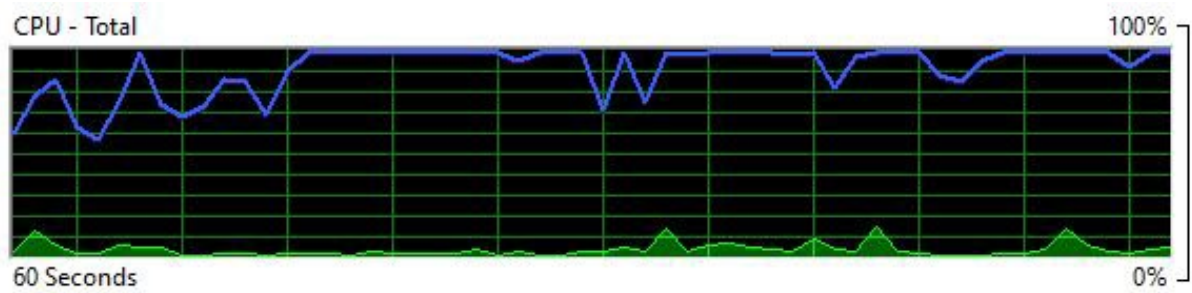


Figure 4.16: CPU status before the experiment

When the experiment for detection was started, the CPU usage started to rise and an average was recorded at 5000 episodes during which the mechanism was training the agent and at 7500 episodes when the agent had learned much of the environment and required minimal resources to perform detection. Following are the CPU usages for the three permutations of LSTM+CNN, LSTM+DNN, and CNN+DNN.

#### *LSTM+CNN*

When built with the combination of LSTM and CNN for the neural network, the mechanism i raises the CPU usage in episode 5000 from the baseline of 2% to 51% (Figure 4.17) representing a 49% rise.

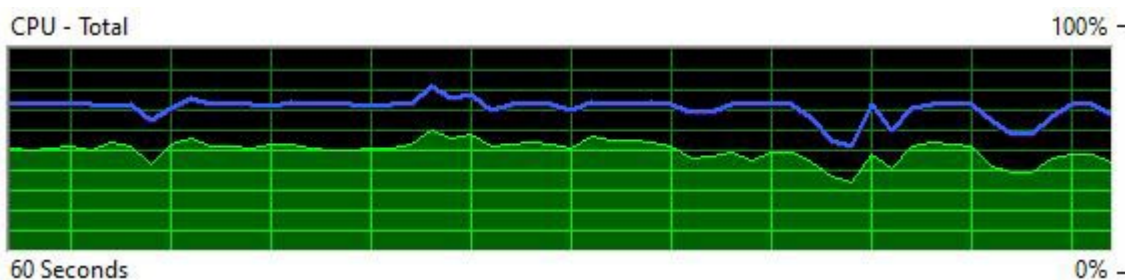


Figure 4.17: CPU usage during training with LSTM+CNN

The CPU usage dropped and remained at an average of 20% (Figure 4.18) when 7500 episodes were reached. This usage remained until the end of the process.

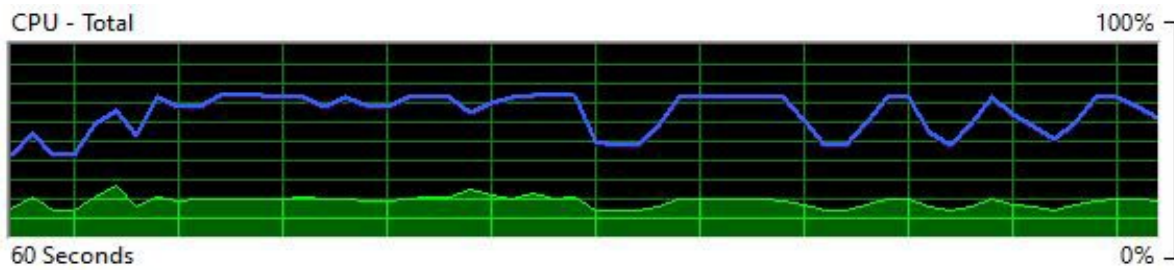


Figure 4.18: CPU usage during testing with LSTM+CNN

#### LSTM+DNN

When the experiment was run with the LSTM and DNN as the combination forming the neural network for value approximation, the CPU usage that was initially at 2% at the beginning of the process had risen and at 50000 episodes it had reached an average of the 42% which represented a 40% increase in CPU usage. Figure 4.19 shows an average usage of 30% in the 60 seconds of recording.

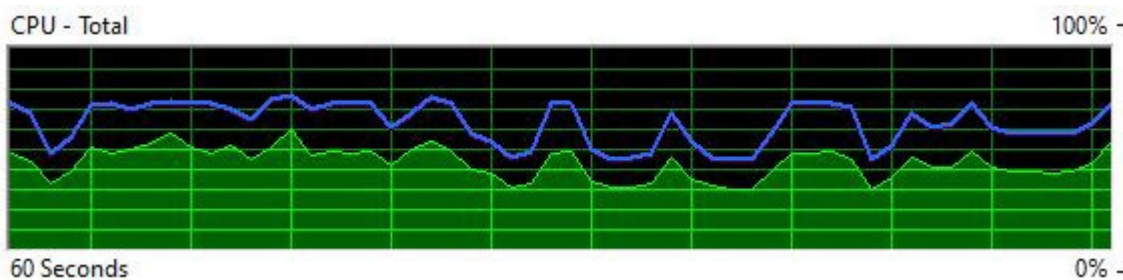


Figure 4.19: CPU usage during training with LSTM+DNN

As the process got to episode 7500 the usage had dropped to 13%. The graph in Figure 4.20 shows that the usage had stabilised and remained constant for the remainder of the experiment.

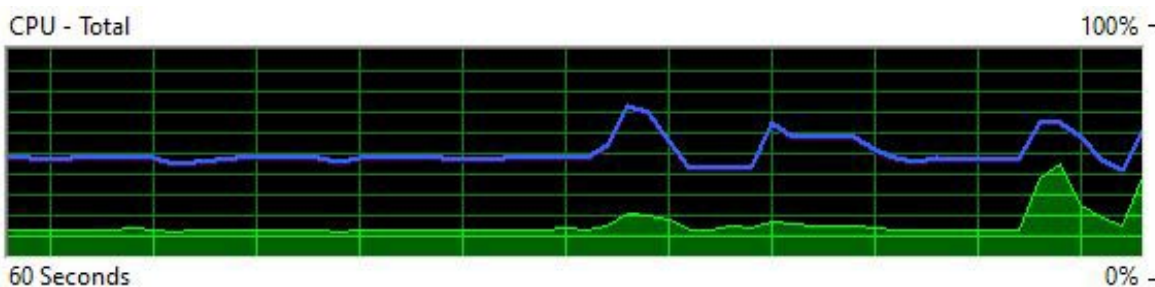


Figure 4.20: CPU usage during testing with LSTM+DNN

## CNN+DNN

The final permutation of CNN and DNN had the CPU usage rising to an average of 38% shown in Figure 4.21 from a baseline of 2%. The 36% rise was recorded at 5000 episodes that were during the training of the agent. Figure 4.22 shows a CPU usage dropped to 8% at 7500 episodes until the end of all 9000 episodes.

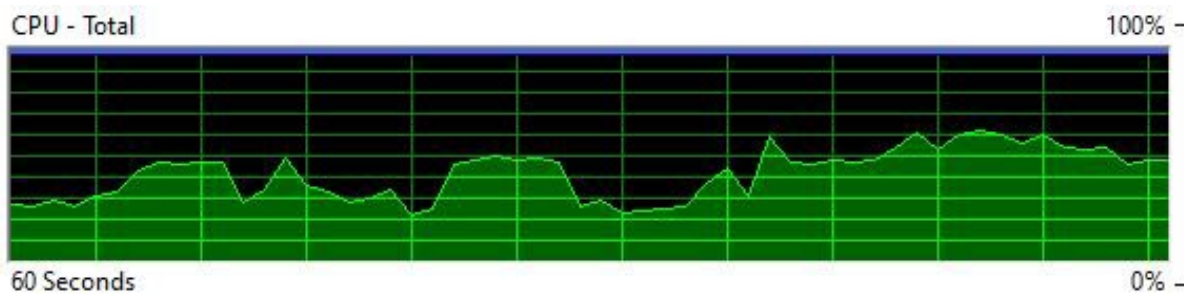


Figure 4.21: CPU usage during training with CNN+DNN

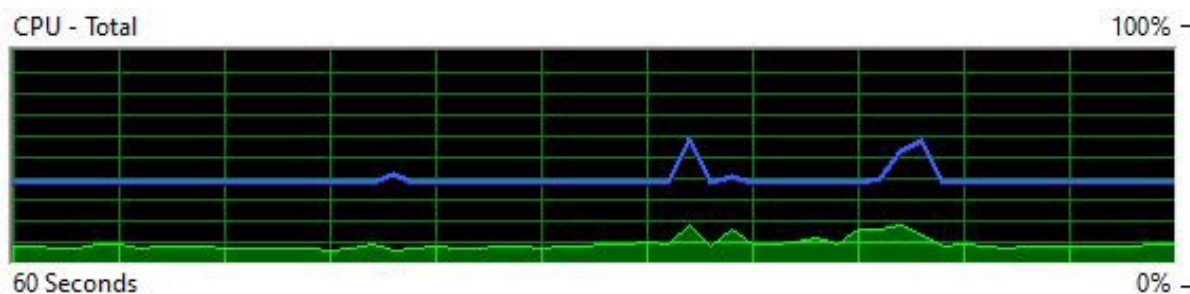


Figure 4.22: CPU usage during testing with CNN+DNN

CNN+DNN at 8%

### 4.7.13 Memory Usage

The memory usage was monitored for all the experiments and there was no variation in the memory usage between the different combinations. The memory usage before the experimentation was at a baseline of 21%. When the HDRL mechanism was initiated in the detection of DDoS attacks the memory usage rose from the 21% presented in Figure 4.23 to 23% presented in Figure 4.24, representing a 2% rise in memory usage.

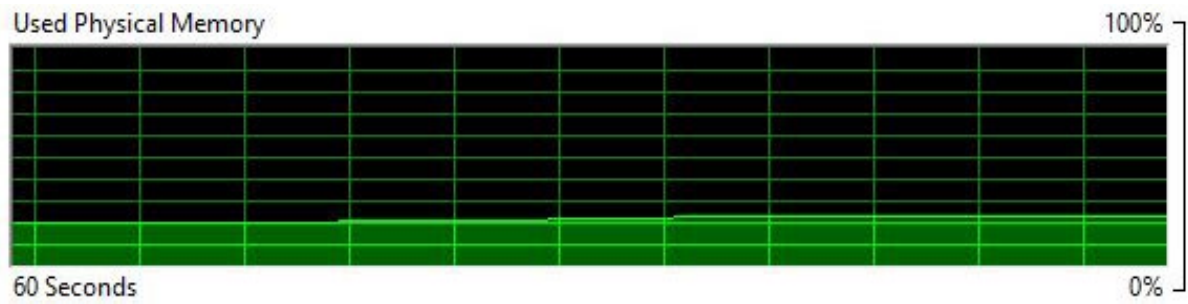


Figure 4.23: Memory Usage before detection experimentation

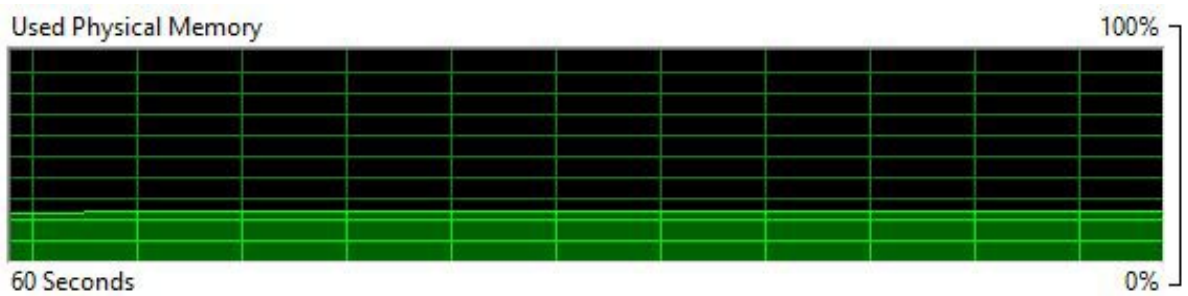


Figure 4.24: Memory Usage during detection experimentation

#### 4.8 Analysis Based on Parameters

Based on the evaluation from this study, the HDRLM showed the capability to detect DDoS attacks efficiently in an SDN environment. The study used the CICIDS 2017 DDoS dataset to evaluate the HDRLM.

Table 4.7: Summarised HDRL CPU usage

|          | CPU Usage Baseline (Before experiment) | CPU Usage (%) -During Training | CPU Usage (%) During Testing |
|----------|----------------------------------------|--------------------------------|------------------------------|
| LSTM+CNN | 2                                      | 51                             | 20                           |
| LSTM+DNN | 2                                      | 42                             | 30                           |
| CNN+DNN  | 2                                      | 38                             | 8                            |

Table 4.8: Summarised HDRL memory usage

|          | Memory Usage Baseline (Before experiment) | Memory Usage (%) -During Training and Testing |
|----------|-------------------------------------------|-----------------------------------------------|
| LSTM+CNN | 21                                        | 23                                            |
| LSTM+DNN | 21                                        | 23                                            |
| CNN+DNN  | 21                                        | 23                                            |

The comparative analysis in Table 4.7 shows that the mechanism performed more efficiently on CPU usage when the value approximation of the HDRLM with a combining CNN and DNN for value approximation. The combination of CNN and DNN achieved an acceptable accuracy of 98.16% and a False Positive Rate of 0.02 while the CPU usage presented in Table 4.7 was the lowest at 8% compared to 20% when a combination of LSTN and CNN was used and a 30% when LSTM and DNN were used.

#### 4.9 Validation

The study considered other state of the art mechanisms for a comparative validation of the HDRLM performance. Three mechanisms designed either using Deep Reinforcement Learning algorithms or similar algorithms for DDoS attack detection in Software Defined Networks were identified for the validation. In comparison, the HDRLM showed that it had improved on both effectiveness and efficiency.

*Table 4.9:Comparative validation results of the HDRLM*

| <b>Mechanism</b>           | <b>Algorithm Used</b> | <b>Accuracy</b> | <b>CPU usage</b> | <b>Memory Usage</b> |
|----------------------------|-----------------------|-----------------|------------------|---------------------|
| HDRLM (Proposed)           | Hybrid DQN            | 98.16%          | 8%               | 2%                  |
| Haider et al. (2019)       | Ensemble (CNN+CNN)    | 99.48%          | 8.46%            | 8%                  |
| Mousavi, et al. (2014)     | SOM                   | 96%             | 16%              | 30%                 |
| Hossen and Janagam, (2018) | DQN                   | 95%             |                  |                     |

Table 4.9 shows that the HDRLM when compared with another mechanism proposed by Haider et al. (2019), that uses an Ensemble neural network for DDoS attack detection in software Defined Network and obtained 99.48%, the Deep Reinforcement Learning mechanism obtained a lower accuracy but improved the CPU usage by achieving a 8% from 8.46% and a Memory usage of 2% from 8%. Furthermore, another comparison with Mousavi, et al. (2014) showed that the HDRLM had improved on accuracy 98.16% against 96%,CPU usage 8% against 16% and Memory usage 2% against 30%.

On effectiveness the validation considered a Deep Reinforcement Learning mechanism proposed by Hossen and Janagam, (2018). Results presented in Table 4.9 show that the HDRLM achieved a higher accuracy of 98.16 as compared to 95% obtained in detection of DDoS attacks by Hossen and Janagam, (2018).

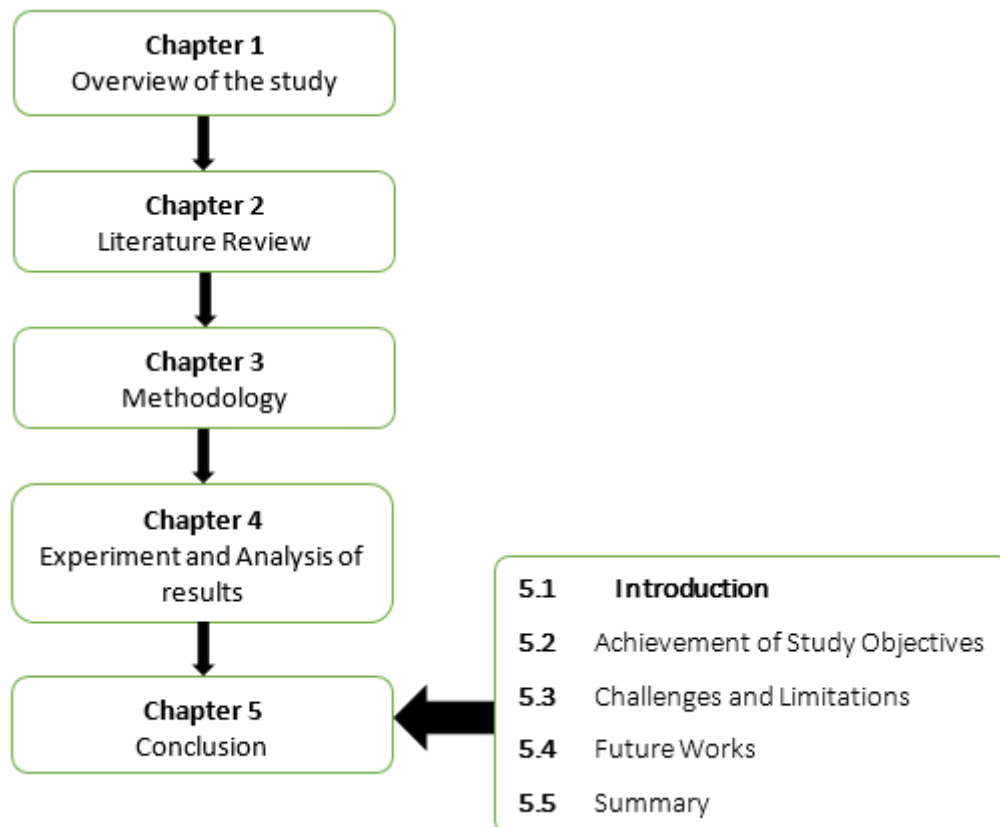
The study has demonstrated that the HDRLM when compared with other state of the art mechanisms has improved in achieving a high detection accuracy of 98.16% while using minimum computational resources.

#### 4.10 Summary

The evaluation of the designed Deep Reinforcement Learning detection mechanism for DDoS attacks in SDN using the CICIDS 2017 dataset provided an empirical basis to ascertain the reliability of the mechanism. By evaluating the effectiveness and the efficiency of the HDRL mechanism, the study proves the utility of the mechanism. There are several parameters available to evaluate a detection mechanism depending on what the study objectives are. This study used five parameters namely accuracy, false positive rate, average reward, CPU usage, and Memory usage to evaluate the mechanism accuracy with several. Accuracy, false positive rate, and average reward were used to evaluate the effectiveness of the mechanism. The computer CPU and Memory are critical resources when it comes to computing and performance of any task. Therefore, the use of these critical resources is to be considered when deciding at they come at a cost. By observing the usage of the CPU and memory the study established the efficiency of the designed HDRL detection mechanism. The mechanism achieved an accuracy of 98.16% with a false positive rate of 0.02 and demonstrated the ability to learn by itself by gaining average reward rising from -12 to 0. This was achieved with a CPU usage of 8% and memory usage of 2%. This is considered a good performance when validated with close related mechanisms.

## Chapter 5 : Conclusion

This chapter presents a summary of the study and outlines how the objectives of the study have been achieved. It also highlights some of the challenges faced during the study and proposes areas for further study. The chapter is outlined as in the chapter map below.



### 5.1 Introduction

The study designed an efficient Hybrid Deep Reinforcement Learning mechanism (HDRLM) for detecting DDoS attacks in Software Defined networks that attained the highest accuracy of 98.16%, the lowest CPU usage of 8%, and memory usage of 2% compared to other mechanisms. The mechanism was designed with two heterogeneous neural networks for value approximation. Three types of neural networks namely Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), and Deep Neural Networks (DNN) were identified to have higher performance (Muhammad Diyan , Murad Khan, 2020) and were therefore considered in the design. Using a combination of two of the three neural networks, three permutations were subjected to experimentation to establish

the best performing combination that was used in the mechanism to achieve the highest accuracy and efficiency.

To evaluate the mechanism, the study used the CICIDS 2017 dataset which has DDoS attacks. From the literature four feature subsets were identified and a fifth feature subset was identified through experimentation using the Weka application. The CICIDS2017 dataset was first pre-processed and then the ensemble-based multi-filter feature selection (EMFFS) approach (Osanaiye et al., 2016), was used to obtain a feature subset.

*Table 5.1: Feature subsets from Literature and experimentation*

|    | Feature subset   | Features                                                                                                                                                                      |
|----|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. | Feature subset 1 | Total Length of Fwd Packets, Total Length of Bwd Packets, Bwd Packet Length Mean, Avg Bwd Segment Size, Subflow Fwd Bytes, Subflow Bwd Bytes                                  |
| 2. | Feature subset 2 | Fwd.IAT.Total, Flow.IAT.Max, Active.Max, Active.Min, Fwd.IAT.Max, Active.Mean, Idle.Min, Idle.Max, Flow.IAT.Std, Fwd.IAT.Mean                                                 |
| 3. | Feature subset 3 | Bwd packet length Std, Avg. packet size, flow duration, Flow IAT Std                                                                                                          |
| 4. | Feature subset 4 | Destination Port, Total Length of Fwd Packets, Total Length of Bwd Packets, Fwd packets Length Min, ACK flag Count, Init_Win_bytes_forward, Init_Win_bytes_backward, Idle Min |
| 5. | Feature subset 5 | Destination port, Flow IAT Std, Idle Min                                                                                                                                      |

Table 5.1 presents a summary of the five feature subsets obtained from literature and experimentation. The five feature subsets were evaluated on their performance with three permutations of the HDRLM that had a combination of either LSTM and CNN or LSTM and DNN or CNN and DNN in the algorithm for value approximation. Two parameters namely accuracy and false positive rate (FPR) were used to identify the best feature subset among the five. Feature subset 4 (made of the following features: *Destination Port, Total Length of Fwd Packets, Total Length of Bwd Packets, Fwd packets Length Min, ACK flag Count, Init\_Win\_bytes\_forward, Init\_Win\_bytes\_backward, Idle Min*), was the best with the highest accuracy and lowest FPR obtained.

In the evaluation of the performance of the HDRLM, five parameters were used namely Accuracy, False positive rate, Average Reward, CPU usage, and Memory Usage. An analysis based on the parameters was done from the results of the experiment to establish the performance of the designed mechanism.

Other researchers have proposed similar mechanisms in their works before this study. We considered results that were obtained in their proposed mechanism in comparison to what was obtained from the HDRLM. The comparison validated that despite not making significant improvements on the accuracy, the HDRLM had improved on CPU usage and memory usage and can be considered to be an efficient detection mechanism for DDoS attacks in SDN.

With the Software Defined Network architecture which has the control plane as the central decision point for the network, it provides a single point of failure. Therefore, this mechanism was designed to be placed on the control plane of the SDN to reduce the risk that DDoS attacks pose to the SDN.

Apart from proving to be accurate and more efficient, the use of this Hybrid Deep Reinforcement Learning for DDoS attack detection provides a potential future for intrusion detection systems with the ability to learn the environment by use of the reward functionality.

## 5.2 Achievements of Study Objectives

The main objective of the study was to design an efficient Hybrid Deep Reinforcement Learning mechanism for the detection of DDoS attacks in Software Defined Networks which was achieved and demonstrated through experimentation. To arrive at the main objective three other objectives were first achieved as detailed in the following section:

1. **To identify and evaluate the best feature selection technique and best features for DDoS attack detection from available datasets to achieve high classification accuracy in Software Defined Networks.**

The study evaluated available feature selection techniques from the literature that has produced the best results. Apart from the techniques, the actual feature subsets identified in the literature were adopted as candidate feature subsets as presented in Section 3.2.8. The five feature subsets were extracted from the CICIDS2017 and are presented in Table 5.1. To identify the most appropriate dataset from the many available, an analysis made in Section 2.7 of the available

datasets was first conducted that considered the commonly used datasets in similar research work in the past five years. Five datasets namely KDD CUP 99, NSL-KDD, CAIDA, and CICIDS2017 were identified from literature and using a framework called an Evaluation framework for IDS and IPS datasets proposed by Gharib et al., (2017) The CICIDS2017 dataset emerged as the best dataset to use as presented in Section 2.8. From the literature, four feature subsets, shown in Table 3.4, were identified and a fifth feature subset presented in Table 3.3, was obtained from an experiment that applied the EMFFS approach as identified to be the best feature selection technique from literature. The five feature subsets were evaluated in an experiment that identified the best feature subset as presented in Section 4.6.8.

## **2. Determine and evaluate efficient classifiers based on deep reinforcement learning**

Many classifiers have different capabilities due to their inherent structure. To identify the most efficient classifiers that best suit a deep reinforcement learning mechanism a triangulation approach was used where two approaches namely literature and experimentation were applied. The literature was used to inform the experimentation. In the literature, three classifiers namely Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), and Deep Neural Network (DNN) were identified as the most efficient and high-performance classifiers. To maximise the capabilities in the classifiers, a hybrid approach that combined two classifiers, CNN and DNN, was adopted. To identify the best combination, the study used an experiment as presented in Section 4.3 in which three permutations that contain two of the three classifiers were evaluated. Section 4.4 presents parameters used to identify the best combination. The combination of CNN and DNN emerged as the best performing in terms of effectiveness with 98.16% detection accuracy, 0.02 false positive rate and apposite average reward trajectory and efficiency obtained an 8% CPU usage during detection and 2% of Memory usage. Therefore, the combination of CNN and DNN was adopted in the design of the proposed mechanism.

**3. Design an efficient hybrid DDoS attack detection mechanism that will require minimal or no retraining to detect new attacks for SDN.**

The study used a design science approach to design the HDRLM. Full details are presented in Chapter 3, Section 3.2. The HDRLM presented in Section 3.3.13 is designed following the concept of the Markov Decision Process where an agent interacts with the environment through observing the state of the environment and taking an action based on the state. Thereafter the agent receives a reward based on the action taken in a given state. The HDRLM has the agent as the classification component and performs classification by observing the environment in this case the feature subset, through the states presented as instances, and determining if the state presented is either a DDoS attack or a benign. The classification component of the HDRLM is not trained with already labelled data but learns from experience which is a collection of four elements namely state, action, reward and next state. For every action the classification component took, the aim was to gain the maximum reward possible. Therefore, for the classification component to decide what action to take in had to refer to a value for every action. Instead of maintaining a table of the values the study, combined two heterogeneous neural networks to form a hybrid value approximator. The neural network component used CNN and DNN that were identified in Section 3.3.4 and were further evaluated to achieve the second objective presented in Section 4.7.

The study in addition developed a reward strategy presented in Section 4.2.3, that was used by the classification component. The reward strategy was designed using the labels of the feature subsets from the CICIDS2017.

The HDRLM is capable to learn by itself without prior training and can be deployed in an environment immediately and once it reaches optimal levels of its performance it can detect new DDoS attacks. However, the HDRLM falls short in that it requires more to reach optimal performance because it learns in the environment, unlike mechanisms that are trained with already labelled data. In addition, the concept of learning through trial and error makes it a disadvantage for the HDRLM

adoption in a business industry that strives at reducing the risks. The shortfalls can be dealt with by subjecting the HDRLM to high fidelity simulated environments.

### 5.3 Challenges and Limitation

The study had a limitation by only focusing on DDoS attacks as a single type of attack despite several attacks that the SDN is vulnerable to. Due to the concern on network security and user privacy, the experiment was limited to a simulated environment.

The tools used in the experiment were largely open-source due to budgetary limitations. However, this did not compromise the quality of the study as the best tools were selected among those that have been used in other research and produced state-of-the-art results and were developed specifically for research and by research institutions. To use these tools more time was used to configure them to meet the needs of the study and create the best environment. This challenge of using open source tools can be overcome by budget support to minimise the time taken to configure the tools for the study.

The challenge coming due to network security limiting the evaluation of the performance of HDRLM in a real network can be dealt with by creating high fidelity simulations. High fidelity simulations used in medicine and the military (Jyothi et al., 2016) have proven to be more close to real-world scenarios.

### 5.4 Future work

This study did not completely exhaust all the work on the HDRLM. Therefore, there are still more research areas that can improve the HDRLM and the detection of DDoS in SDN. Some of the areas to be considered for future research are:

- How to improve the reward function of the HDRL mechanism that would enhance its learning capability.
- Using the HDRLM to detect other forms of attacks on the SDN.

- Establishing the best placement of the detection mechanism in the SDN architecture as the control plane is already overloaded with decision making.
- Evaluate the HDRL in a High-Performance Computing environment to ascertain the impact of the environment on the performance on the HDRM.

## 5.5 Conclusion

This study designed an efficient hybrid Deep Reinforcement Learning for DDoS attacks detection in the Software Defined Networks (SDN). The emergence of the SDN has brought several benefits into the network architecture. However, with it also comes challenges such as a single point of failure due to its centralised architecture. With DDoS attacks still growing in complexity and increasing in incidents, the SDN faces a great threat, especially on its controller. The designed DDoS attack detection mechanism using Deep reinforcement learning is placed on the control plane of the SDN to address the problem of a single point of failure.

Design science was used as an approach in the designing of the mechanism shown in Figure 3.12 . A literature review was conducted on previous work dealing with the problem of DDoS attacks. This was followed by an experiment in which the CICIDS2017 dataset was used to evaluate the performance of the HDRL mechanism in a simulated gym environment. The evaluation was based on five parameters namely; accuracy, false positive rate, Average reward, CPU usage and, memory usage. An accuracy of 98.16% was achieved with a false positive rate of 0.02 and a positive average reward trajectory, these three parameters prove that the mechanism is effective and can perform detection. The CPU usage was at 8% while the memory usage was at 2%, and when compared with other mechanisms, the CPU usage, and memory usage have shown that the mechanism is efficient in usage of the computation resources while performing the detection.

Therefore, this mechanism is suggested to be placed at the control plane of the SDN for the efficient and effective detection of DDoS attacks on the SDN and can be a basis for other mitigation interventions.

## References

- Abdelaziz, A., Fong, A. T., Gani, A., Garba, U., Khan, S. S. U., Akhonzada, A., Talebian, H., Choo, K. K. R., Gani, A., Anuar, N. B., Abdelaziz, A., Khan, M. K., Hayat, A., & Khan, S. S. U. (2017). Secure and dependable software defined networks. *Journal of Network and Computer Applications*, 61(4), 1–19. <https://doi.org/10.1016/j.jnca.2015.11.012>
- Abdulraheem, M. H., & Ibraheem, N. B. (2019). A detailed analysis of new intrusion detection dataset. *Journal of Theoretical and Applied Information Technology*, 97(17), 4519–4537.
- Abdulrahman, A. A., & Ibrahem, M. K. (2019). Evaluation of DDoS attacks Detection in a New Intrusion Dataset Based on Classification Algorithms. *Iraqi Journal of Information & Communications Technology*, 1(3), 49–55. <https://doi.org/10.31987/ijict.1.3.40>
- Abubakar, A., & Pranggono, B. (2017). Machine learning based intrusion detection system for software defined networks. *Proceedings - 2017 7th International Conference on Emerging Security Technologies, EST 2017*, 138–143. <https://doi.org/10.1109/EST.2017.8090413>
- Ajaeiya, G. A., Adalian, N., Elhajj, I. H., Kayssi, A., & Chehab, A. (2017). Flow-based Intrusion Detection System for SDN. *Proceedings - IEEE Symposium on Computers and Communications, July*, 787–793. <https://doi.org/10.1109/ISCC.2017.8024623>
- Akhonzada, A., Ahmed, E., Gani, A., Khan, M. K., Imran, M., Guizani, S., Anuar, N. B., Abdelaziz, A., Khan, M. K., Hayat, A., & Khan, S. U. (2016). Secure and dependable software defined networks. *Journal of Network and Computer Applications*, 61, 199–221. <https://doi.org/10.1016/j.jnca.2015.11.012>
- Alansary, A., Oktay, O., Li, Y., Folgoc, L. Le, Hou, B., Vaillant, G., Kamnitsas, K., Vlontzos, A., Glocker, B., Kainz, B., & Rueckert, D. (2019). Evaluating reinforcement learning agents for anatomical landmark detection. *Medical Image Analysis*, 53(Midl), 156–164. <https://doi.org/10.1016/j.media.2019.02.007>
- Alkasassbeh, M., Al-Naymat, G., B.A, A., & Almseidin, M. (2016). Detecting Distributed Denial of Service Attacks Using Data Mining Techniques. *International Journal of Advanced Computer Science and Applications*, 7(1), 436–445. <https://doi.org/10.14569/ijacsa.2016.070159>
- Alrawashdeh, K., & Purdy, C. (2016). Toward an Online Anomaly Intrusion Detection System Based on Deep Learning. *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 195–200. <https://doi.org/10.1109/ICMLA.2016.0040>
- Alsmadi, I., & Xu, D. (2015). Security of Software Defined Networks: A survey. *Computers and Security*, 53, 79–108. <https://doi.org/10.1016/j.cose.2015.05.006>
- Alyasiri, H. (2018). *Developing Efficient and Effective Intrusion Detection System using Evolutionary Computation Hasanen Alyasiri*. November.
- Andrew, A. M. (1998). Reinforcement Learning: An Introduction. *Kybernetes*, 27(9), 1093–1096. <https://doi.org/10.1108/k.1998.27.9.1093.3>
- Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6), 26–38. <https://doi.org/10.1109/MSP.2017.2743240>
- Arun Raj Kumar, P., & Selvakumar, S. (2013). Detection of distributed denial of service attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems. *Computer Communications*, 36(3), 303–319. <https://doi.org/10.1016/j.comcom.2012.09.010>

- Arunraj, R., & Umarani, C. (2017). *A Study on Applying Reinforcement Learning in Intrusion Detection and Prevention System*. 6–8.
- Aslansefat, K., Sorokos, I., Whiting, D., Tavakoli Kolagari, R., & Papadopoulos, Y. (2020). SafeML: Safety Monitoring of Machine Learning Classifiers Through Statistical Difference Measures. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12297 LNCS, 197–211. [https://doi.org/10.1007/978-3-030-58920-2\\_13](https://doi.org/10.1007/978-3-030-58920-2_13)
- Balkanli, E., Alves, J., & Zincir-Heywood, A. N. (2014). Supervised learning to detect DDoS attacks. *IEEE SSCI 2014: 2014 IEEE Symposium Series on Computational Intelligence - CICS 2014: 2014 IEEE Symposium on Computational Intelligence in Cyber Security, Proceedings*, 1–8. <https://doi.org/10.1109/CICYBS.2014.7013367>
- Barati, M., Abdullah, A., Udzir, N. I., Mahmod, R., & Mustapha, N. (2015). Distributed Denial of Service detection using hybrid machine learning technique. *Proceedings - 2014 International Symposium on Biometrics and Security Technologies, ISBAST 2014*, 268–273. <https://doi.org/10.1109/ISBAST.2014.7013133>
- Barbhuiya, F. A. (2014). *Design and Development of Intrusion Detection System : A Discrete Event System Approach*. April.
- Batta, M. (2020). Machine Learning Algorithms - A Review . *International Journal of Science and Research (IJ, 9(1)*, 381-undefined. <https://doi.org/10.21275/ART20203995>
- Behal, S., & Kumar, K. (2016). Trends in Validation of DDoS Research. *Procedia Computer Science*, 85(Cms), 7–15. <https://doi.org/10.1016/j.procs.2016.05.170>
- Benmessahel, I., Xie, K., & Chellal, M. (2018). New Improved Training for Deep Neural Networks Based on Intrusion Detection System. *IOP Conference Series: Materials Science and Engineering*, 435(1). <https://doi.org/10.1088/1757-899X/435/1/012034>
- Bhaya, W., & Manaa, M. E. (2014). Review clustering mechanisms of distributed denial of service attacks. *Journal of Computer Science*, 10(10), 2037–2046. <https://doi.org/10.3844/jcssp.2014.2037.2046>
- Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2015). An empirical evaluation of information metrics for low-rate and high-rate DDoS attack detection. *Pattern Recognition Letters*, 51, 1–7. <https://doi.org/10.1016/j.patrec.2014.07.019>
- Branch, N., & Azad, I. (2014). *m nl ad in e e V by e th rsio is n fil O e is nly Bo m nl ad in e e V by e th rsio is n fil O e is nly*. 7(2), 59–70.
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). *OpenAI Gym*. 1–4. <http://arxiv.org/abs/1606.01540>
- Cepheli, Ö., Büyükçorak, S., & Karabulut Kurt, G. (2016). Hybrid Intrusion Detection System for DDoS Attacks. *Journal of Electrical and Computer Engineering*, 2016, 1–8. <https://doi.org/10.1155/2016/1075648>
- Chilimbi, T., Suzue, Y., Apacible, J., & Kalyanaraman, K. (2014). Project ADAM: Building an efficient and scalable deep learning training system. *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2014*, 571–582.
- Chuan-long, Y., Yue-fei, Z., Jin-long, F., & Xin-zheng, H. (2017). A Deep Learning Approach for Intrusion Detection using Recurrent Neural Networks. *IEEE Access*, 5, 1–1. <https://doi.org/10.1109/ACCESS.2017.2762418>

- Clarke, R. J. (2005). Research Models and Methodology. *Research Methodologies*, 1–65.  
<https://doi.org/DOI.10.1016/j.febslet.2005.03.016>
- Dongsoo, L. (2015). *Improving Detection Capability of Flow-based IDS in SDN*.
- Dooley, K. (2002). Simulation Research Methods Kevin Dooley. *Arizona State University*, 829–848.
- Ewees, A. A., & Eleraky, R. M. (2019). Features Selection for Facial Expression Recognition. *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 1–6.
- Fan, J., Wang, Z., Xie, Y., & Yang, Z. (2019). *A Theoretical Analysis of Deep Q-Learning*.  
<http://arxiv.org/abs/1901.00137>
- Fedus, W., Ramachandran, P., Agarwal, R., Bengio, Y., Larochelle, H., Rowland, M., & Dabney, W. (2020). Revisiting fundamentals of experience replay. *37th International Conference on Machine Learning, ICML 2020, Part F16814*, 3042–3052.
- Fouladi, R. F., Kayatas, C. E., & Anarim, E. (2016). Frequency based DDoS attack detection approach using naive bayes classification. *2016 39th International Conference on Telecommunications and Signal Processing, TSP 2016*, 104–107. <https://doi.org/10.1109/TSP.2016.7760838>
- Gharaibeh, M., & Papadopoulos, C. (2009). *DARPA-2009 Intrusion Detection Dataset Report*. 1–14.
- Gharib, A., Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2017). An Evaluation Framework for Intrusion Detection Dataset. *ICISS 2016 - 2016 International Conference on Information Science and Security, Cic*, 1–6. <https://doi.org/10.1109/ICISSEC.2016.7885840>
- Ghiassian, S., Rafiee, B., Lo, Y. L., & White, A. (2020). Improving performance in reinforcement learning by breaking generalization in neural networks. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, 2020-May(1)*, 438–446.
- Gou, S. Z., & Liu, Y. (2019). *DQN with model-based exploration: efficient learning on environments with sparse rewards*. 1–10. <http://arxiv.org/abs/1903.09295>
- Gu, Y., Li, K., Guo, Z., & Wang, Y. (2019). Semi-supervised k-means ddos detection method using hybrid feature selection algorithm. *IEEE Access*, 7, 64351–64365.  
<https://doi.org/10.1109/ACCESS.2019.2917532>
- Haider, S., Akhunzada, A., Ahmed, G., & Raza, M. (2019). Deep Learning based Ensemble Convolutional Neural Network Solution for Distributed Denial of Service Detection in SDNs. *2019 UK/China Emerging Technologies, UCET 2019, November*, 1–4.  
<https://doi.org/10.1109/UCET.2019.8881856>
- Hamed, T., Dara, R., & Kremer, S. C. (2018). Network intrusion detection system based on recursive feature addition and bigram technique. *Computers and Security*, 73, 137–155.  
<https://doi.org/10.1016/j.cose.2017.10.011>
- Han, G., & Jiao, Y. (2017). Feature selection and regional labeling of significant detection for pulmonary nodules in CT images. *2016 IEEE International Conference on Signal and Image Processing, ICSIP 2016*, 60–63. <https://doi.org/10.1109/SIPROCESS.2016.7888224>
- Haq, N. F., Onik, A. R., Hridoy, M. A. K., Rafni, M., Shah, F. M., & Farid, D. M. (2015). Application of Machine Learning Approaches in Intrusion Detection System : A Survey. *International Journal of Advanced Research in Artificial Intelligence*, 4(3), 9–18.
- Hausknecht, M., & Stone, P. (2015). Deep recurrent q-learning for partially observable MDPs. *AAAI Fall Symposium - Technical Report, FS-15-06*, 29–37.

- Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Two Paradigms on Research Essay Design Science in Information Systems Research. *MIS Quarterly*, 28(1), 75–79.
- Hossen, S., & Janagam, A. (2018). *Analysis of Network Intrusion Detection System with Machine Learning Algorithms ( Deep Reinforcement Learning Algorithm )* Saddam Hossen. October.
- Hu, J., Wu, C., & Yang, M. (2020). Improving the learning efficiency of DQN by an unsupervised auxiliary task. *ICAART 2020 - Proceedings of the 12th International Conference on Agents and Artificial Intelligence*, 2, 889–894.
- Ibrahim, J. (2017). *SDN-Based Intrusion Detection System Literature review*. 16(March), 621–624.
- Jain, A. K. (2018). *Application of Intrusion Detection System in Automatic Evidence Collection using Digital Forensics*.
- Jeya, P Gifty, Ravichandran, M; Ravichandran, C. S. (2012). Efficient Classifier for R2L and U2R Attacks. *International Journal of Computer Applications (0975 – 8887)*, 45(21).
- Jyothi, V., Wang, X., Addepalli, S. K., & Karri, R. (2016). BRAIN: Behavior Based Adaptive Intrusion Detection in Networks: Using Hardware Performance Counters to Detect DDoS Attacks. *Proceedings of the IEEE International Conference on VLSI Design, 2016-March*, 587–588. <https://doi.org/10.1109/VLSID.2016.115>
- Kale, M., & Choudhari, D. M. (2014). DDoS Attack Detection Based on an Ensemble of Neural Classifier. *International Journal of Computer Science and Network Security*, 14(7), 122–129.
- Kalkan, K., Gur, G., & Alagoz, F. (2017). Defense Mechanisms against DDoS Attacks in SDN Environment. *IEEE Communications Magazine*, 55(9), 175–179. <https://doi.org/10.1109/MCOM.2017.1600970>
- Karatas, G., Demir, O., & Koray Sahingoz, O. (2019). Deep Learning in Intrusion Detection Systems. *Proceedings of the International Research Conference on Engineering and Technology*, 113–116. <https://doi.org/10.1109/ibigdelft.2018.8625278>
- Ketkar, N. (2017). Deep Learning with Python - A Hands-on Introduction. In *Deep Learning with Python*. <https://doi.org/10.1007/978-1-4842-2766-4>
- Kim, G., Lee, S., & Kim, S. (2014). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications*, 41(4 PART 2), 1690–1700. <https://doi.org/10.1016/j.eswa.2013.08.066>
- Kim, J., Shin, N., Jo, S. Y., & Kim, S. H. (2017). Method of intrusion detection using deep neural network. *2017 IEEE International Conference on Big Data and Smart Computing, BigComp 2017*, 313–316. <https://doi.org/10.1109/BIGCOMP.2017.7881684>
- Kokila, R. T., Thamarai Selvi, S., & Govindarajan, K. (2015). DDoS detection and analysis in SDN-based environment using support vector machine classifier. *6th International Conference on Advanced Computing, ICoAC 2014*, 205–210. <https://doi.org/10.1109/ICoAC.2014.7229711>
- Kreutz, D., Rothenberg, C. E., Azodolmolky, S., Uhlig, S., Verissimo, P., & Ramos, F. (2015). Software-Defined Networking : A Comprehensive Survey. *Proceedings of the IEEE*, 103(1), 14–76. <https://doi.org/10.1109/JPROC.2014.2371999>
- Kumar Ahuja, G., & Kumar, G. (2014). Evaluation Metrics for Intrusion Detection Systems-A Study. *International Journal of Computer Science and Mobile Applications*, 2(11), 11–17. <https://www.researchgate.net/publication/311108073>
- Lashkari, A. H., Gil, G. D., Mamun, M. S. I., & Ghorbani, A. A. (2017). Characterization of tor traffic

- using time based features. *ICISSP 2017 - Proceedings of the 3rd International Conference on Information Systems Security and Privacy, 2017-Janua(Cic)*, 253–262.  
<https://doi.org/10.5220/0006105602530262>
- Leavy, P. (2017). *Research Design-Quantitative, Qualitative, Mixed Methods, Arts-Based, and Community-Based Participatory Research Approaches*.
- Lima, R. F., & Pereira, A. C. M. (2016). A fraud detection model based on feature selection and undersampling applied to web payment systems. *Proceedings - 2015 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT 2015, Dcc*, 219–222. <https://doi.org/10.1109/WI-IAT.2015.13>
- Maidstone, R. (2012). Discrete Event Simulation, System Dynamics and Agent Based Simulation: Discussion and Comparison. *System, August*, 1–6.
- Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, M. K., & Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, G. B. (2016). TensorFlow: A System for Large-Scale Machine Learning. *Usenix:The Advanced Computing Sytems Association, 101(C)*, 582–598.  
[https://doi.org/10.1016/0076-6879\(83\)01039-3](https://doi.org/10.1016/0076-6879(83)01039-3)
- Mazini, M., Shirazi, B., & Mahdavi, I. (2018). Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms. *Journal of King Saud University - Computer and Information Sciences*. <https://doi.org/10.1016/j.jksuci.2018.03.011>
- Mishra, V. P., & Shukla, B. (2018). Development of simulator for intrusion detection system to detect and alarm the DDoS attacks. *2017 International Conference on Infocom Technologies and Unmanned Systems: Trends and Future Directions, ICTUS 2017, 2018-Janua*, 803–806.  
<https://doi.org/10.1109/ICTUS.2017.8286116>
- Mousavi, S. M., & Affairs, P. (2014). *Early Detection of DDoS Attacks in Software Defined Networks Controller Early Detection of DDoS Attacks in Software Defined Networks Controller*.
- 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/ICNC.2015.7069319>
- Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *2015 Military Communications and Information Systems Conference, MilCIS 2015 - Proceedings*, 1–6.  
<https://doi.org/10.1109/MilCIS.2015.7348942>
- Muhammad Diyan , Murad Khan, B. N. S. and K. H. (2020). Scheduling Sensor Duty Cycling Based on Event Memory and Reinforcement Learning. *Sensors*.
- Niyaz, Q., Sun, W., & Javaid, A. Y. (2016). A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN). *DI*, 1–18. <https://doi.org/10.4108/eai.28-12-2017.153515>
- Osanaïye, O., Cai, H., Choo, K. R., Dehghantanha, A., Xu, Z., & Dlodlo, M. (2016). Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing. *EURASIP Journal on Wireless Communications and Networking*. <https://doi.org/10.1186/s13638-016-0623-3>
- Ozerdem, O. C., Olaniyi, E. O., & Oyedotun, O. K. (2017). Short term load forecasting using particle swarm optimization neural network. *Procedia Computer Science, 120*, 382–393.  
<https://doi.org/10.1016/j.procs.2017.11.254>

- Panigrahi, R., & Borah, S. (2018). A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems. *International Journal of Engineering & Technology*, 7, 479–482. <https://www.researchgate.net/publication/329045441>
- Peffer, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3), 45–77. <https://doi.org/10.2753/MIS0742-1222240302>
- Phan, T. V., Bao, N. K., & Park, M. (2017). A Novel Hybrid Flow-Based Handler with DDoS Attacks in Software-Defined Networking. *Proceedings - 13th IEEE International Conference on Ubiquitous Intelligence and Computing, 13th IEEE International Conference on Advanced and Trusted Computing, 16th IEEE International Conference on Scalable Computing and Communications, IEEE International*, 350–357. <https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0069>
- Potluri, S., & Diedrich, C. (2016). Accelerated deep neural networks for enhanced Intrusion Detection System. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*. <https://doi.org/10.1109/ETFA.2016.7733515>
- Rukavitsyn, A., Borisenko, K., & Shorov, A. (2017). Self-learning method for DDoS detection model in cloud computing. *Proceedings of the 2017 IEEE Russia Section Young Researchers in Electrical and Electronic Engineering Conference, ElConRus 2017*, 544–547. <https://doi.org/10.1109/ElConRus.2017.7910612>
- Sahu, S. K., Sarangi, S., & Jena, S. K. (2014). A detail analysis on intrusion detection datasets. *Souvenir of the 2014 IEEE International Advance Computing Conference, IACC 2014*, 1348–1353. <https://doi.org/10.1109/IAdCC.2014.6779523>
- Salehinejad, H., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2017). *Recent Advances in Recurrent Neural Networks*. 1–21. <http://arxiv.org/abs/1801.01078>
- Scarfone, K., Mell, P., & Mell, P. (2007). *Guide to Intrusion Detection*.
- Sengupta, N., Sen, J., Sil, J., & Saha, M. (2013). Designing of on line intrusion detection system using rough set theory and Q-learning algorithm. *Neurocomputing*, 111, 161–168. <https://doi.org/10.1016/j.neucom.2012.12.023>
- Servin, A., & Kudenko, D. (2008). Multi-agent reinforcement learning for intrusion detection. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4865 LNAI(June), 211–223. [https://doi.org/10.1007/978-3-540-77949-0\\_15](https://doi.org/10.1007/978-3-540-77949-0_15)
- Sharafaldin, I., Gharib, A., Lashkari, A. H., & Ghorbani, A. A. (2017). Towards a Reliable Intrusion Detection Benchmark Dataset. *Software Networking*, 2017(1), 177–200. <https://doi.org/10.13052/jsn2445-9739.2017.009>
- Sharafaldin, I., Habibi Lashkari, A., & Ghorbani, A. A. (2018). *Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization*. *Cic*, 108–116. <https://doi.org/10.5220/0006639801080116>
- Shi, J., & Yang, Y. (2017). *Intrusion Detection in Software Defined Network*. *October*, 1–5.
- Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A Deep Learning Approach to Network Intrusion Detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*. <https://doi.org/10.1109/TETCI.2017.2772792>
- Singh, S., Raiwani, Y. P., & Singh, L. (2019). *Evaluation of Network Intrusion Detection with Features*

- Slay, J., & Moustafa, N. (2015). *Creating Novel Features to Anomaly Network Detection Using DARPA-2009 Data set Big Data Analytics for Intrusion Detection System: Statistical Decision-making using Finite Dirichlet Mixture Model View project I am developing a new Geometric AREA analysis t. July*. <https://doi.org/10.13140/RG.2.1.1993.7765>
- Srivastava, S. (2014). Weka: A Tool for Data preprocessing, Classification, Ensemble, Clustering and Association Rule Mining. *International Journal of Computer Applications*, 88(10), 26–29. <https://doi.org/10.5120/15389-3809>
- Tang, T. A., Ali, S., Zaidi, R., McLernon, D., Mhamdi, L., & Ghogho, M. (2018). *This is a repository copy of Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks. Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks*. 25–29. <http://eprints.whiterose.ac.uk/129091/>
- Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R., & Ghogho, M. (2016). Deep learning approach for Network Intrusion Detection in Software Defined Networking. *Proceedings - 2016 International Conference on Wireless Networks and Mobile Communications, WINCOM 2016: Green Communications and Networking*. <https://doi.org/10.1109/WINCOM.2016.7777224>
- Wang, R., Jia, Z., & Ju, L. (2015). An entropy-based distributed DDoS detection mechanism in software-defined networking. *Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015*, 1, 310–317. <https://doi.org/10.1109/Trustcom.2015.389>
- Weidong Kuang. (2021). Fundamentals of Reinforcement Learning: Learning Objectives. *University of Texas Rio Grande Valley*, 2–4.
- Xiao, P., Qu, W., Qi, H., & Li, Z. (2015). Detecting DDoS attacks against data center with correlation analysis. *Computer Communications*, 67, 66–74. <https://doi.org/10.1016/j.comcom.2015.06.012>
- Yuan, X., Li, C., & Li, X. (2017). DeepDefense: Identifying DDoS Attack via Deep Learning. *2017 IEEE International Conference on Smart Computing, SMARTCOMP 2017*, 1–8. <https://doi.org/10.1109/SMARTCOMP.2017.7946998>
- Yusof, A. R. A., Udzir, N. I., Selamat, A., Hamdan, H., & Abdullah, M. T. (2018). Adaptive feature selection for denial of services (DoS) attack. *2017 IEEE Conference on Applications, Information and Network Security, AINS 2017, 2018-Janua*, 1–4. <https://doi.org/10.1109/AINS.2017.8270429>
- Zhang, J., Liang, Q., Jiang, R., & Li, X. (2019). A feature analysis based identifying scheme using GBDT for DDoS with multiple attack vectors. *Applied Sciences (Switzerland)*, 9(21). <https://doi.org/10.3390/app9214633>

## Appendix A: Editor's Report



**VICE-CHANCELLOR**  
Prof. Samson M.I. Sajidu,  
*BSc Mlw, MPhil Cantab, PhD Mlw*

▶ Connect with Excellence

**UNIVERSITY OF MALAWI**  
P.O. Box 280, Zomba, Malawi  
**TEL:** (265) 1 524 222  
**FAX:** (265) 1 524 046  
**EMAIL:** vc@unima.ac.mw

Our Ref:

Your Ref:

23 August 2022

To whom it may concern

**LANGUAGE EDITING OF THESIS BY GABRIEL PRECIOUS MUSA**

This is to certify that a thesis submitted in fulfilment of the requirements for the degree of **Master of Computer Science** at the Namibia University of Science and Technology titled **EFFICIENT HYBRID DEEP REINFORCEMENT LEARNING MECHANISM FOR DISTRIBUTED DENIAL OF SERVICE ATTACK DETECTION IN SOFTWARE DEFINED NETWORKS** by Gabriel Precious Musa was submitted to me for language editing.

The thesis was professionally edited and tracked changes, comments and suggestions were made in the document accordingly. The editing process did not alter the content of the research or the author's intentions and the author is at liberty to accept or reject my suggestions.

Yours faithfully,

Dr. Emmanuel Ngwira  
Associate Professor of English  
English Department  
**UNIVERSITY OF MALAWI**