ADDIS ABABA UNIVERSITY

COLLEGE OF NATURAL SCIENCES

Ensemble-Based DDoS Attack Detection Model for Software-Defined Networks by Utilizing Flow-Based Features

Ermias Tsegu Gizaw

A Thesis submitted to the Department of Computer Science in Partial Fulfilment of the Degree of Masters of Science in Computer Science.

Addis Ababa, Ethiopia

October, 2023

ADDIS ABABA UNIVERSITY
COLLEGE OF NATURAL SCIENCES

Ermias Tsegu Gizaw

Advisor: Ayalew Belay (Ph.D.)

This is to certify that the thesis prepared by Ermias Tsegu Gizaw, titled: *Ensemble-Based DDoS Detection Model for Software-Defined Networks by Utilizing Flow-Based Features* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

Name		Signature	Date
Advisor:	Ayalew Belay (Ph.D.)		
Examiner:	Dida Midekso (Ph.D.)		
Examiner:	Minale Ashagrie (Ph.D.)		

ABSTRACT

In this modern era, networking has advanced in a swift manner. The need for businesses to incorporate the benefits of having enormous amount of dynamic applications, services, physical objects, machines, etc is skyrocketing. Therefore, the networking infrastructure become more complex in terms of networking devices and resource utilization. Therefore, the traditional networking paradigm becomes ineffective and inefficient to handle those requirements. As a result, Software Defined Networking (SDN), gets more consideration from researchers and practitioners.

In contrast to the traditional networking paradigm, SDN has efficient resource utilization, simple network management capability, better performance, network virtualization capability, and network programmability capability. But, it is with some serious network security issues like network tampering, unauthorized access, flow rule conflict, poor controller deployment, and Distributed Denial of Service attack (DDoS). Among these security issues DDoS is one of the devastating attacks on SDN. As a result, several studies are conducted to detect DDoS on SDN networks by utilizing statistical approaches, traditional machine learning (ML) approaches, and state-of-the-art techniques like DL. The traditional ML techniques are less efficient in contrast with the state-of-the-art approaches like DL. On the other hand, the state-of-the-art techniques are computationally complex. To overcome this problem, we proposed an ensemble-based DDoS detection model for SDN by utilizing flow-based dataset.

The experiment is conducted using the InSDN dataset. The dataset contains the normal group that contains the normal traffic, the metasploitable-2 group which contains attacks that target the metasploitable-2 server, and the Open Virtual Switch (OVS) group contains attacks that target the OVS machine. Because the dataset contains attacks like DoS, DDoS, Web Attacks, R2L, Malware, Probe, and U2R attacks, it is a must for us to separate the DDoS attack data to prepare it for our purpose. Furthermore, we split the dataset into 70% for training, and 30% for testing purposes. The result shows that the adaptive boosting ensemble technique has the highest accuracy with a value of 100%. But, when it comes to latency the gradient boosting algorithm has the minimum latency with a value of 60.6 ms. On the contrary, the KNN_DT-based stacking algorithm has the highest latency with 119,431.5 ms.

Keywords: Software Defined Network (SDN), Distributed Denial of Service Attack (DDoS), Machine Learning (ML), Ensemble-Based Algorithms.

DEDICATION

I like to dedicate this thesis work to Yanet Yared, Kalkida Yared, Eelol Samson, Yael Yared, Yonatan Samson, Fenet Micahel, Betselote Solomon, Hadoran Estifanos, Yabtsega Samson and Benayas Estifanos who are the new generation of the family. This thesis work is dedicated to you with the assumption that it will motivate each of you to a new and better achievement both in life and academia.

ACKNOWLEDGEMENTS

The entire learning process started when Addis Ababa University College of Natural Sciences Department of computer science allowed me to be one of its successful students. Though there are so many administrative aspects that the university has to address, it is also important to acknowledge and recognize the University and its community for what they provide me through the years. In association with this, I would like to thank my dear instructors because they gave me enormous knowledge and experience, and supported me immensely throughout my stay at the university. I am very grateful to my advisor Ayalew Belay (Ph.D.) for the enormous support that he provides me as an instructor while he gave me the courses during lecture time and as an advisor for this particular research work.

Finally, I would like to take this opportunity to thank my family for being extremely helpful, supportive, and a source of motivation throughout the learning process. Friends and collageous who helped me through the process, I am very thankful for your support. Much more than all these I would like to thank the almighty God for supporting me and giving me courage, strength, and blessing throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

List of Algorithms

LIST OF ACRONYMS

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CSV	Comma Separated Values
DDoS	Distributed Denial of Service attack
DL	Deep Learning
DPI	Deep Packet Inspection
DNN	Deep Neural Network
DT	Decision Tree
GPU	Graphics Processing Unit
IoT	Internet of Things
KNN	K-Nearest Neighbor
LG	Logistic Regression
ML	Machine Learning
NaN	Not a Number
NB	Naïve-Bayes
NFV	Network Function Virtualization
NN	Neural Network
ONF	Open Network Foundation
OVS	Open Virtual Switch
QoE	Quality of Experience
QoS	Quality of Service
R2L	Remote-to-Local
RF	Random Forest
RNN	Recurrent Neural Network
RST	Reset
SDN	Software Defined Networking
SPF	Shortest Path First
SYN	Synchronize

SVM	Support Vector Machine
SOM	Self-Organizing Map
TCP	Transfer Control Protocol
TPU	Tensor Processing Unit
U2R	User-to-Root
UDP	User Datagram Protocol
WAN	Wide Area Network

CHAPTER ONE: INTRODUCTION

1.1 Background

In this very technological time and moment of this modern era, the rapid increment of the development of networking has been a continuous trend. AL-Badrany and Al-Somaidai [1] stated that networking is exponentially growing and increasing concerning scalability and complexity. As a result, having a genuine approach is a must to overcome the ripple effects of the rapid advancements of computer networks and the surrounding technologies. For this, [1] argues that Software Defined Networking (SDN) can provide elegant solutions for network complexity. Due to this, recently, SDN has been gaining lots of attention from the network research community [1].

Zhai et al. [2] describe SDN as a new type of networking architecture. This new architecture is dynamic, flexible, manageable, and adaptable. According to Kyaw et al. [3], SDN is composed of three layers. These are "the infrastructure layer, control layer, and application layer." Devices like switches and routers are part of the infrastructure layer also known as the data plane. According to the configuration of the flow table, network devices in this layer forward or drop incoming packets. Therefore, the core function of the control layer is to setup the flow rules to the forwarding devices to control packet forward or packet drop. On the other hand, the application layer which is the upper layer of the architecture "also known as the management plane" consists of applications that leverage the functionalities provided by the northbound interface.

So, the principal philosophical idea behind SDN is a segregation of the control and data layer. In addition, the implementation of the idea of a logically centralized scheme to make network activities under centralized control and management can be considered a pivotal contribution to this architecture [4]. As a result of this implementation, now it is possible to have "a global view of the network" activities. Therefore, SDN can program the network through the application programming interface and add the required functions to the network [5]. Besides, applications in SDN can manage and control networks while ensuring load balancing, access control, and routing. Therefore, these contributions can be considered the most significant benefits of this new networking paradigm [5].

Though SDN is a technology with several benefits and advancements that are designed by putting the current technological advancement in perspective, there is one specific drawback of the architecture, that is the concept of centralization which makes it vulnerable to Distributed Denial of Service (DDoS) attack in particular. DDoS is a type of attack that makes available resources unavailable by sending fake packets to the victim. In this attack, multiple connected devices initiate the attack. According to Bavani et al. [6], DDoS attacks can be categorized as volumetric attacks, protocol attacks, and application layer attacks. The Principal objective of a volumetric attack is to devastate the victim's network bandwidth. Therefore, a huge amount of traffic is injected by the attacker. The protocol attack on the other hand uses the weakness of the protocol stack of the network and transport layer. On the contrary, the application layer attack uses the weakness of the protocol stack of the application layer. Anagha et al. [7] stated that SDN architecture is prone to DDoS attacks. A malicious user can rule out the competent service level of SDN by consumption of resources in the data plane layer of the architecture and affecting the network services in the control plane layer of the architecture. Such kinds of malicious activities, and attacks can bring the DDoS threat a real challenge specifically for SDN.

As a result, when it comes to detecting and mitigating DDoS attacks on SDN, Machine Learning (ML) approaches have a great deal of contribution. According to [5], ML algorithms play a significant role in traffic classification, security, and traffic management. These opportunities extends to both traditional networks and SDN environments [5]. There are several ML techniques, such as supervised, unsupervised, semi-supervised, and reinforcement learning. Moreover, the ensemble techniques combine several single ML learners to enhance the performance of a single learner in terms of accuracy and efficiency [8]. As a result, Mabayoje et al. [8] argue ensembles have become prominent and applied to various fields like pattern recognition and data mining. According to Wang et al. [9], ensemble learning techniques can be categorized as boosting, bagging, and stacking. The boosting algorithms “use the initial training data set to train the base learner. Then, adjusts the weight of the training sample according to the prediction result of the base learner to train the next base learner.” The final result is determined by the combined effect of the individual learners. The bagging algorithm obtains different training sets by repeatedly sampling the training sample to train several base learners, and the final result will be the combined result of each learner. The stacking algorithm on the other hand uses the initial training set to train multiple base learners, and the combined result of individual base learners will be the final result.

1.2 Motivation

The emergence of SDN has introduced new possibilities and challenges in the field of network security. For example, Bhayo et al. [10] argue that IoT devices with SDN can help to detect any type of intrusion in IoT networks. From another perspective, one significant challenge is the increasing threats of DDoS attacks, which can disrupt network operations and compromise the availability of services [11]. To address this challenge, researchers have explored the applications of ML techniques in detecting and mitigating DDoS attacks [12,13].

However, there remains a need for further research in utilizing ensemble-based techniques specifically tailored for SDN. Therefore, the motivation for this research originates from the critical importance of protecting SDN environments against DDoS attacks. By developing and evaluating ensemble-based ML techniques, this research aims to enhance the efficiency, performance, and accuracy of DDoS detection in SDN by utilizing state-of-the-art flow-based datasets that is particularly generated from SDN networks.

1.3 Statement of the Problem

According to Anagha et al. [7], DDoS attacks are catastrophic, especially for SDN. As a result, several research works have been conducted in recent years to detect and mitigate the vicious impact of this deadly attack. From the literature survey conducted for this research, we observe four broad categories of research works regarding detecting DDoS attacks on the SDN environment. These are statistical approaches [6,14,15], traditional ML techniques [3,16], DL-based state-of-the-art approaches [17], and ensemble-based techniques [18].

The statistical approaches use a particular threshold value to compare it with a calculated entropy value to detect the occurrence of a DDoS in SDN-based networks. However, because the threshold value depends on the experience of the researchers, these approaches have shortcomings in effectively detecting DDoS attacks. When these approaches are compared to traditional ML techniques like SVM (Support Vector Machine), KNN (K-Nearest Neighbour), etc., they are relatively inefficient and ineffective.

Regarding the second category, generally, traditional ML techniques are less accurate when compared with state-of-the-art techniques like DL [19]. Moreover, concerning the data sets, these research works utilize outdated datasets like KDD1999 which was generated two decades ago. As

a result, the current network protocols and complex network scenarios are not part of the dataset features. Similarly, to make the problem even worse, these outdated datasets are generated from conventional networks. Therefore, the OpenFlow protocol of the SDN network is not part of these datasets. Due to this, developing a model with these datasets and deploying them in the current complex network scenarios and the SDN network architecture is not realistic. Therefore, these datasets can not be efficient, effective, and convenient choices to work with SDN-based research.

Besides, concerning DDoS attacks, most researchers [13,15] consider only the UDP flood attack. Because of this, other flooding attacks like TCP flood and ICMP flood were not part of these research works. Therefore, the proposed models by these studies can only detect the UDP flood attacks of DDoS on SDN.

The most recent studies are in the third category and utilized state-of-the-art techniques like DL [17]. These research findings have excellent accuracy compared with the statistical and ML approaches but with considerable training time, testing time, and cost of computational resources [18]. In this perspective, Mbasuva and Zoid [20] argue that, though current state-of-the-art DL techniques enhance learning accuracy, they are also computationally expensive and take enormous training and testing time. Spending more than 100 GB of RAM and using a GPU acceleration is a similar characteristic among DL models [18]. Furthermore, most of the studies in this category also utilized outdated, and non-SDN representative datasets. As a result, the efficiency and performance of these researches are also in a dire situation for improvement.

When it comes to ensemble-based approaches like boosting, bagging, and stacking for detecting DDoS on SDN so far, there has been little attempt. The authors [18,20] study an ensemble of DL techniques. These techniques have enormous time, space, and computational overhead even compared with a single DL technique. Besides, some attempts exist to address the boosting method, but the nature of the dataset used is still a pivotal issue. As a result, there is a need for further study the ensemble-based techniques to address these research gaps. The major attempt of the this research work is to consider these gaps and develop an ensemble-based DDoS detection model for SDN by utilizing a flow-based dataset generated from an SDN networking environment and with up-to-date traffic and protocols in it. Thus, the proposed work will enhance the performance and efficiency of DDoS detection on SDN-based networks.

1.4 Objectives

The general and specific objectives of this research work are discussed as follows:

General Objectives

The general objective of the research work is to Design ensemble-Based DDoS attack Detection Models for SDN by utilizing Flow-Based Feartures.

Specific Objectives

To accomplish the general objective of the research work, the following specific objectives are carefully crafted.

- To survey the state-of-the-art approaches, tools, techniques; and best practices of existing research work to have a frame of reference.
- To gather software components and dataset preparation and pre-processing for training and testing ML models.
- To develop system prototype of the proposed model.
- To evaluate the performance of the proposed model through the prototype.

1.5 Methodology

Literature Review:

In doing this research, a literature review will be conducted. This helps to get the overall understanding of how other researchers conduct such kinds of research and understand the research trend, to acquire the techniques, approaches, and best practices.

Data Collection and Preparation:

In conducting this research, data collection, and preparation techniques are important activities. The proposed research work utilizes a flow-based dataset for the sake of having accurate results. In addition, the already collected data will be pre-processed before using it to train and test the ML-based DDoS detection model.

Prototype Development:

In this research journey, developing the models is the pivotal component and contribution specifically when it comes to the ensemble-based DDoS detection model. As a result, the ensemble-based DDoS classification Model for SDN will be developed. Several open-source and free tools like Python programming and the Anaconda development environment will be used to develop the models. In addition, Ryu and Mininet tools will be used to create a simulation environment to further test the developed model in a simulated environment.

Evaluation:

During the research activity, the developed ensemble-based DDoS detection models for the SDN will be evaluated using a testing datasets. The standard metrics such as accuracy, precision, recall, f1-score, and latency used as evaluation metrics.

1.6 Scope and Limitations

This research work develops and experiments with ensemble-based ML models for SDN controllers to detect DDoS attacks. An approach to compare and contrast the accuracy, efficiency, and effectiveness of the ML model is proposed. Besides, the feature selection strategy is implemented in a way to include flow-based features. But, the effects of the features selection approach on the accuracy and efficiency of the models will not be addressed in this particular research work.

In SDN, DDoS attacks can take place at all layers of the architecture. The DDoS attack detection and classification scheme that is designed and proposed in this research work are only considering attacks that are targeting the control layer of the SDN Architecture. Furthermore, though there are various types of DDoS attacks like volumetric attacks, protocol attacks, and slow rate attacks, this research work considered only volumetric attacks like UDP flood and TCP flood.

1.7 Applications of Results

According to Luong et al. [21], a DDoS attack is a fatal and widespread threat on today's Internet. This research work discussed that massive DDoS attacks appear more frequently. For instance, according to this particular research work, in 2016 the world witnessed one of the largest DDoS attacks at the time. Dyn, a DNS service provider company, experienced a massive DDoS attack, at its peak, the system suffered incoming traffic at a rate of 1.2Tbps. In 2018, Github experienced

the largest DDoS attack in history, at a rate of 1.35Tbps. Therefore, having the disastrous side of a DDoS attack in mind is a must for one to find an accurate and most precise solution for the problem at hand. From this perspective, the proposed research work will provide a solution for this particular application area.

1.8 Organization of the Rest of the Thesis

The rest of the thesis work is composed as follows: Chapter Two discusses the literature review on the research domain so that the basic background of the research is covered to have a general context. In Chapter Three, related works that are associated with many previous research will be discussed. Chapter Four of the research work gives enormous emphasis on designing the proposed solution for the identified research problems; while Chapter Five discusses implementation details of the proposed solution and the performance of each ensemble-based technique in terms of accuracy and efficiency. Finally, Chapter Six concludes the research work together with a future work recommendation.

CHAPTER TWO: LITERATURE REVIEW

This Chapter discusses a detailed background knowledge of the intended research topic and objectives. A comparative view of SDN networks and conventional networks, the limitations of conventional networks, advantages of SDN networks over conventional networks, SDN network architecture, SDN network workflow, and OpenFlow protocol have been given great emphasis. Then, the security threats on SDN are discussed briefly to have a general understanding of attacks in SDN. How and Why DDoS has been a critical source of problems on SDN networks will be answered. What SDN features are exploitable, and what attacks are possible attacks on these particular SDN features have been given focus. Next, an overview of ML techniques, roles of ML techniques on SDN, drawbacks of traditional ML techniques, and advantages and disadvantages of state-of-the-art DL techniques will be discussed. Finally, the discussion will be concluded after giving the appropriate attention to ensemble-based ML techniques.

2.1 Software-Define Networks and Conventional Networks

Most of the networking functionalities in traditional networking are implemented in dedicated appliances and hardware like switches and routers. These network devices are from various vendors and suppliers; hence required vendor-specific and low-level commands to manage, operate, and maintain the network. Because of these, such conventional networks become rigid, unscalable, and difficult to manage and configure. In addition, complexities in the business need and implemented policies make managing and operating conventional networks enormously cumbersome [22].

SDN the new network paradigm has tremendous contributions to facilitating network management and enabling programming capabilities for today's complex and sophisticated network infrastructures [22,23,24]. Because of this new networking architecture, once the SDN infrastructure is deployed it is not necessary to configure the individual networking devices repeatedly in a given network [23]. Users are expected to only define simple network rules on the controller to let the network infrastructure act according to the specified need. Compared with conventional networks, SDN proposes a control plane and forwarding plane separation strategy [23,24,25]. Due to this strategy, SDN can probably replace the conventional networking architecture in the near future [22].

2.1.1 Conventional Networks

Almost all researchers in the field unanimously agreed that this day, due to the swift expanision of the Internet communication and mobile communication technologies, in regard to its enormous amout of dynamic applications, services, physical objects, and machines, the networking infrastructure becomes more heterogeneous and complex; mostly in terms of networking devices, and resources [26,27,28,29]. As the complexity of networks skyrocketed and frequent changes in the requirements and needs of network infrastructure are introduced to handle the highly variable business need, it becomes more difficult for conventional networks to provide a robust, flexible, maintainable, and manageable service. According to Parashar et al. [30], managing a network is a very challenging task. Beyond maintaining a secure network with high functionalities, network devices must include low-level configuration to implement the complex operation [22].

"The emergence of IoT and the adoption of multi-tenant data centers generate enormous amounts of traffic and add more complexity to networks." From another perspective, network traffic control and orchestration require adapting "to the time-varying changes in link utilization, bandwidth allocation, latency, energy consumption, and jitter. In this regard, unfortunately, the traditional network architecture is not well designed in terms of enabling fine-grained and Quality of Service (QoS) aware traffic engineering over the network." The highly tight integration of the control layer and data layer of the conventional networking architecture complicates the network traffic monitoring process. It is extremely time-consuming and expensive to manage the network devices separately, especially in a time-varying and multi-tenant data center environment. This result, in less QoS-aware flow control and inefficient resource utilization [29].

As Figure 2.1 depicts, the controller plane and the data plane of network devices for the conventional network architecture are encapsulated and tightly coupled [23,31]. Thus, "programmable networks" has been proposed by several researchers to facilitate network evolution, and SDN proposes a new strategy in which the forwarding hardware devices of the conventional network are Segregated from the control decisions. As a result, "It promises to dramatically simplify network management and enable innovation and evolution" [32].

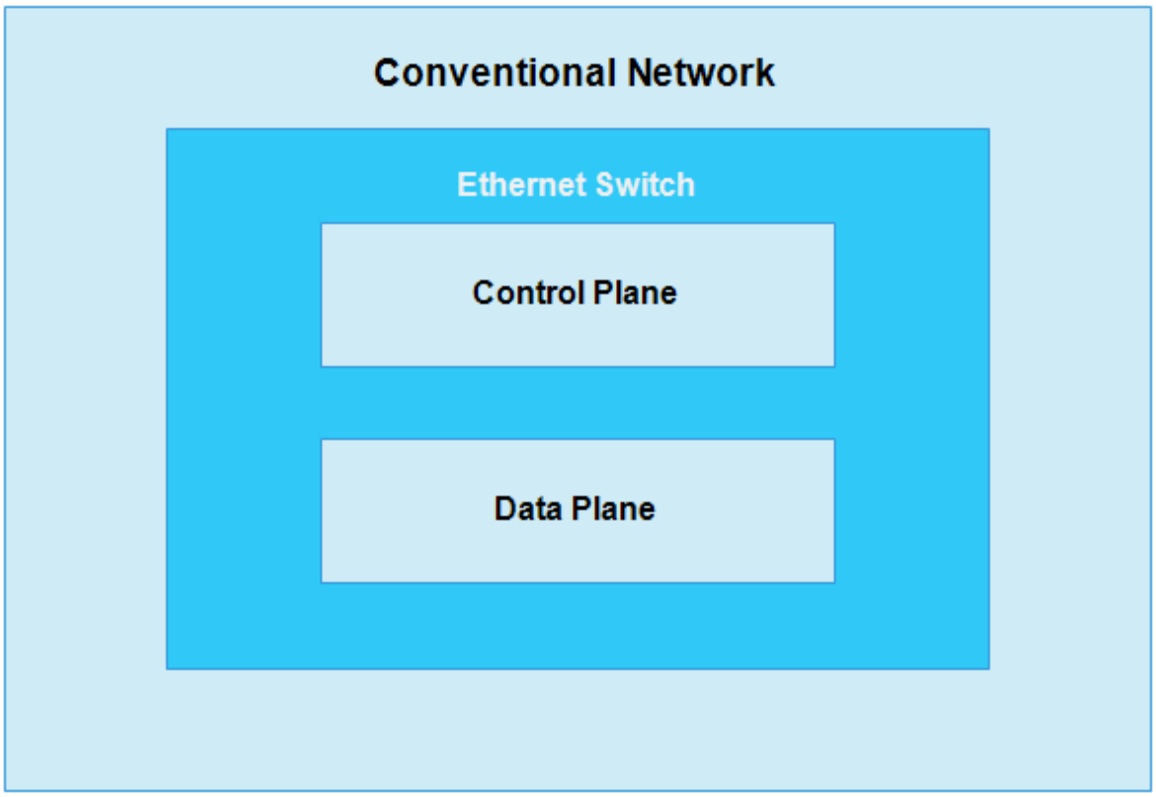

Figure 2.1: Conventional Networks

2.1.2 Software-Defined Networks

These days, SDN becomes one of the most important technologies for both academia and the information and technology industry. According to Liu et al. [23] "the market profits of SDN in the global operator market are expected to reach $9.5" billion by 2023. On the contrary, Alsaedi et al. [29] stated that the "SDN market accounted for $10.88 billion in 2015 and is expected to reach $134.51 billion by 2022." Though these different research works stated different statistics, the skyrocketed advancement of the SDN is a reality. According to Singh et al. [33], SDN is one of the biggest hopeful outcomes for the upcoming Internet. Google one of the biggest tech giants "deploys the B4 network and achieves an unprecedented 95% network utilization. Microsoft is trying to use the Opendaylight controller to improve the video experience of Skype users. SDN is also listed as one of the top ten innovative technologies by MIT" [23]. These are important indicators for the valuable contribution of SDN in terms of solving the complexity and difficulty of conventional network management among others.

SDN offers an "innovative way to design, establish, and manage networks. This network architecture is proposed by the Stanford University Clean Slate Research Group" [23,24,34]. This recent advanced technology with a new implementation, design, functionalities, and management approaches for handling networks "allows the abstraction and decentralized management of the low-level network functionalities by decoupling the network logic from the data forwarding devices into the logically centralized distributed controllers" [23,29,31].

As it is depicted in Figure 2.2, SDN separates the data plane and control plane as different physical components. This implementation can solve the problem of interdependency between the data layer and the control layer. The breakdown of the tightly coupled data plane and control plane of the traditional network architecture makes the SDN architecture to be more controllable, secure, and economical in terms of resource utilization [26].

For the purposes of communication between control and data plane, the OpenFlow protocol is used. Because of the segregation of the control and the data plane, and the implementation of the OpenFlow protocol now more programming capability of the network architecture can be obtained.

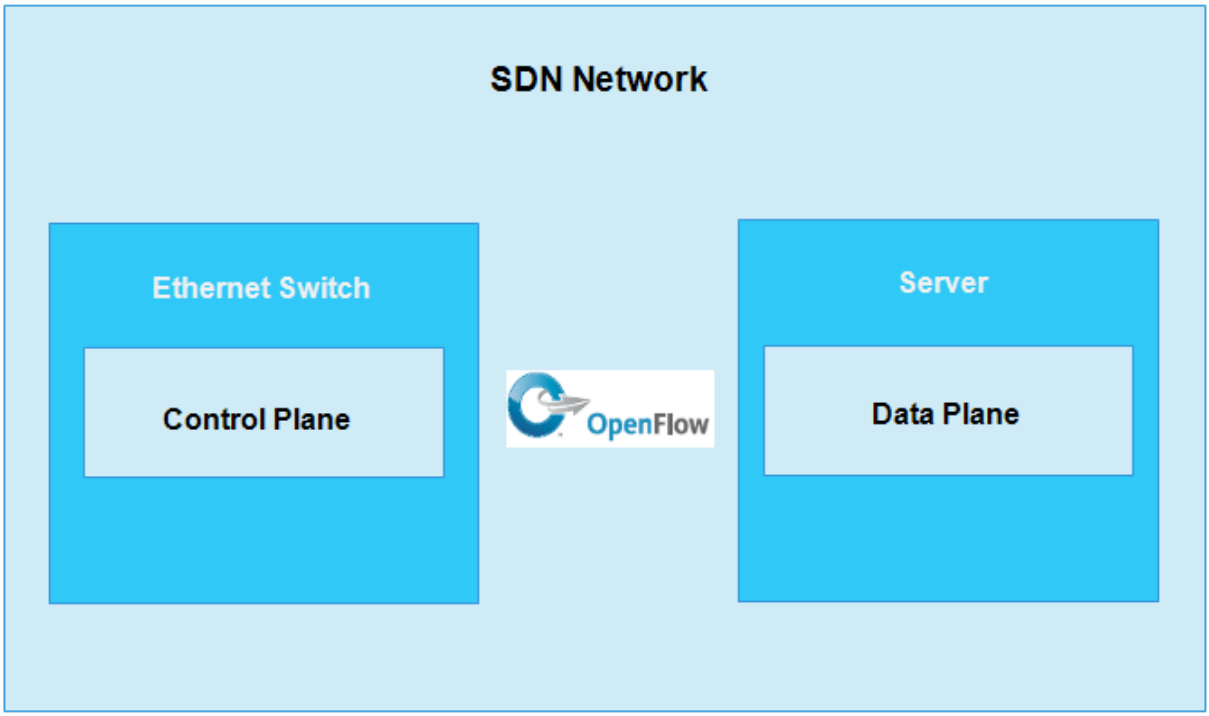

Figure 2.2 SDN Networks

2.1.3 Limitations of Conventional Networks

Evidently, conventional networks have been serving businesses for several years. But, these days traditional networks become very incapable of supporting the very dynamic, swift, and innovative technological environments. Furthermore, today almost everything is connected via the Internet which is accessible from anywhere anyplace, and anytime [33]. As a result, the very fundamental reasons for the limitation of conventional networks can be discussed from several perspectives. For instance, management complexity, challenges in configuring policies, scalability issues, device vendor dependency, and lack of programmability can be mentioned. Some of the limitations of conventional networks are discussed briefly as follows.

- **Scalability Issue:** The tightly coupled design of traditional network devices like switches and routers together with the highly scalable, and changeable business environments and requirements can be mentioned as a major factor. Moreover, in conventional networks, to forward the packets, routers that use algorithms that define how and where to forward the data packets are used. Both forwarding and decisions of forwarding the data packet are made by the router using a routing algorithm [30]. Therefore, because, each device has its routing information, it is absolutely difficult to have the global network view that actually helps to have network scalability, intelligence, and programmability [29].
- **Challenges in Configuring Policies:** traditional network devices like routers, switches, and security devices like firewalls necessitated initial configurations and changes configurations as and when required "based on a specific embedded operating system (OS) or firmware to achieve high throughput and performance", and this is basically because traditional networks don't have a global view of the network. This was indeed a cumbersome activity because of the manual intervention required for every configuration on every node through which the traffic flows to respond to a wide range of network events and applications [22,32,35]. In this regard, network Administrators have to "manually transform the high-level policies into low-level configuration commands while adapting to changing network conditions. Often, they also need to accomplish these very complex tasks with access to very limited tools."
- **Device Vendor Dependencies:** Limitations like vendor dependencies, complexity, inconsistent policies, and the inability to scale make things a bit complicated and worse specifically when it comes to implementing multi-tenant datacenter on traditional networks

[32]. According to Alsaeedi et al. [29], "As data center grows in scale, network management operators struggle with too many manual reconfiguration processes, which may reach nearly 40 percent of the most common network operations problems. As a result, network management and performance tuning are quite challenging, and thus error-prone" [32]. Furthermore, traditional networks are not well designed to adapt and self-manage in case of, unpredictable faults and load changes in large-scale networks.

- **Lack of Programmability:** Conventional networks lack "programmability, and hence cannot meet the application layer's needs in real-time. The time-varying and tremendous amount of traffic in the application layer require global network visibility and abstraction for better QoS provisioning, and resource utilization, and avoiding the need to enforce global policies by carefully crafting" device-by-device configurations [29]. Therefore to overcome these challenges of conventional networks, SDN separates control and data layer. It also implements a programmable interface between the these layers on a standardized protocol called OpenFlow protocol [30].

2.1.4 Advantages of Software-Defined Networks

Unlike, conventional networks the SDN architecture distributes the tightly coupled control and forwarding logic into different layers [28,29,30,33]. Because the control and data layers are segregated, forwarding and decisions making are separated from each other. As a result of this separation, vital functions of the networks are logically centralized to the controller. Furthermore, it allows the network controller to handle and manage the networks according to user choice and requirements. According to [23,24,30], this feature separation of the controller and the data plane helps in managing high-speed complex systems. Besides, it allows the application plane to execute several functions on the network such as flow table management, load balancing, routing, DDoS, and intrusion detection and mitigation.

Moreover, the inherent advantages of SDN such as "logically centralized control, global view of the network, software-based traffic analysis, and dynamic updating of forwarding rules", allows SDN to be implemented by various networks, such as "transport networks, optical networks, wireless networks, IoT, edge computing, wide area networks (WAN), cloud computing, network function virtualization (NFV)" [25,28]. As an example, a centralized controller which collects information and determines the best forwarding policies can detect and handle element failures,

and then provide alternative routes. This property makes SDN an excellent candidate to support network reliability [31].

As He and Zhang [24] discussed, from the perspective of the network operators, SDN provides the possibility to optimize the management of network resources through software manual programs. The open standard of SDN simplifies network operations and increases the transparency of network development. As a result, it is of great importance to study SDN network measurement, collect network flow information, and obtain network performance indicators, which can better meet the needs of network applications such as network planning, anomaly detection, load balancing, and network security [24]. Predominantly, flexibility, redundancy, performance, automated load balancing, on-demand network scalability, change networks' rules on the fly, low cost on network operators, ability to program networks, manage large networks and improve delivery time can be listed as some of the advantages of SDN among several others[22,30]. The pivotal advantages of SDN are discussed briefly as follows.

- **Efficient Resource Utilization:** As Tseng et al. [36] argues, because SDN utilizes resources more efficiently by segregating control plane and data plane and introducing a logically centralized controller, it gains momentum among several network service providers including data centers, WANs, and cloud computing.
- **Network Management Simplicity:** The SDN network architecture promises to simplify network management just because it entails the control, management, and configuration of all network devices from a central controller [29,37]. Since the main driver for SDN arose from the modern data centers with huge volumes of servers and switches and huge data handled per second, manual configuration of all these networking equipment proved costly as well as time-consuming. In this regard, SDN provided the best possible solution by offering central control and management [37].
- **Better Performance:** By avoiding hardware restrictions and decoupling the control and data layer of conventional network architecture, "the SDN control plane will have a strong processing capability, and it will provide greater speed and flexibility in routing instructions and the energy management of networking equipment such as routers and switches" [26,30]. Concerning this, SDN results in a more efficient and innovative design and delivers better performance with higher flexibility without the use of proprietary devices [33].

- **Network Virtualization:** The SDN architecture facilitates and revolutionalized the transition of networks from a hardware-based mode to a software-based mode. This is mainly due to virtualization, where physical network devices are replaced by software that performs the same function. SDN is a result of several previous research efforts to make networks programmable, so that, the new architecture becomes more flexible and innovative. In this new paradigm, basic network functions such as routing or packet processing are performed by virtualized programmable routers, whose purpose is to enable rapid deployment of custom services and provide dynamic network configuration as needed [38].
- **Network Programmability:** The fact that SDN decouples the network control and forwarding functions makes the underlying infrastructure to be abstracted from applications and network services. The control plane is becoming centralized in a controller node, and the switch is considered a simple forwarding element executing the controller's rules on each traffic flow. As a result, the network controlling plane collects network statistics which enables the network to be more intelligent and directly programmable[38].

2.1.5 Software-Defined Network Architecture

The SDN architecture includes an "application layer, northbound interface, control layer, southbound interface, and data layer" components [24,32]. Figure 2.3 depicts the three-tier architecture of SDN. In this architecture, the uppermost layer of the architecture is the application layer. The northbound interfaces are used as a mechanism for the communication between the controller layer which is found right below the application layer, and the application layer itself. The control plane on the other hand is responsible to handle all logics and devices that are extremely important to establish rules and communication capabilities. The last layer in the architecture is the data plane also known as the infrastructure layer. Every data that is forwarded to the network is taken care of by the data plane. The data plane provides programmable interfaces through the southbound interfaces "which provide users with a complete set of APIs. As a result, users can use the APIs to program, configure, control, and manage the network on the controller" [23]. OpenFlow is one of the widely and most known southbound APIs that is used in academia and the IT industry.

- **Application Layer:** It is the uppermost plane in the SDN architecture which is composed of business applications [28]. This layer is also called "the management plane". Several applications that are developed by software developers to manage and interact with the network infrastructure are included in this particular layer [22]. This layer can include applications to configure, manage, monitor, and secure the network among others [24,32]. The communication between this layer and the controller layers is conducted using the northbound interface [24,32]. The provided information and business needs can determine the implemented control logic by the application to modify the network behavior [28].
- **Control Layer:** "The controller layer is the brain of the SDN network" [28,30,32]. As a result, it is responsible to control the whole system. For every packet that is generated within the network, the controller makes decisions, and then those decisions are deployed in the OpenFlow switches. It is also responsible for logically centralizing the distrusted network beyond exposing and abstracting the network state information for data layer to the application layer [28]. In another perspective, requirements from applications are interpreted into policies and allocated to the forwarding device by the controller. Additionally, "the controller provides essential functionalities that most network applications need, such as shortest path routing, network topology storage, device configuration, state information notifications, etc" [28].
- **Infrastructure Layer:** This is a data layer that is responsible for forwarding packets. Devices like Routers and Switches are operating in this layer of the SDN architecture [24,28,30,32]. The devices are exposed to the control layer through the Southbound communication interface. These devices are responsible to hold the flow table that has the flow rules. Depending on the flow rules packets are forwarded to the appropriate destination. Furthermore, depending on the specified action on the flow rule packets are processed [28].
- **Northbound API:** This is an open interface between the SDN controller and the network application which provides a common open programming interface for SDN applications [24]. These application programs communicate with the SDN controller layer through this interface.
- **Southbound API:** The southbound interface is an open interface between the SDN controller and the data plane [24]. The OpenFlow protocol is used in the Southbound interface of the SDN architecture. According to Parashar et al. [30] it is the most commonly used Southbound protocol for SDN that "is standardized by the Open Networking Foundation (ONF)."

- **Eastbound and WestBound API:** These are interfaces that are used by the controllers to communicate with each other. Because the controller is a centralized entity in the SDN network, it is vulnerable to a single point of failure. As a result, having a redundant controller in the network design is considered an important technique to mitigate this problem. When such a scenario takes place, the redundant controllers communicate with each other using these two interfaces.

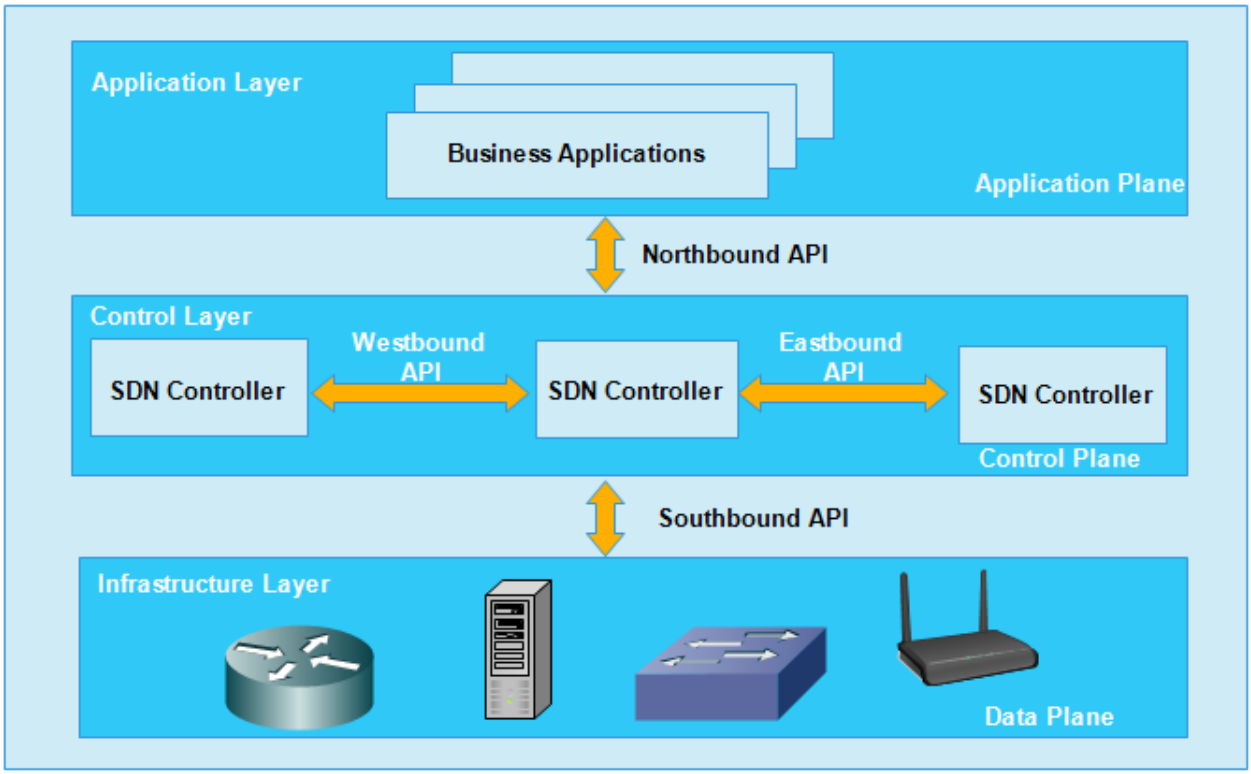

Figure 2.3: SDN Architecture

2.1.6 OpenFlow Protocol

OpenFlow protocol is an open standard and it is a very popular SDN specification. This protocol defines network devices' functionality and the communication protocol between data and control plane [30]. According to [22,29], OpenFlow is the de facto standard of SDN. A wide range of acceptance of this particular protocol by the academic society and the business society makes SDN very successful [22]. It serves as the southbound interface, allowing the controller direct access and control of data-forwarding network devices. The Open Network Foundation (ONF) has standardized OpenFlow to address the dynamic nature and high bandwidth of today's applications while reducing management complexity [29].

The ONF OpenFlow Switch Specification Version 1.5.1 [39] which is the latest specification document at the time of this writing has a detailed specification for the OpenFlow Switch component. “The specification covers the components and the basic functions of the switch, and the OpenFlow switch protocol to manage an OpenFlow switch from a remote OpenFlow controller”. As it is depicted in Figure 2.4, the remote controllers are communicating with the OpenFlow switch using the OpenFlow protocol.

The OpenFlow switch components are composed of the OpenFlow channel, group table, meter table, port, flow table, and pipeline. The flow tables and the group tables are used to perform packet lookups and forwarding while the OpenFlow channels are used to communicate with the external controller. “The OpenFlow logical switch communicates with the controller and the controller manages the switch via the OpenFlow switch protocol. Using this protocol, the controller can add, update, and delete flow entries in flow tables. The flow tables of the OpenFlow switch components contain a set of flow entries.” The flow entries have match fields, counters, and a set of instructions to apply to matching packets.

“Packet matching starts at the first flow table and may continue to additional flow tables of the pipeline. If a matching entry is found, the instructions associated with the specific flow entry are executed. If no match is found in a flow table, then the outcome depends on the configuration of the table-miss flow entry”. For example, the packet may be forwarded to the controllers over the OpenFlow channel, dropped, or may continue to the next flow table. Instructions that are associated with the flow entry either contain actions or modify pipeline processing. Packet forwarding, packet modification, and group table processing can be the actions that are included to describe the instruction. “Pipeline processing instructions allow packets to be sent to subsequent tables for further processing and allow information, in the form of metadata, to be communicated between tables. Table pipeline processing stops when the instruction set associated with a matching flow entry does not specify the next table; at this point, the packet is usually modified and forwarded.”

Port is one of the components of the OpenFlow switch specification at which flow entries are forwarded. “This is usually a physical port, but also a logical port defined by the switch or a reserved port defined by this specification can be a possible alternative. Reserved ports may specify generic forwarding actions such as sending to the controller, flooding, or forwarding using

non-OpenFlow methods switch processing." On the other hand, switch-defined logical ports may specify link aggregation groups, tunnels, or loopback interfaces. The group table component of the specification is designed to contain group entries. Each group entry contains a list of action buckets which is a set of actions in a group. These action buckets consist of specific semantics dependent on the group type. The actions in one or more action buckets are applied to packets sent to the group.

The OpenFlow switch protocol which helps the OpenFlow Switch to communicate with remote Controllers supports three message types; these are Controller-to-switch, asynchronous, and symmetric messages. "Controller-to-switch messages are initiated by the controller and used to directly manage or inspect the state of the switch. While the asynchronous messages are initiated by the switch and used to update the controller about network events and changes to the switch state". Finally, the symmetric messages are initiated by either the controller or the switch and sent without solicitation.

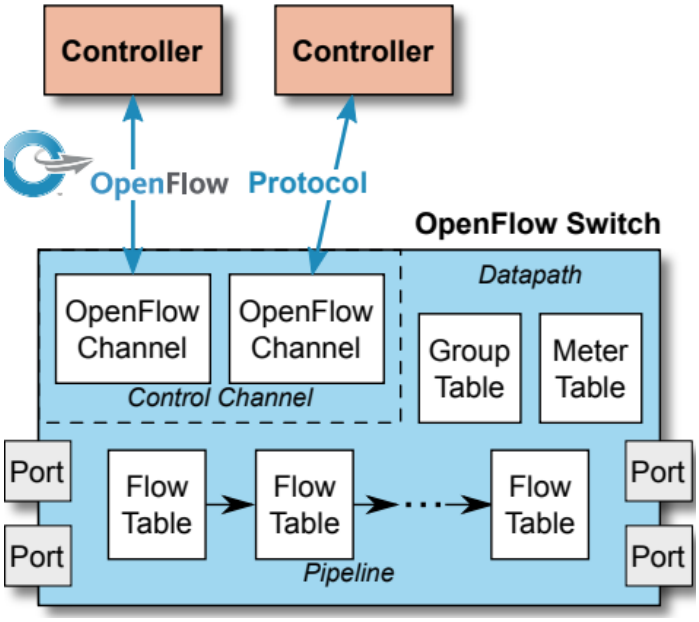

Figure 2.4: Components of OpenFlow Switch [39]

2.1.7 Basic Operation of Software-Defined Network

To have a firm understanding of SDN architecture, it is so crucial to have a clear understanding of the basic SDN workflow. Figure 2.4 shows the OpenFlow switch components. The data layer of SDN is composed of devices like switches and routers. According to Liu et al. [23], "this layer

includes basic installations such as software-based and hardware-based devices that perform specific data processing by receiving instructions from the controller layer". Typically, an OpenFlow switch which is composed of components like the flow buffer, flow table, and group table is used as a communication interface between the data layer and the central controller layer [22]. When the switch gets packets from the related input ports, it will automatically keep the packet in the flow buffer and find the corresponding rule for this packet from the flow table. If the rule is located, then the packet is forwarded to the corresponding output port or according to actions that are located in the flow entery of the switch the packet will be locally proceesed. Else, the request will be sent to the controller through the OpenFlow client for a new rule [22,28]. SDN uses a standardized protocol named OpenFlow that is used to have a global view of the SDN network. This is a protocol in which the server tells network switches about where to send packets and this protocol is defined in between the southbound interface. When a new packet arrives from a source IP address to a switch, if that address is unmatched or new to the flow table, by default that particular packet is forwarded to the controller. Then the controller generates and sends a flow rule to the switch [20,30,32].

According to the OpenFlow Switch Specification Version 1.5.1, [39], the OpenFlow switch has an OpenFlow pipeline component. This component Contains one or more flow tables, each with multiple flow entries. This particular component determines how packets interact with those flow tables. In this regard, the OpenFlow switch must have a minimum of one ingress flow table and may have more. "Starting at zero the flow tables are numbered in the order they can be traversed by packets."

Ingress and egress processing are the two stages of pipelining processing. The major distinction between the two processings is, Ingress processing is performed in the context of the input port named the ingress port while egress processing is processed within the context of the output port known as the egress port. The first egress table indicates the separation of the two stages; all tables with a number less than the first egress table must be used as ingress tables. "Always pipeline processing starts with the ingress processing when the first packet arrives at the ingress port of the OpenFlow switch. The packet must be first matched against flow entries of flow table Zero. Depending on the outcome of the match in the first table, other ingress flow tables may be used or the OpenFlow switch may be performed egress processing in the context of the specific output

port if the outcome of the ingress processing is to forward the packet to an output port." Since egress processing is optional, the OpenFlow switch may not have it or may not be configured to use one. In this case, "the packet must be processed by the output port, and in most cases, the packet is forwarded out of the switch. On the other hand, if a valid egress table is configured as the first egress table, the packet must be matched against the flow entries of the flow table."

"If a flow entry is found, the instruction set included in that flow entry is executed. A flow entry can only direct a packet to a flow table number that is greater than its own flow table number, in other words, pipeline processing can only go forward and not backward." On the other hand, if a packet does not match a flow entry in a flow table, this is called a table miss; and depending on the behavior of the table miss configuration instruction, like dropping the packet, passing the packet to another table, or sending the packet to the controller over the control channel via packet-in message packet processing will take place.

2.2 Security Issues of Software-Defined Networks

According to Liu et al. [23] "SDN lacks sufficient multi-level protection" mechanisms because the SDN security research is still in the infancy stage. As a result, much more investigation and researches need to be conducted. This research work argues that "in the past few years, researchers are focused on the development of SDN functions, such as resource scheduling and rule delivery." The importance of SDN security was not under appropriate consideration. Moreover, despite the enormous advantage of SDN like providing high openness and programmability, the very basic inherent nature of the network architecture results in new security issues [22,23]. According to Liu et al. [23], the security issues of SDN are closely related to its characteristics. In this research, the following characteristics or inherent nature of SDN are identified and considered as the main reasons for several security issues including the devastating DDoS vulnerabilities in the SDN environment.

"Centralized control and un-enough security protection mechanism, which makes SDN controller become an external malicious target."

1. "Complex interaction of various application programs. There is a high coupling between applications, which causes flow rules to conflict easily."
2. "Lack of adequate application authorization and authentication mechanisms, makes it vulnerable to malicious application attacks."

3. “Not enough security and encryption measures in the message exechange process between the control and the data layer. Flow rules are easy to suffer malicious tampering during the process of publishing.”

2.2.1 Attacks in Software-Defined Networks

The various security-related problems existing in SDN can be manifested in the following aspects: malicious applications, controller’s vulnerability, legitimacy and consistency of the flow rules, the northbound interface standardization issue, and communication security of the southbound interface, among other aspects [23]. According to Parashar et al. [30] attacks on SDN architecture can be network manipulation attacks, leakage of data, DDoS attacks, etc., and can occur at the application layer, northbound interface, control layer, southbound interface, and data plane layer. As a result, it is a very important technique to classify the security threats of SDN at different layers of the architecture and the communication interfaces as it is shown below.

- **Security issues on the application layer:** The application layer is the uppermost layer of the SDN architecture. This layer of the SDN architecture is affected by several security issues; for instance, according to Liu et al. [23], malicious applications such as viruses, trojans, and worms can attack SDN through software agents. Network devices at the data layer of the SDN architecture completely trust the flow rules delivered by the control layer of the SDN architecture and execute without consideration. In this case, if a malicious application participates in the formulation of the flow rules, it will bring unpredictable damage to the network. As a result, malicious applications are one of the most security threats to SDN [23]. Since there is a lack of sufficient trust management mechanisms between the control layer and the application layer, authentication and authorization problems are also mentioned as critical security issues of the SDN architecture. According to Liu et al. [23], currently, there is no complete application authorization and authentication module under the SDN environment. As a result, the controller is directly exposed to the attacker. From another perspective, attackers can tamper with the network configuration, steal information about the network, and seize the network assets by embedding malware in projects [22]. Because There are a large number of third-party open-source applications in SDN, applications not only bring challenges to application configuration management but also produce a higher degree of coupling which results in flow rule conflict [23].

- **Security Issue on Northbound Interface: "**Due to the diversity and constant update of SDN applications, there are no uniform provisions on the methods of authorization and authentication." As a result, at present, the largest security problem of the northbound interface is the standardization problem [23].
- **Security issues on the control layer:** Poor controller deployment, hijacked controllers, and DDoS are some of the severe security issues in the SDN architecture control layer. In this layer, the attacker creates a series of illegal accesses to impose excessive load on the controller, which causes the system resources an available to legitimate users hence DDoS occurs. On the other hand, "the attacker can use the admin station to hijack the controller in such a way causing the user's legitimate request to be rejected is inevitable. In some cases, the attacker can use physical or logical methods to destroy the controller and modify the network" [23]. Finally, poor controller deployment is another security issue of the control layer.
- **Security issues on the southbound interface:** In this part of the SDN architecture the security issues are mainly caused by the leak of the OpenFlow Protocol. As a result of this, Man-in-the-Middle (MITM) based attacks like DNS spoofing, session hijacking, and port mirroring are possible attacks. This particular attack between the switch and the controller can be used to interrupt and alter the sending policies issued to the switch with the specific goal to get control of packet sending [22]. From another perspective, DDoS to flood the flow table and flow buffer of the OpenFlow switch is a possible security issue of SDN for this particular part of the SDN architecture.
- **Security issues on the data layer:** The data layer section of the SDN architecture lacks an effective authentication mechanism between the basic equipment and the controller, and this is one of the security issues of SDN. As a result, security threats like identity impersonation and illegal access can happen [23]. Furthermore, malicious or incorrect flow rules can be injected to overwhelm the limited flow table space to initiate a DDoS attack [23].

Table 2.1: Summary of Security Issues on SDN

Plane	Security issues	Description
Application Layer	Malicious Application	Viruses, trojans, and worms can attack SDN through software agents.
	Network Tampering	Tamper with the network configuration, and steal information about the network.
	Unauthorized Access	Lack of mechanisms to verify whether the application is secure.
	Flow Ruel Conflict	Flow rules generated by the application appear to conflict.
Northbound Interface	Unauthorized Access	Lack of authorization and authentication
	Lack of Standardization	The largest problem of the northbound interface is the standardization problem
Control Layer	DoS / DDoS	Impose excessive load on the controller.
	Hijacked / Rouge controller	Take total control of the controller.
	Poor Controller Deployment	Inefficient deployment of the controller.
Southbound Interface	Man-in-the-Middle (MITM)	The attacker intercepts network information.
	DoS / DDoS	impose excessive load on the bandwidth.
Data Layer	Unauthorized Access	Lack of authorization and authentication
	DoS / DDoS	impose excessive load on the OpenFlow switch.

2.2.2 DDoS Attack and its Types

The principal objective of DDoS attacks is to enormously utilize the network's resources and services under attack so that legitimate services and devices become unavailable to legitimate users [30]. SDN's inherent characteristics are a very important attack point specifically for DDoS. When an attacker generates enormous malicious traffic from multiple attack nodes, because, the OpenFlow switch has no rule for the newly generated traffic it simply forwards this particular attack traffic to the centralized controller to decide what to do with it. In this regard, the centralized controller will be overwhelmed with the number of requests that are generated by the OpenFlow switches. As a result, the controller either stops responding or becomes very slow in its operation [30].

As the Internet grows in popularity, an increasing number of businesses begin to conduct online transactions. Serious DDoS attacks will result in massive economic losses and may even jeopardize national security [25]. If the attack is performed from several different distributed clients or hosts by targeting a single server, then a DDoS attack happens. In this situation, malicious and unauthorized users can take control of the target under consideration and the security and profitability of the business will compromise [30].

The DDoS attack is the most common and devastating attack in which the attacker first sends an attack command remotely to have overall control over a collection of malware-infected systems that form a network of bots also known as zombie computers [20]. Cybercriminals of these days can use botnets ranging from thousands to millions to launch a DDoS attack. These botnets are then used to enormously exhaust the target system with Internet traffic such as Syn floods, UDP floods, TCP connection exhaustion, etc. As a result, the overwhelming amount of traffic exceeds the capability of the target victim so that the target system refuses to serve legitimate users [25].

As Figure 2.5 illustrates, it is possible to categorize DDoS attacks into volumetric attacks, application layer attacks, and protocol attacks [20]. The volumetric attack sends enormous traffic to exhaust the network bandwidth. The traffic includes spoofed attacks and ICMP. The application layer attack also known as the Layer 7 attack, targets web applications by sending requests that may initially appear as legitimate users' requests, and in a short period, the victim is overwhelmed. The protocol-type attack is designed to exhaust the processing capacity of network infrastructure resources such as servers and firewalls, as well as network infrastructure management tools [20].

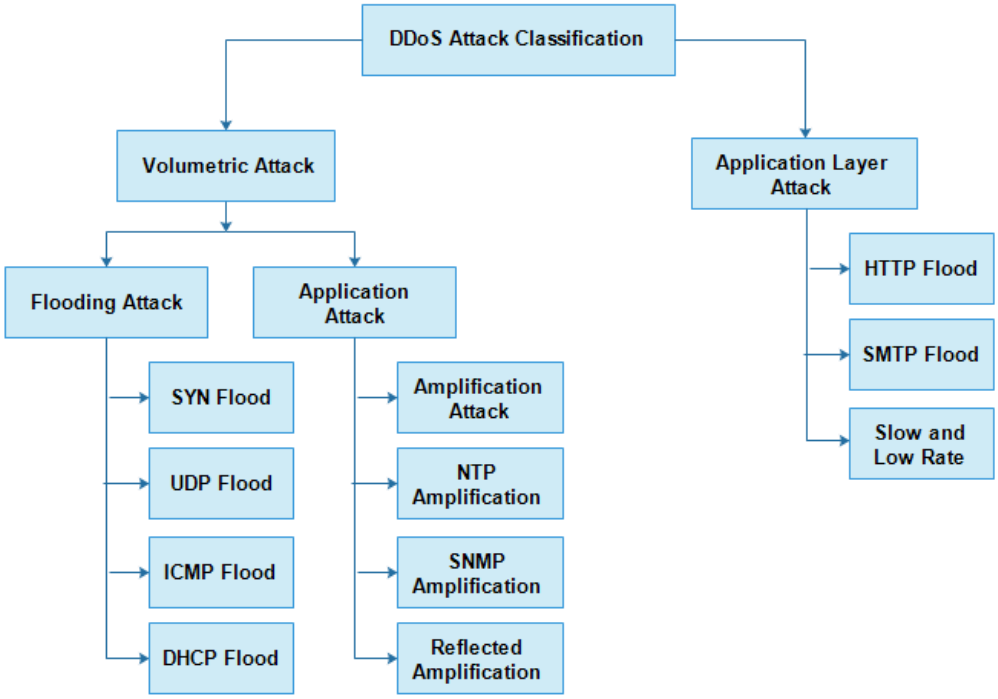

Figure 2.5: Types of DDoS Attacks

2.2.3 DDoS Attack on Software-Define Networks

Though SDN has enormous advantages over conventional networks, it has security threats that should not be overlooked. Many serious and disastrous attacks can affect this new network paradigm, and DDoS is the most common and devastating attack. This attack can cause enormous resource conception to shut down devices and corresponding services for legitimate users. Concerning SDN, it can attempt to make a large number of traffic to control and data plane, or the control plane bandwidth to create a situation that makes the network unavailable to legitimate users by exhausting the computing and memory resources [25,26,32]. As discussed earlier, several techniques can be used to perform DDoS attacks, such as "network-based approaches like flooding through TCP SYN, ICMP, or UDP packets, and host-based approaches, where one or several hosts target specific applications to exploit their authentication protocol or a particular algorithm" [32].

Attackers generate several new flows that have spoofed IP addresses to flood the bandwidth of the control plane, the OpenFlow switches, and the SDN controller. These Spoofed IP addresses are sent from multiple sources which results in network failure for legitimate hosts. "Because the spoofed addresses do not match any existing flow rules in the flow table, it results in a table-miss situation. Such a situation results in generating massive packet-in messages sent to the SDN

controller from the victim OpenFlow switch, which consumes communication bandwidth, memory, and CPU in both control and data plane of the SDN. Moreover, Since the OpenFlow switch buffers packet-in messages before sending them to the controller if several new flows are received within a very short time, the buffer fills up. This results in sending the entire packets of the new flow to the controller instead of sending the new flow's headers-only packet-in messages, thus resulting in higher consumption of the control plane's bandwidth and potentially delaying the installation of new flow rules received from the SDN controller. Another situation in which massive new flows could result is filling up the forwarding table of the OpenFlow switch. As mentioned earlier, such a table includes different flow rules that direct the switch about forwarding the packets and is updated and managed by the controller. Having several new flows results in the forwarding table of the switch. At some point, the forwarding table fills up, and therefore, upon receiving a new flow rule from the controller, it is unable to install it hence dropping the packet and sending an error message to the controller. Moreover, the switch would not be able to forward packets until there is free memory in its forwarding table, resulting in delays and dropping of incoming packets. On the controller side, a high arrival rate of packet-in messages exceeding the controller processing capability results in overwhelming the controller and making it unreachable to legitimate traffic. This could result in the failure of the entire network, as the controller implements the logic of the SDN and manages the application and the OpenFlow switches" [32].

Figure 2.6 represents the impact of DDoS attacks on a switch, a controller, and a southbound API. As it is depicted, both the attack traffic and the normal traffics are new flow for the OpenFlow switch. When the attacker sends an enormous amount of traffic the buffer and the flow table of the switch will be overwhelming and legitimate packets become rejected. Whenever the switch encounters new flow entries it is just forwarded to the controller. As a result, the southbound communication becomes exhausted and this situation creates DDoS on the communication channel. Finally, when a large amount of attack traffic reaches the remote controller, the controller becomes busy processing that illegitimate traffic and becomes overwhelmed to give legitimate service to legitimate hosts.

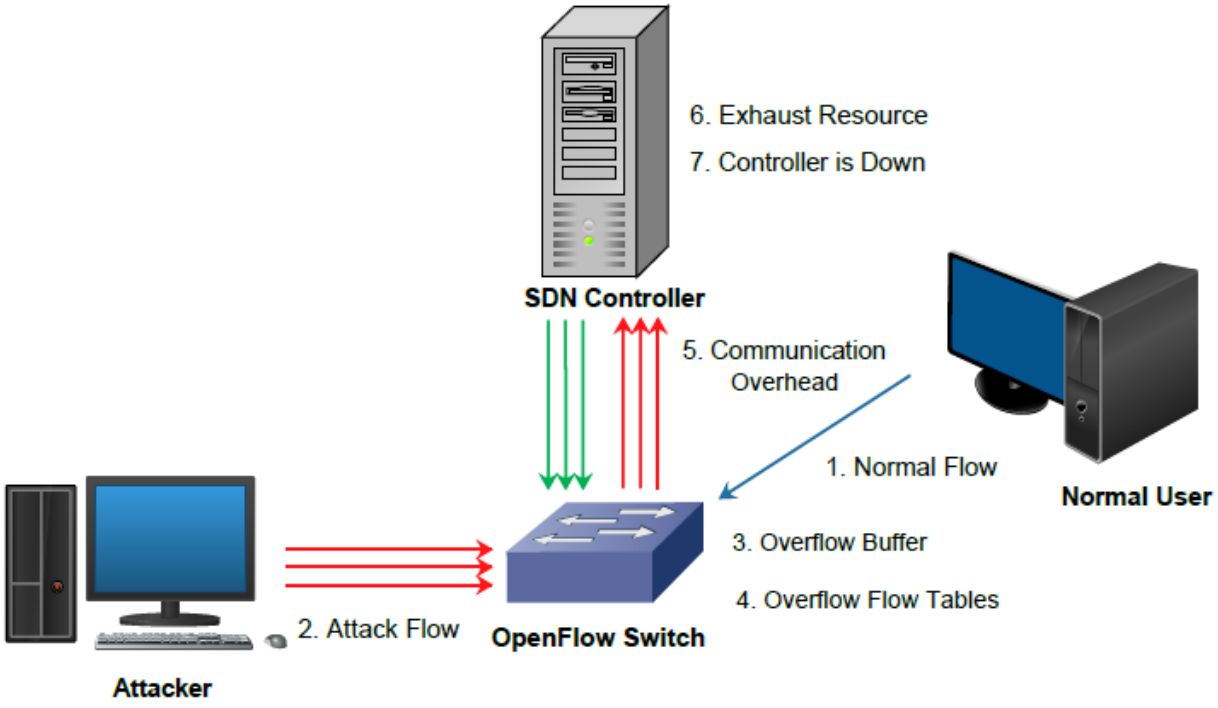

Figure 2.6: DDoS Attack on SDN

2.3 Machine Learning Techniques

ML techniques and algorithms like SVM, KNN, LR, ANN, K-means, etc. have been widely used to solve complex problems in engineering and science fields [40]. These ML approaches typically have two major phases. These are the training and decision-making phases. "During training, ML methods are applied to learn the system model using the training dataset. During the decision-making phase, the system can obtain the estimated output from each new input by using the trained model" [28]. When it comes to networking, "ML has the ability to effectively predict and efficiently schedule network resources based on massive data inputs" apart from identifying and detecting malicious activities in the network [26]. However, because of the distributed nature of conventional networks, "ML techniques are hard to be applied and deployed to control and operate the network". But, now SDN provides a new possibility to have intelligence inside the networks. Features like " a logically centralized controller, global view of the network, software-based traffic analysis, and dynamic updating of forwarding rules make it easier to apply ML techniques for SDN " [28].

2.3.1 Overview of Machine Learning Algorithms

ML algorithms can be classified with respect to different classification criteria. For instance, "according to task type, the ML models can be divided into regression models, classification models, and structured learning models" while according to the training method, ML models can be classified as supervised, unsupervised, semi-supervised learning, and reinforcement learning [26]. The regression model whose "output is a numerical value and cannot be enumerated is called the prediction model. Whereas the classification model is divided into binary classification models and multiple classification models." The supervised learning algorithms use a labeled training sample while the unsupervised learning models use unlabeled sample data for training the learning model. In addition, "semi-supervised learning algorithms develop mathematical models from incomplete training data, where some portion of the sample input data does not have labels. Unlike the ML methods just mentioned, reinforcement learning is concerned with how the agents ought to take actions in a specified environment to maximize some notion of cumulative reward" [26].

- **Supervised Learning:** "given a labeled training dataset these algorithms are used to build the system model representing the learned relationship between the input and output. After training, when a new input is fed into the system, the trained model can be used to get the expected output". KNN, DT, RF, NN, and SVM can be mentioned as an example of this learning technique [28,40].
- **Unsupervised Learning:** unlike supervised learning, these algorithms are given a set of inputs without labels. "Fundamentally, an unsupervised learning algorithm aims to find patterns, structures, or knowledge in unlabeled data by clustering sample data into different groups according to the similarity between them. Unsupervised learning techniques are widely used in clustering and data aggregation". K-means and SOM can be mentioned as examples of this learning technique [28].
- **Semi-supervised Learning:** "Because of the high cost of labeling data", there is only a small amount of labeled data in many practical problems, whereas "a large amount of unlabeled data is readily available". As a result, semi-supervised learning techniques advanced quickly. Semi-supervised learning makes use of both labeled and unlabeled data. As a result, it is a method of learning that combines supervised and unsupervised learning techniques. It focuses on how

to train and classify using a small number of labeled samples and a large number of unlabeled samples. Semi-supervised learning has the same applications as supervised learning [26,28].

- **Reinforcement Learning:** In this learning technique, there is "an agent, a state space, and an action space component. The agent is a learning entity that interacts with its surroundings to determine the best action to take to maximize its long-term reward. The long-term reward is a discounted cumulative reward that includes both immediate and future rewards" [28].

Table 2.2: Summary of ML Types Processess and Techniques

ML Types	Processes	Example ML Techniques
Supervised	Regression and Classification	ANN, RF, RNN, LR, KNN, SVM, etc.
Unsupervised	Clustering and Association	K-means, Ant colony.
Reinforcement Learning	Classification, Control, and Regression.	Q-learning, Deep Feed Forward Neural Network.

2.3.2 Roles of Machine Learning on Software-Defined Networks

The main objective of networking in recent years is to create intelligent networking architecture that allows for as much intellectualization, activation, and customization as possible [26]. "By performing data analysis, network optimization, and automated provision of network services, ML techniques can provide intelligence to SDN controllers. In other words, the learning capability enables the SDN controller to autonomously learn to make optimal network decisions" [28]. When it comes to increasing the capacity of network equipment, ensuring high-quality computer network operation, and enhancing maintenance and management capabilities of networks, the use of "intelligent tools and technologies to improve the overall performance of the network is extremely important" [26]. In this regard, it is vital and suitable to apply ML techniques in SDN. There are two main enabling factors for this to happen. First, advancements in computing technologies like graphical processing unit (GPU) and tensor processing unit (TPU) create a worthy opportunity to apply promising ML techniques. Second, the inherent nature of the SDN architecture enables the controller to have a global view of the network which actually eradicates the difficulties to use ML on traditional distributed networks [28,30]. The distributed network architecture which makes each node only have a partial view of the entire system makes it difficult to apply ML to the network.

On the contrary, “the centralized SDN controller has a global network view and can collect various network data, which facilitates the applications of ML techniques in SDN”. Below are some of the uses of ML on the SDN.

- **Traffic Classification**: This is “an important network function, which provides a way to perform fine-grained network management by identifying different traffic flow types. With the help of traffic classification, network operators can handle different services and allocate network resources more efficiently. The widely-used traffic classification techniques include a port-based approach, Deep Packet Inspection (DPI), and ML techniques”. But, because of the shortcomings of the Port-based techniques and DPI together with the growth of the complexity and dynamism of the network, these methods can’t be efficient anymore. For instance, nowadays applications have dynamic ports, and the radical growth of applications makes it difficult for DPI techniques to be more accurate. As a result, implementing ML-based approaches is necessary for the job. Furthermore, the global network view of the SDN controller helps the ML models to have global control over the network in such a way that traffic collection and analysis can be enormously facilitated [28].
- **Routing Optimization:** One of the fundamental functions in networking is routing. “In SDN, the controller can control the routing of the traffic flow by modifying flow tables in switches”. The most widely used routing optimizations are Shortest Path First (SPF) and the heuristic algorithm, but because of the inefficiency of SPF and the high computational cost of the heuristic algorithm, ML-based approaches are implemented in the SDN controller to create the routing policies for new flow entries [28].
- **Quality of Service (QoS)/ Quality of Experience (QoE) prediction:** “QoS parameters such as loss rate, delay, jitter, and throughput are network-oriented metrics that network operators typically use to assess network performance. Because SDN is a centralized architecture that can collect statistics from switches at per-port and per-flow granularity levels it is possible to apply ML algorithms that can be used to predict QoS/QoE” [28].
- **Resource Management:** One of the primary requirements of network operators to improve network performance is efficient and effective network resource management. In this regard, “SDN segregates the control and data plane, to make the network programmable with a centralized controller and global network view”. As a result, ML-based approaches can be used to facilitate the resource management work of the SDN [28].

- **Security:** “Security is always an important aspect that needs to be considered by network operators. Only secure networks can be accepted and used by users”. DDoS attack detection is one of the important aspects of network security among others. In this regard, ML-based approaches have been widely implemented in recent years in SDN network architecture [28].

2.3.3 Role of Machine Learning Techniques to Detect DDoS on SDN

AI plays significant and enormous roles in determining the costs and operations of today’s advanced and complex networks. As a result, its huge contribution to the network security domain has been efficient and effective. “Detection of complex attack patterns and network optimization are obviously extremely important applications of AI.” ML which is one of the most important AI mechanisms has been widely used for several classification and prediction problems and has provided accurate results. On the other hand, because of “the flexible and multidimensional nature of the SDN architecture unlike the traditional network architecture, combined with ML methods for extracting and analyzing network data, the SDN controller will allow the detection of anomalies in general and DDoS specifically.” Thus, the security of the network is significantly enhanced and guaranteed [26].

According to Eliyan and Pietro [32], solutions that address DDoS attacks can be categorized into two. These are intrinsic solutions and extrinsic solutions. “The focus of the intrinsic solution is on the components of the SDN and the functional elements while the focus of the extrinsic solution is on the network flows and their characteristics and features. These groups can be sub-categorized further. The sub-categories of the intrinsic solutions are table-entry solutions, scheduling-based solutions, and architecture-based solutions, while the extrinsic solutions are sub-categorized into statistics-based solutions and ML solutions.” In the ML approach, the defense mechanism is to train an ML algorithm on anomalous and attack-free flows in such a way that the ML model will be able to identify between attacks and normal traffics for the new unseen data [32].

2.3.4 Ensemble-Based Machine Learning Techniques

By combining predictions from two or more base models, ensemble learning techniques have achieved state-of-the-art performance in a variety of ML applications [41]. Despite their significant success in knowledge discovery, traditional ML methods may fail to produce satisfactory results when dealing with complex data, such as imbalanced, high-dimensional, noisy data, and so on.

This is because it is difficult for these methods to capture multiple data characteristics and underlying data structures [42]. Ensemble learning begins by extracting a set of features using various transformations. Multiple learning algorithms are used to produce weak predictive results based on these learned features. Finally, ensemble learning combines the informative knowledge obtained from the preceding results to achieve knowledge discovery and improved predictive performance through voting schemes in an adaptive manner [9,26,42].

Due to the dominance of DL in various AI applications, ensemble-based deep neural networks have recently demonstrated significant performance in improving the generalization of the learning system. However, because modern deep neural networks typically have millions to billions of parameters, the time and space overheads for training multiple base deep learners and testing with the ensemble deep learner are significantly higher than those for traditional ensemble learning. Though several fast ensemble deep learning algorithms have been proposed to promote the deployment of ensemble deep learning in some applications, further advances are still required for many applications in specific fields where developing time and computing resources are usually limited or the data to be processed is of high dimensionality [43].

Model complexity that is affected by model bias-variance tradeoff is one of the important aspects of ML in general and ensemble-based techniques specifically as these techniques create individual weak learners from the underfit or overfit model complexity and modified it until minimal total error is obtained. From Figure 2.7, we can observe that moving from left to right, the total error of the learning model declines continuously until reaching the bottom, followed by a rapid upward trend when the model complexity increases. The trends of this bias and variance are opposite in such a way that the former drop dramatically before remaining steady, while the latter holds steady before rising heavily. As a result, it is possible to deduce that when increasing the model complexity to improve the performance of this model, it is important to consider to reach to a delicate balance between bias and variance [42].

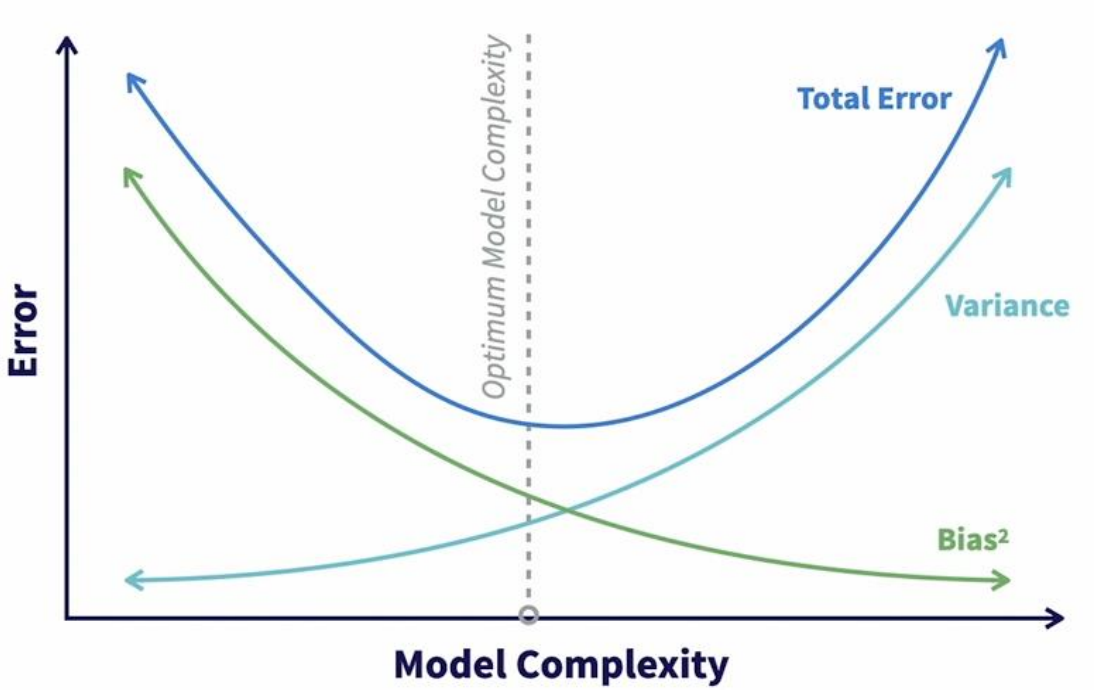

Figure 2.7: Bias Variance Tradeoff [42,44]

Base learners, also known as weak learners, are obtained by running the base learning algorithm on training data. In most instances of ensemble learning constructs, the single base learning algorithms are referred to as homogenous ensembles; however, in some instances, it constructs by employing multiple learning algorithms, which are referred to as heterogeneous ensembles. The benefit of ensemble learning is that it can boost weak learners, increasing the overall accuracy of the learning algorithm on training data [45]. As it is shown in Figure 2.8, on the left side, a simple underfit weak learning model with low variance and high bias is created. As a result, high total error, and simple model complexity is created. On the right side, complex overfit learning model with low bias and high variance. In this regard, ensemble-based techniques modified the overfit weak learner or the underfit weak learner and create other weak learners in such a way that the cumulative results of the combined weak learners yield a minimum total error of the model that is intended to achieve.

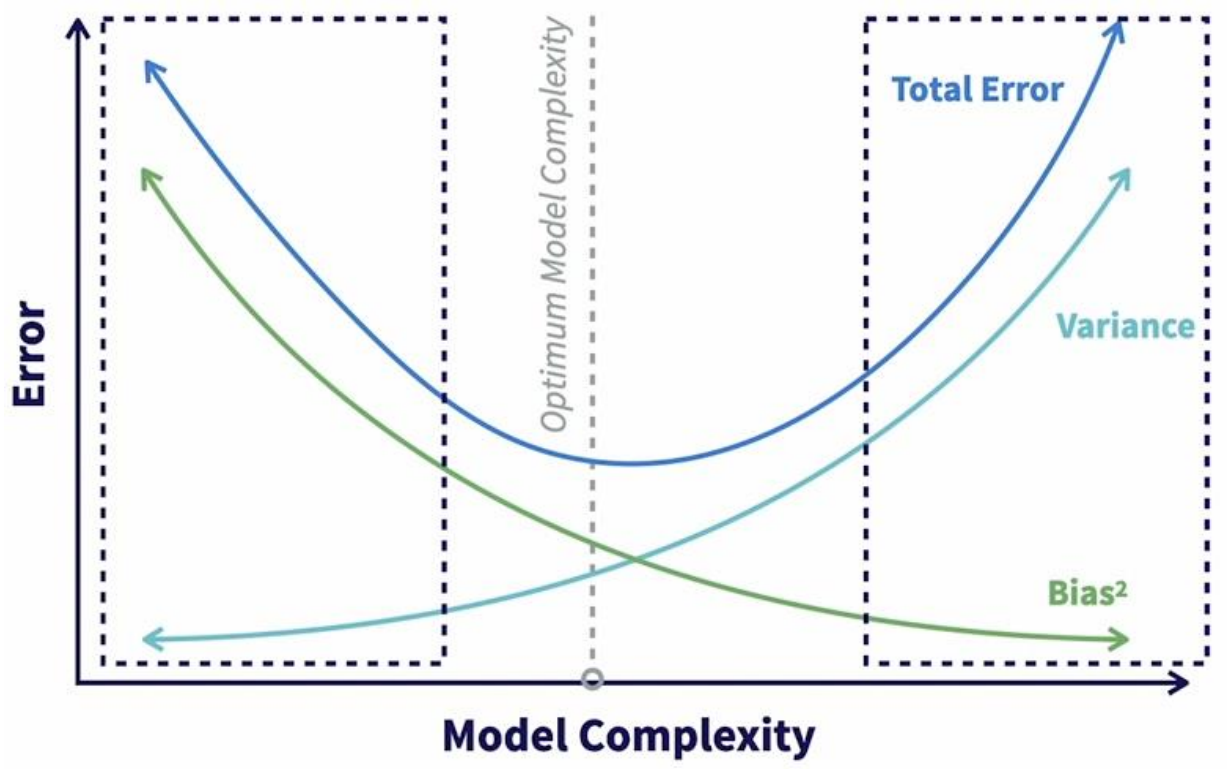

Figure 2.8: Model Underfit and Overfit

Ensemble-based learning techniques are categorized as Boosting, Bagging, and Stacking. Some researches show the bagging approach and boosting approaches outperform other conventional ML methods with a confidence level of more than 99.5% [26]. Below a brief overview of each technique is discussed.

- **Boosting algorithm:** This algorithm trains the best learner using the initial training data set and adjusts the weight of the training sample based on the prediction result of the base learner. We should especially increase the weight of the sample dealing with the incorrect result. Increase your focus on the incorrectly "predicted samples, redistribute the sample weights, run the next round of training, re-train the base learner, and use the test set to get the final prediction result" as it is illustrated in Figure 2.9 [9].

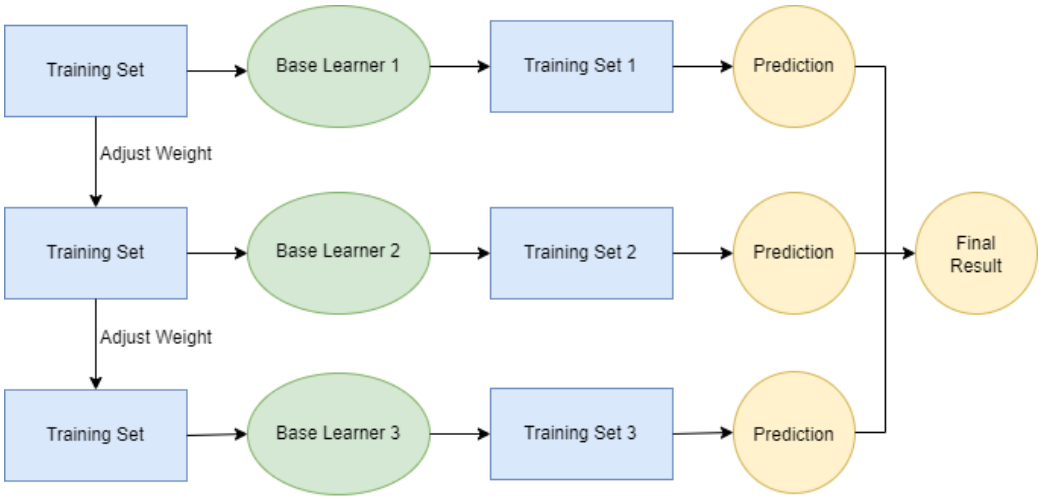

Figure 2.9: Boosting Technique of Ensemble Learning [9]

- **Bagging algorithm:** by sampling the training samples repeatedly different training sets can be obtained. "One of the sample sets will be collected by randomly extracting K samples from the training set and placing them in the sampling set." We can obtain N sampling sets after repeating the Process N times. Each sample set is used to train each base learner. In general, when dealing with the classification task, we use the voting method to combine these N single learners, and when dealing with the regression task, we use the average method to combine these N base learners [9]. Figure 2.10 illustrates the bagging ensemble techinqe discussed.

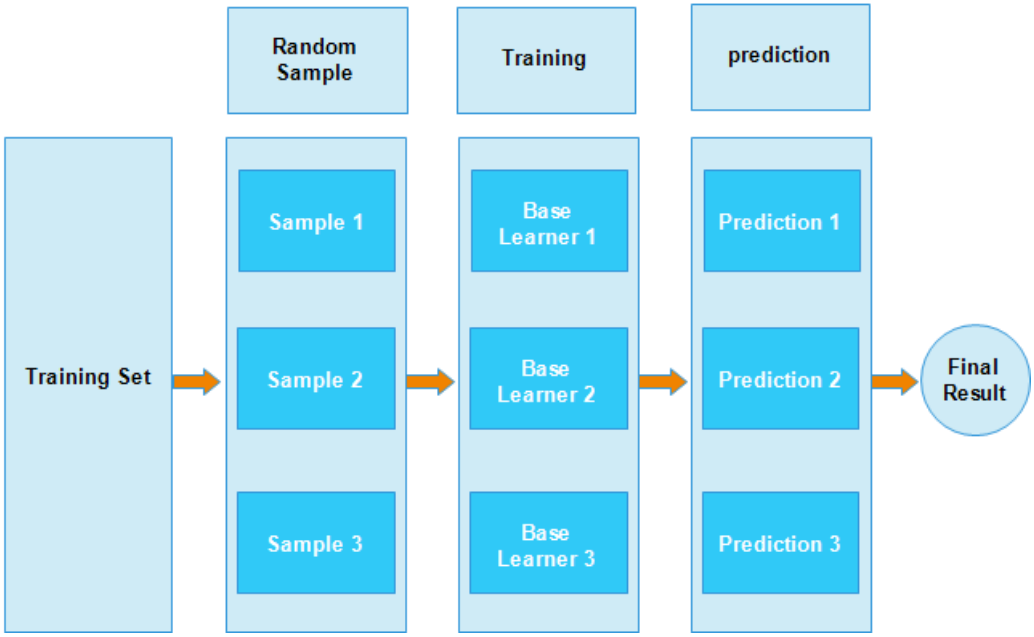

Figure 2.10: Bagging Technique of Ensemble Learning

- **Stacking Algorithm:** The stacking algorithm works by using the initial training set to train multiple base learners, who then generate a new sample set to train secondary learners as it is shown in Figure 2.11. “In the first level of the stacking algorithm, there are various types of base learners, and the correlation between base learner algorithms is low. Each base model’s prediction result is used as a feature of the training data for the secondary learner, and the label of the new sample remains the same as the label of the original sample. Following the training of all basic models, a new data set is generated for secondary learner training. Similarly, the test set trains each base model, and the average value of the prediction results is used as a feature of the new sample before predicting the secondary learners” [9].

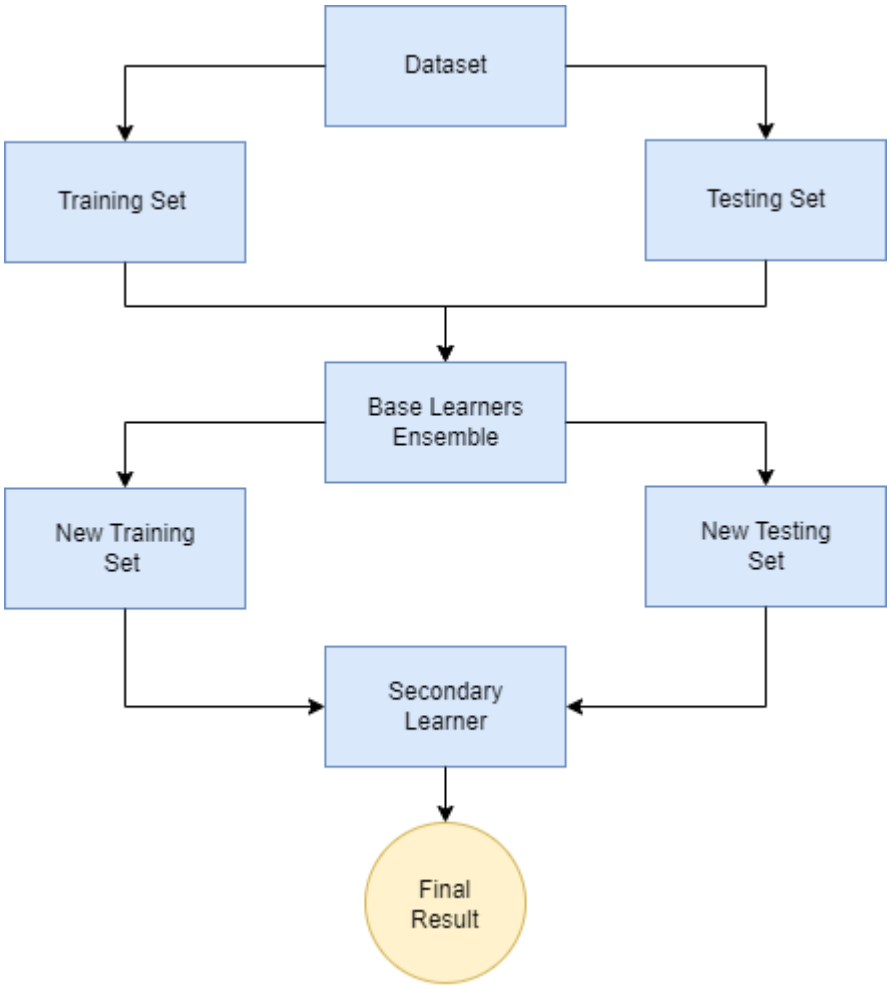

Figure 2.11: Stacking Technique of Ensemble Learning [9]

CHAPTER THREE: RELATED WORK

This chapter presents previous research works related to DDoS detection for SDN. The corresponding methods, tools, techniques, results gained are coverd, In addition, the challenges related to the methods are discussed.

3.1 Statistical Approaches

Bavani et al. [6] use the entropy-based technique to detect DDoS attacks for SDN. This research work determines the mean entropy using a statistical approach and the rate of packet drops to identify the occurrence of a DDoS attack. In this research work, first, the entropy value for all the normal packets is calculated. Then the attack packets are launched to a particular destination host. Finally, the entropy value of the attack packets is calculated. This research work calculates the entropy value after receiving the first 50 packets for 50 windows. The entropy value is calculated by averaging the entropy value of each of the 50 windows and compared with a threshold entropy value which is 1.5 to determine the availability of DDoS. The major shortcoming of this research work is that selecting a particular window size and a threshold value is very subjective and depends on the experience of the researcher.

Yadav et al. [46], proposed entropy-based DDoS detection for the SDN. In this research, a DDoS attack detection and mitigation algorithm is designed. This research work proposes a traffic generation module to generate normal traffic, an attack launcher to generate attack traffic, a time interval calculator module to reset the entropy, and entropy calculator modules to generate entropy. If entropy greater than the threshold value, then DDoS is detected and the mitigation module of the scheme will block the particular port where the DDoS traffic is originated from. Similar to the previous research work, this particular research work is also subject to a random selection of a threshold value to compare and contrast it with the calculated entropy value to decide whether DDoS occurs or not.

Lotlikar and Shah [47], consider attacks at the data forwarding layer and the control layer. Rate limiting is the first strategy that is applied to prevent attacks on the control layer bandwidth. In this strategic technique, a threshold limit of traffic that the server would be able to withstand is set. Another strategy is to prevent the switch's flow table from being flooded with flow rules. Each flow rule is assigned two different timeout values, idle timeout sets up to indicate the flow rule is

inactive, and hard timeout sets up to indicate the expiry of allocated time for that particular flow rule. So, according to this research removing flow rules with specific time out avoids the switch's flow table from being overflowed. As it is discussed by the authors of the paper, beyond the specification of a subjective threshold value that is highly affected by the experience of the researchers, limiting the rate of traffic flow certainly affects all the flow rules as there will be legitimate nodes as well.

3.2 Machine Learning Approaches

Ahmad et al. [48], present a particular setup where the SDN controller is exposed to DDoS attacks to draw important conclusions. According to the researchers, the controller of the SDN paradigm is the most attractive layer for DDoS attacks. As a result, these authors evaluate four ML techniques to detect DDoS attacks on the SDN controller layer. SVM, NB, DT, and LR are the four ML techniques that are selected to be experimented with. This research work created a dataset with traffic flow that is generated with Scapy. In this research work, the UDP flood attack is used for the DDoS attack alongside normal traffic. The dataset is marked with normal and attack traffic. The experiment result shows that SVM has an accuracy of 97.5%. Though there are various volumetric DDoS attacks, this research work considers only UDP flood attacks. As a result, the proposed models are not capable of detecting other Volumetric-based DDoS attacks, and this can be the main limitation of this research work.

Cui et al. [49], conducted research work to detect and defend the SDN controller against DDoS attacks using the K-Means algorithm of conventional unsupervised ML. The proposed system by these researchers is composed of two parts. The first part is designed for attack detection and location. This component is implemented with the K-Means clustering algorithm and the traffic dictionary. The traffic dictionary is used to inquire about the traffic pattern of individual OpenFlow switches and locate the switches that are under attack. The second part of the system is for defense purposes. The research work is evaluated in a simulated environment with a Floodlight controller and Mininet. This research work uses the DARPA 1999 Intrusion Detection Data Set to evaluate the feasibility of the proposed solution. One of the shortcomings of this dataset is that it is not generated from the SDN network. Besides, the DARPA 1999 dataset is outdated in a way that it doesn't represent today's complex network traffic and protocols.

Polat et al. [50] study the SVM, KNN, ANN, and NB models to detect DDoS in SDN through feature selection methods. Filter-Based feature selection, Wrapper-Based feature selection, and Embedded-Based feature selection methods are designed and experimented with the above ML algorithms. According to the researchers, the pivotal objective of this particular research work is to reduce the features that have a low impact on the classification and to provide the use of highly effective features. In this research, the researchers prepare experimental SDN topology to generate and collect both attack traffic and normal traffic. The attack traffic consists of TCP flood, UDP flood, and ICMP flood attack. Generally, a dataset having 65,000 samples related to network traffic flow at the moment of the DDoS attack, was generated. The normal traffic consists of 64,000 samples also collected with the same approach. The experimental result of this research shows the KNN classifier with wrapper feature selection method and with only six features has the highest accuracy of 98.30%. But, this model has a 6% false negative rate when detecting TCP flood attacks. Besides, it classifies 5.4% of TCP flood attacks as ICMP flood attacks. In general, compared to state-of-the-art techniques like DL and ensemble-based approaches, there is still room to enhance the efficiency of this approach in addition to the fact that the feature selection mechanism is cumbersome and time taking.

Kyaw et al. [13]. developed a machine learning-based DDoS attack classifier for SDN by utilizing polynomial SVM and compared it to existing linear SVM. For dataset generation and testing of the proposed approach, the researchers created a simulated SDN topology using the Ryu controller and Miniedit in the Mininet Emulator. This research creates a depth two network with nine switches and 64 hosts. The open virtual switch is used for the proposed system. The researchers generate normal traffic and UDP flooding attack traffic to simulate the normal situation and attack situation. Then in the feature extraction module of the proposed model, a feature set consisting of five features is extracted. The proposed polynomial SVM approach gains an accuracy of 95.38% accuracy. Though the proposed polynomial SVM technique scores a better accuracy compared with the traditional SVM technique, it needs improvement compared to state-of-the-art techniques like DL and ensemble-based techniques. In addition, the research work considers only UDP flood attacks. As a result, the proposed model cannot detect other volumetric-based DDoS attacks.

3.3 Deep Learning Approaches

Ahuja et al. [51], proposed DL-based approaches to detect DDoS on SDN. This research work experimented with CNN, Long Short Term Memory (LSTM), CNN-LSTM, Stacked Auto Encoder-Multi-Layer Perceptron (SAE-MLP), and Support Vector Classifier-Self Organized Map (SVC-SOM) models. The dataset consists of 22 features and is available in the Mendeley data repository. There are three protocols such as TCP, UDP, and ICMP in the dataset. In this research, the SAE-MLP classification algorithm has the highest performance with an accuracy rate of 99.75%. Though the result of the experiments show promising figures, the general fact that DL approaches consume more resources and time during training and prediction can be mentioned as one of the shortcomings of this particular research work. From another perspective, the nature of the dataset used is not well described, thus difficult to assess its negative and positive impact on the performance of the model.

Wang and Liu [52], proposed a DDoS attack detection method based on information entropy and DL. In this approach, suspicious traffics are inspected through an information entropy detection module in the controller. Then, a fine-grained packet-based detection is executed by the CNN model to distinguish between normal traffic and attack traffic. According to the research, this approach achieves 98.98 % of accuracy. The experiment of this research is conducted using the POX controller and Mininet environment to create a simulated SDN environment. The CICIDS2017 dataset is used for training and testing purposes. In this research, Hping3 is used to simulate DDoS attacks, while normal traffic is generated by a script. According to Elsayed et al. [53], The CICIDS2017 dataset has “288,602 missing class labels and 203 missing information instances” in addition to the extremely huge amount of redundant data. Furthermore, the dataset is generated from a conventional network thus the new protocols and attacks that can be encountered with the SDN architecture are not represented in the dataset.

Elsayed et al. [54], proposed “DDoSNet, an intrusion detection system against DDoS attacks in an SDN environment”. This method is based on the DL technique, combining the Recurrent Neural Network (RNN) with an autoencoder. The CICDDOS2019 dataset which contains a large amount of different DDoS attacks that can be carried out through application layer protocols using TCP/UDP is used. The attack dataset consists of UDP, SNMP, NetBIOS, LDAP, TFTP, NTP, SYN, WebDDoS, MSSQL, UDP-Lag, DNS, and, DDoS-based attacks. In the data preprocessing

stage socket features are removed to prevent the learning model from overfitting problems. According to the research, the proposed method has 99% of accuracy in the detection of DDoS for an SDN network. According to the authors, the original dataset used in the experiment has large missing and infinite values. As a result, in addition to the poor quality of the dataset, the general fact that DL techniques need more resources and computational cost, according to [20], is the major limitation of this research work.

3.4 Ensemble-Based Approaches

Deepa et al. [55], propose and design ensemble learning methods for DDoS detection in SDN environment. In this research work, the researchers propose an ensemble technique by adopting different ML algorithms. The proposed algorithms are KNN, NB, SVM, and Self-Organizing Map (SOM). This particular research work implements ensembles as an approach to detect DDoS attacks in an SDN environment. To handle DDoS attacks in an SDN environment, this research proposes a combination of two machine learning models KNN with SOM, NB with SOM, and SVM with SOM. For experimentation, Mininet an emulation tool for SDN is used. The emulator is used to create virtual hosts, controllers, and switches. According to the research, the ensemble approach outperforms compared to the original KNN, NB, SVM, and SOM classifiers. The SVM-SOM has a higher accuracy of 98.12%. This research work uses the Center for Applied Internet Data Analysis (CAIDA 2016) dataset that contains TCP, UDP, and ICMP packets. This dataset is not generated from SDN networks thus it doesn't represent the protocols and attack scenarios of the SDN network architecture paradigm. Besides, the researchers use a threshold time value from the source to the destination of the packet to differentiate the attack from the legitimate traffic.

Shirmarz et al. [56] proposed a DDoS attack discriminator that is designed for SDN. In this research work, an ensemble classifier is proposed to improve the DDoS flow detection accuracy. Three classification algorithms, including Decision Tree (DT), Neural Network (NN), and SVM are used to discriminate the DDoS. Finally, a boosting ensemble which is the combination of these three algorithms is proposed to improve the classification accuracy. This research work uses the CICDOS2019 dataset to train and experiment with the proposed models. The research uses 90% of the dataset portion for training and the rest 10% for testing purposes. To extract the features from the dataset the CICFlowMeter two-way traffic flow generation tool was used. The output of the tool contains 76 features and only 24 features are used to train the model. This dataset has two

classes that consist of normal flow or DDoS malicious flow which had been labeled before. Because the attack flows are less than the normal flows the dataset is unbalanced, the research uses weight sampling to make the samples balanced. This research work has an accuracy of 99.4%. Though there are at least three main ensemble learning models, this research work addresses only boosting techniques, and finding the wisdom from the other ensemble techniques for this particular domain is a gap that needs to be investigated.

Haider et al. [18], proposed a CNN-based ensemble framework for efficient DDoS attack detection in SDN. The proposed framework is evaluated on a current state-of-the-art Flow-based dataset under established benchmarks. This work proposed a novel approach to utilizing a DL-based ensemble. The CICIDS-2017 dataset that is used in this research work is fully labeled and comprised of at least 80 features of network traffic which includes both benign and multiple types of attack traffic. CICFlowMeter is used to extract and calculate features. A total of 140,000 samples are loaded for classification with a distribution of 60:40 for benign and DDoS traffic. The dataset is classified as a training and testing dataset with an 80% and 20% of ratio respectively. In this research, two similar DL models are combined to build up a new ensemble model. In this research high detection accuracy is demonstrated in a combination of two single CNN models (CNN ensemble). In terms of anomaly detection, this ensemble-based learner has an accuracy of 99.4%. But, according to Yang et al. [43], because modern DL has millions and billions of parameters, the time and space overhead from training multiple base learners and testing with ensemble DL are costly compaired to training a single learner.

Mbasuva and Zodi [20], proposed an ensemble deep learning intrusion detection system to detect DDoS attacks in SDN. The proposed approach builds an ensemble of CNN, Deep Neural Network (DNN), and Recurrent Neural Network (RNN) models. The proposed work uses the CIC-IDS2017 dataset. The dataset was loaded on the WEKA toolkit for the data cleaning process so that the missing values, NaN values, duplicates, and infinity values were able to replace by the mean values of the attribute it belongs. The architecture of the proposed ensemble model combines CNN, DNN, and RNN models to achieve the best performance and accuracy. For the purpose of evaluating the performance and efficiency of the proposed model, this research work uses evaluation metrics such as confusion matrix, loss model, detection accuracy with precision, recall, receiver operating characteristic, and under curve (ROC AUC) graph. According to the research, the CNN, DNN,

and RNN single models achieved accuracies of 98.8%, 98.27%, and 97.49% respectively. On the other hand, the research work stated that the proposed ensemble CNN+RNN+DNN has the highest accuracy of 99.05% detection rate. The CIC-IDS2017 which doesn't represent the SDN network paradigm, and it is the limitation of this research work. Besides, DL has millions and billions of parameters [43], and using ensembles of them multiplies the number of parammeters with the number of base learners. This makes the computational cost to be too expensive.

Firdaus et al. [57] proposed DDoS detection using ML techniques with ensemble learners. At the experimental stage, the researchers use InSDN as a dataset. This study consists of two stages. The first stage is the clustering and classification stage. This stage goes into two processes. These are feature selection and normalization and algorithm clustering and classification. While the second stage is ensemble algorithm clustering and classification. This research work implements a simulation of the proposed work using the Mininet emulator. This research work proposes K-means++ and Random Forest (RF) algorithms. K-means++ is used for clustering and dividing the training data into small parts. The purpose of this algorithm is to find groups in the data with the number of groups initialized by variable K. Next the RF classifier is used to detect the DDoS attack. According to the research, RF is an ensemble of the decision tree. For experiment, the authors create virtualized SDN with eight hosts where six hosts are legitimate while two hosts configured as attackers, and an accuracy of 100% is obtained. This research uses eight features to train and test the proposed models. Out of the eight features, six of them are socket features, which according to [53,54], make a model overfit. Therefore this might be the reason behind the accuracy of the proposed model.

3.5 Summary

In summary, while some researches have explored the use of statistical and traditional ML techniques, as well as state-of-the-art approaches like DL and ensemble-based techniques, signigicant gaps still exist. These gaps include issues related to data quality, outdated and non-SDN representative data, efficiency, performance, and coverage of volumentric DDoS attack types. Furthermore, given that ensemble-based models can achive high accuracy comparable to DL approaches while consuming fewer resources, there is ample opportunity for further experimentation to solve the listed gaps. Therefore, this research work aims to contribute to these aspects. Table 3.1 provides a comprehensive summary of the reviewed related work.

Table 3.1: Summary of Related Work

Techniques	Publication	Dataset	Gaps
Statistical Approach	Bavani et al. [6]	-	Subjective window size and threshold value is.
	Yadav et al. [46]	-	Selecting of threshold value is subjective.
	Lotlikar and Shah [47]	-	Selecting of threshold value is subjective. Limiting the rate of traffic flow affects legitimate traffics.
Machine Learning Approach	Ahmad et al. [48]	Self-generated	Only UDP flood attack is considerd. As a result, other volumetric DDoS attacks are not detected.
	Cui et al. [49]	DARPA 1999	Non-SDN representative and outdated dataset.
	Polat et al. [50]	Self-generated	Low efficiency compared with DL and ensemble approaches.
	Kyaw et al. [13]	Self-generated	Only UDP flood attack is considerd, and low accuracy compared with DL and ensemble approaches.
Deep Learning Approach	Ahuja et al. [51]	Mendeley	Dataset is not described. Besides, DL consume more computational cost.
	Wang and Liu [52]	CICIDS 2017	Poor quality and outdated dataset. Besides DL consume more computational resources.
	Elsayed et al. [54]	CICDDoS 2019	Poor dataset quality and high computational resource consumption.
Ensemble-Based Approach	Deepa et al. [55]	CAIDA 2016	Poor and non-SDN representative dataset, and usage of threshold value which is subjective.
	Shirmarz et al. [56]	CICDDoS 2019	Only boosting technique of ensemble is addressed.
	Haider et al. [18]	CICIDS 2017	Time and space overhead.
	Mbasuva and Zodi [20]	CICIDS 2017	Time and space overhead.
	Firdaus et al. [57]	InSDN	Usage of socket-based features that makes models overfit.

CHAPTER FOUR: THE PROPOSED SOLUTION

This particular chapter of the research work discusses the proposed solution and its architecture. In this regard, exploring and cleaning the data, splitting it into training and testing sets, fitting and evaluating the initial training model, tuning hyperparameters, evaluating on a testing dataset, are some of the major activities under the proposed solution.

4.1 Proposed Architecture

The proposed architecture for the ensemble-based DDoS detection model in SDN comprises several components, including data preparation, data pre-processing, train-test split, and fit and evaluate. The architecture has been inspired by a literature survey conducted in the field. As Figure 4.1 depicts, the data preparation component is one of the components of the proposed architecture. Since the dataset that is used in this research work is a comprehensive dataset that is used for IDS detection for SDN and it is placed in different files according to the type and group of the data, we need to prepare the dataset for the particular objective of the proposed research work. Therefore, selecting the DDoS attack from the InSDN dataset metasploitable group and OVS group and combining the result from the two groups will yield a DDoS dataset. Then, combining the result with the normal group of the dataset gives a final raw dataset.

The data-preprocessing component of the architecture is composed of activities like dropping unnecessary variables, and one-hot encoding as such the previous raw data can be cleaned and be ready for the train-test split component of the architecture of the proposed solution. This component will split the cleaned dataset into training, and testing sets. The training sets of the proposed solution are also further split into five-fold cross-validation sets so that several ML models on subsets of the variable input data will be trained and evaluated to detect overfitting problems. Finally, the fit and evaluate component of the proposed architecture will train the proposed ensemble-based DoS detection models for SDN, and these models will be tested and evaluated on the testing dataset.

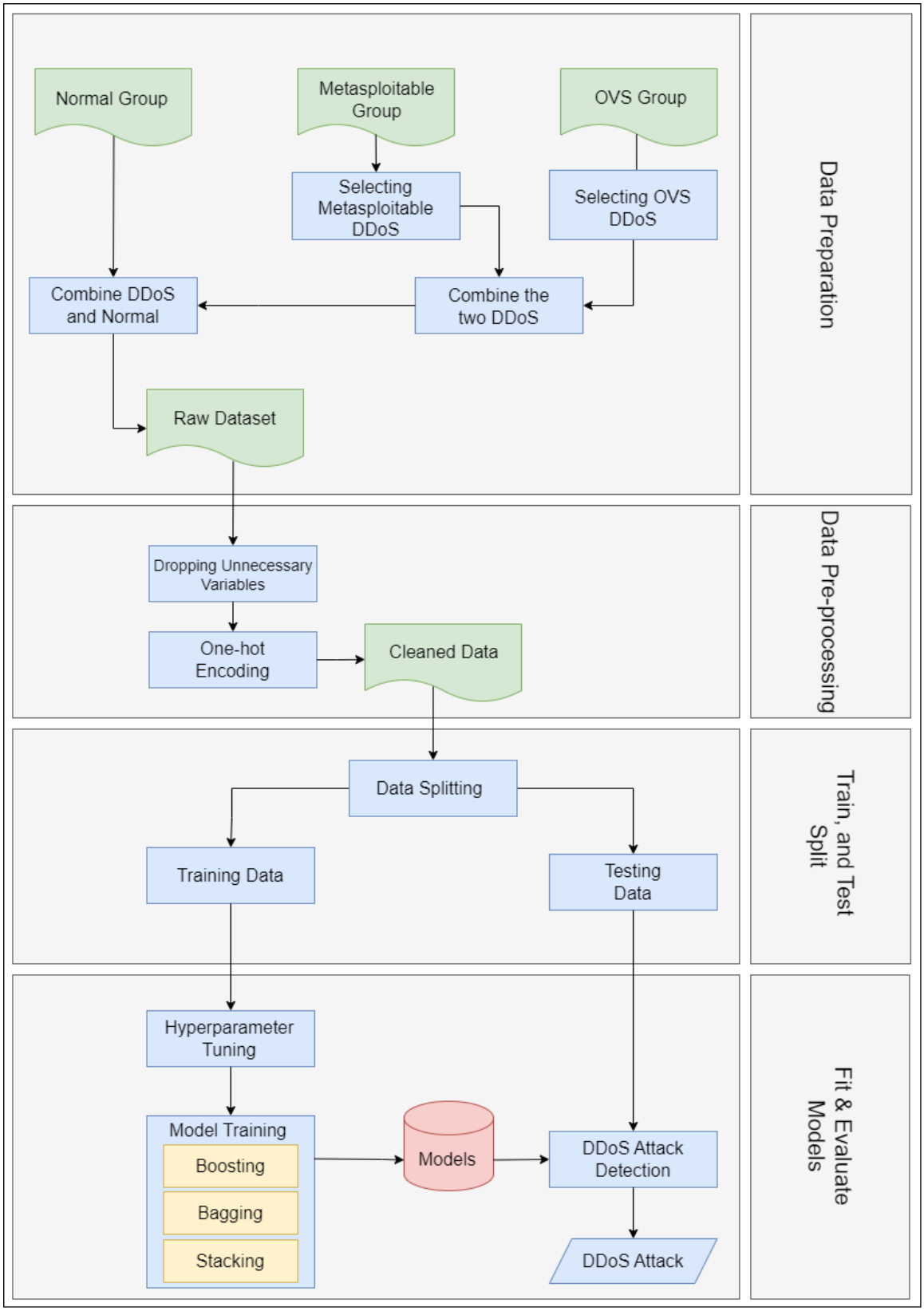

Figure 4.1: Proposed Arichitecute

4.2 Data Preparation

In any ML-based research, data preparation and pre-processing is one of the first and pivotal steps toward the success of the proposed solution [58]. This particular step and process help to create context-aware raw data that can be used for training and evaluating the proposed models. In this regard, because the InSDN dataset is a comprehensive SDN dataset that is generated under the consideration of having a generic IDS dataset, it is a must for us to make a change to it in a way that must fulfill our needs. “This dataset includes the normal and various attack categories that can occur in different elements of the SDN architecture.” DoS, DDoS, brute force attacks, web applications attacks, exploitation, probe, and botnet are attacks that are included in the dataset.

In addition to this, the generated normal traffic data in this dataset contains various popular application services such as HTTPS, HTTP, SSL, DNS, Email, FTP, and SSH. This dataset is “grouped into three based on the traffic types and the target machines. The first group includes the normal traffic only, the second group contains the attack traffics that target a metasploitable-2 vulnerable server.” The last group of the attack traffic is generated from the OpenFlow switch. Besides, according to Lyu and Lu [59], in SDN, traffic flow features are classified as packet-level features and flow-level features, and most research works that are discussed in the related work section use old datasets which do not include flow-level packet features into consideration. In this regard, this particular research work utilizes the enormous potential that the new InSDN dataset provides for the research community.

As a result, having the scope of the proposed research work in mind, it is a must to prepare the InSDN dataset in such a way that it satisfied our need for this particular research work. Thus, as it is indicated in the proposed solution architecture and Algorithm 4.1, the DDoS attack from the metasploitable-2 group and the OVS group of the InSDN dataset has to be segregated and combined to have one common DDoS attack dataset group. Then, the result of the combined DDoS attack dataset group has to be combined with that of the normal group of the InSDN dataset to have the raw dataset for us to train and test the proposed ensemble-based DDoS detection model for the SDN network architecture. Figure 4.2 presents a sample of the prepared dataset.

Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	...	Fwd Seg Size Min	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
185.127.17.56	443	192.168.20.133	53648	6	5/2/2020 13:58	245230	44	40	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Normal
192.168.20.133	53650	185.127.17.56	443	6	5/2/2020 13:58	1605449	107	149	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Normal
192.168.20.133	35108	192.168.20.2	53	6	5/2/2020 13:58	53078	5	5	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Normal
192.168.20.2	53	192.168.20.133	35108	6	5/2/2020 13:58	6975	1	1	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Normal
192.168.20.133	60900	154.59.122.74	443	6	5/2/2020 13:58	190141	13	16	...	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Normal

Figure 4.2: Sample Prepared Dataset

Input: normal_dataset, metasploitable_attack, ovs_attack;

Output: prepared_dataset;

```
START
Normal_dataset ← read normal dataset file
metasploitable_attack ← read metasploitable attack file
ovs_attack ← read OVS attack file
select DDoS attack from metasploitable_attack
select DDoS attack from ovs_attack
DDoS_attack ← combine the two DDoS attacks
prepared_dataset ← combine normal_dataset and ddos_attack
write prepared_dataset to file
END
```

Algorithm 4.1: Dataset Preparation

4.3 Dataset Pre-processing

Numerous factors affect the performance of prediction models, and the genuineness of the data is the pivotal one. In the presence of unnecessary and redundant information in the dataset, it becomes very difficult to discover valuable knowledge during the training phase. Hence, the outcome will be unreliable decisions and inaccurate analytics if one fails to conduct this phase carefully [58]. Having this in mind, we use different techniques to pre-process the raw dataset that is prepared from the InSDN dataset during the data preparation stage of the proposed solution architecture to make it suitable for training and testing purpose of the proposed ensemble-based DDoS detection models for SDN network architecture. In this regard, the following steps are used to pre-process the raw dataset that is going to be used as input for the training and testing stage of the proposed solution architecture. Furthermore, Algorithm 4.2 gives a more detailed view of the data pre-processing.

Concerning data cleaning, which is the process of removing or filling missing values, removing unwanted, and duplicated information from the dataset before it feeds to the learning model is one of the data pre-processing components. But, since the InSDN dataset has no missing and infinite values the data pre-processing component is composed of dropping unnecessary variables and encoding the label data.

- **Drop unnecessary variables:** The original InSDN dataset has 83 features. Though all these features can be used for classification, their importance is not equal [56]. As a result, all important feature sets that are extremely valuable for the specified problem have to be identified and selected for further process.
- **Encoding the label:** We trained our model for binary classification to classify the input traffic as normal or malicious. Therefore, we consider all DDoS classes as an attack category. Then we are encoding the string value from normal and attack labels to binary values of 0 and 1 for normal and attack traffic respectively.

Input: prepared_dataset

Output: cleanded_dataset;

```
START
#use panda to read prepared_dataset file
df ← read(prepared_dataset)
#drop irrelevant values from the dataset.
df.drop([irrelevant_values])
#one-hot encoding
change normal to 0 and attack to 1
write file to cleand_dataset
END
```

Algorithm 4.2: Data Pre-processing

4.4 Train, Test Split

Training and testing is the other component of the proposed solution architecture. In this phase, the already cleaned dataset from the previous step will be classified as a training set and testing set, and this process is further elaborated using Algorithm 4.3.

- **Training Data**: The selection of training samples is a very important aspect of ML data pipelining. This process yields a portion of the size of the dataset for model-fitting purposes.
- **Cross-valdation**: The training set of the dataset is further classified into five-fold cross-validation sets. According to Lyu and Lu [59], when n cross-fold validation set is applied, the training dataset is further separated into smaller n parts. Then each smaller dataset is used as a testing dataset while the rest n-1 dataset parts are used to train the model. This process is repeated until all the n smaller datasets are used as a testing set and every n-1 part as a training set. Finally, the average result is calculated from the individual n results that are yielded from the n-fold smaller dataset to calculate the final result of the proposed model.
- **Testing Data**: Testing any ML learning models on the data used for training can not provide realistic results concerning the models' performance, and this is the main reason for splitting the dataset into separate sets such as training and testing sets.

Input: prepared_dataset

Output: splited files;

```
START
#use panda to read cleand_dataset files
df ← read(cleaned_dataset)
features ← drop('Label')
labels ← df['Label']
#split 70% for training
X_train, X_test, y_train, y_test ←
split(features,labels,test_size=0.3)
END
```

Algorithm 4.3: Train-Test Split

4.5 Fit and Evaluate Models

This component of the proposed solution architecture is a vital phase of the research work. This includes activities such as fitting ensemble-based models, model testing, and model evaluation.

- **Hyperparameter Tuning:** Is another crucial aspect of the proposed ensemble-based DDoS detection model for SDN. Hyperparameter tuning involves optimizing the parameters that are not learned during the training process, such as the learning rate, regularization constants, and number of estimators in the ensemble models. By fine-tuning these hyperparameters, the model's performance and generalization ability can be improved, leading to more accurate and robust DDoS detection in SDN environments.
- **Model Training:** The main idea behind training a model is not to teach it to model the already known data but to generalize the data values that are currently unknown to it; this phenomenon is called generalization and it is an indicator of the model's performance for previously unseen data. Apart from the generalization capability of a model, the bias-variance trade-off is a very important concept to put into consideration during the implementation of ML-based research. Bias is an indication of how simple is a model. A model with having high bias will be much less complex as compared to a model with a low bias. In contrast, a model with high variance and low bias is a model much more complex. But, the ideal mode must have a low bias and

low variance. With this in mind, the boosting, bagging, and the stacking ensemble models will be trained to detect DDoS attack in SDN.

- **Evaluation:** There are various techniques to evaluate models' performance; model overfitting and training-validation loss are some of them. Whenever a model overfitted, the training loss will be low while the testing loss will be high. There are many ways to prevent a model from overfitting, and using more data, choosing a less complex model, feature removal, and regularization are some of them. In a contrary fashion to the phenomena of model overfitting, model underfitting is occurred due to low variance and high bias. If the loss in training and testing sets is significantly high, it is easily identified that the model has been subjected to a model underfitting problem. In addition, in the process of crafting ML models, evaluation is the vital point at which the model is tested concerning different performance indicators like accuracy and confusion matrix.
- **Performance:** The confusion matrix together with the accuracy, the precision, the recall, and F1-Score are the important performance evaluation metrics. A confusion matrix is a table that describes the classification results in detail, whether they are correctly or incorrectly classified, and different classes are distinguished, for binary classification, a 2 by 2 matrix, and for n classification, an n by n matrix [60]. According to Tigist Hidur [61], for binary classification, the result can be classified as:
  - **True Negative (TN):** "number of normal instances that are actually normal and the model predicted as normal."
  - **False Positive (FP):** "number of normal instances that are actually normal but the model incorrectly predicted as an attack."
  - **Fasle Negative (FN):** "number of attack instances that are actually attack but the model incorrectly predicted as normal."
  - **True Positive (TP):** "number of attack instances that are actually attack and the model correctly predicted as an attack."

The other important performance metric is accuracy which is an evaluation phenomenon used as an indicator of how the model is accurate in terms of its classification performance. The accuracy of a model is calculated as the ratio of the number of correctly classified samples to the total number of samples for a given test data set.

$$\text{Accuracy} = \frac{TP+TN}{\text{TP+TN+FP+FN}} \quad (1)$$

On the other hand, precision calculates the ratio of all correctly detected items to all detected items.

$$\text{Precision} = \frac{TP}{\text{TP+FP}} \quad (2)$$

Sensitivity or recall or True Positive Rate (TPR) calculates the ratio of all correctly detected items to all items that should be detected. The equation is given as:

$$\text{Recall} = \frac{TP}{\text{TP+FN}} \quad (3)$$

F1-Score is a very common technique to combine precision and recall into one metric, and calculated as:

$$F1\_Score = \frac{2(Recall*Precision)}{Recall+Precison} \quad (4)$$

Table 4.1: Confusion Matrix

	Predicted Positive	**Predicted Negative**
Labeled as Positive	True Positive (TP)	False Negative (FN)
Labeled as Negative	False Positive (FP)	True Negative (TN)

CHAPTER FIVE: EXPERIMENTATION AND EVALUATION

This chapter of the proposed research work gives enormous emphasis to a carefully crafted experiment on the proposed solution. The development setup, software and hardware tools, programming languages, and libraries that are used for the experimentation are briefly described. The dataset utilized and the corresponding pre-processing techniques will be illustrated. Finally, the implementation detail of the experiment and the evaluation of the different ensemble-based DDoS detection models together with a comparative based analysis with selected previous work results will be covered.

5.1 Development Environment and Tools

The proposed research work experiment utilizes several hardware and software components. A personal computer (PC) with Intel(R) Core(TM) i7-7500 cpu@2.7 GHz processor, 8 GB of RAM, and Windows 10 Enterprise 64-bit operating system is used. When it comes to the development environment, packages, and libraries used, below a brief overview is provided.

Python: python is one of the most important open-source programming languages which is developed under an OSI-approved open-source license.

Anaconda: There are many distributions of python. It is possible to either install python from the official python.org website or choose any popular distribution. For this research work, anaconda distribution is installed and configured which is free and open source. Anaconda simplifies the overall package management and deployment process and comes with a very easy-to-use virtual environment management and some additional tools for coding like Jupyter Notebooks, Visual Studio Code IDE, etc.

Jupyter NoteBook: to start coding and developing the proposed ensemble-based DDoS detection ML models for the SDN, we choose Jupyter Notebooks IDE which is provided by the Anaconda distribution. It is a web-based notebook IDE. As it is described above, this IDE is a preinstalled IDE with Anaconda Python distribution. To start Jupyter Notebooks, we can use the Anaconda Navigator.

Scikit-Learn: This is an open-source python library for ML-based research works and projects. It is mostly used to implement a wide range of algorithms to train and test models. This library module is a well-known and widely used tool in the ML ecosystem.

Numpy: It is another open-source python library that is used for scientific computing. The term numpy stands for Numerical Python. This library is used to perform the most complex mathematical operations that ML depends on. In scikit-learn NuPpy is used to change any data before its utilization into a NumPy array using this library.

Pandas: It is also another open-source python library that is used for data manipulation. It is the most powerful and fastest library for data manipulation. The main data structure in this library is called DataFrame which is similar to a table in an Excel spreadsheet. This library can work with several file formats like Comma Separated Values (CSV) among others. In addition, it can convert data into python objects with rows and columns called DataFrame which allows programming more efficient and easy.

MatPlotlib: is a widely used python library used for plotting images and data visualization. It generates two-dimensional plots from the given data like bar graphs, pie charts, and histograms.

5.2 Dataset Used

The dataset that is used to conduct the research work is called InSDN. This dataset is generated by [53] and reflects these days' Internet attacks that can be launched in the current SDN networks. Since it is a dataset for IDS, it consists of DoS, DDoS, Web Attacks, R2L, Malware, Probe, and U2R (Exploitation) datasets. This dataset is divided into three groups based on the traffic types and the target machines. The first group includes normal traffic only while the second group contains the attack traffics that targets the Metasploitable-2 server. The third group, on the other hand, considers attacks on the OVS machine. The total number of instances in the dataset are 343,939 for normal and attack traffic. The normal traffic is 68,424 while the attack traffic contains 136,743 for the Metasploitable-2 group and 138,772 for the OVS group. The dataset has 83 features and categorized as:

- **Network Identifiers attributes:** "Features contain the common information that is used to define the source and destination flow like IP address, Port number, and Protocol."
- **Packet-based attributes:** "Features that hold the information about the packets like the total number of packets in a forward and backward direction."
- **Bytes-based attributes:** "Features contain information related to the bytes like the total number of bytes in the forward and backward direction."

- **Interarrival time attributes:** “Features hold the information related to the interarrival time in both forward and backward directions.”
- **Flow timers attributes:** “Features that show the information related to the time of each flow like active and inactive.”
- **Flag attributes:** “Features that contain information related to flags like SYN Flag, RST flag, etc.”
- **Flow descriptors attribute:** “Features that contain the traffic flow information like the number of packets and bytes in both forward and backward directions.”
- **Subflow descriptors attributes:** “Features that hold the information related to subflows, such as the number of packets and bytes in forwarding and backward directions.”

5.3 Implementation Details

The main objective of this section is to illustrate how the experiment is conducted. As it is discussed in the proposed solution architecture, dataset preparation is the first stage in the implementation process. In this stage after importing the necessary libraries and dataset groups like the normal group, the metasploitable attack group, and the OVS attack group, the DDoS attacks are separated from other attack information of the InSDN dataset. Then the selected attack samples will be combined with the normal group samples to produce a raw dataset. The final result of this stage yields a raw dataset of 121,942 attack instances and 68,424 normal samples to train and test the proposed ensemble-based DDoS detection models for SDN. Annex A provides the source code implementation for this step.

The next process of the implementation is pre-processing the raw dataset from the previous step. In this stage, after checking for null values, irrelevant variables such as socket features “like source and destination IP, source and destination port, timestamp, and flow IDs are removed”. According to [53,54], these features vary from network to network, and the attacker can generate both normal traffic and attack traffic from the same IP. As a result, training the model with socket features can cause an overfitting problem as the model can be biased with the socket information. After dropping these and other unnecessary variables, we finally select 47 features from the raw dataset using Sickit-Learn Python library. The research work [53] is the base research work for us to select the features. Annex B shows the selected features. Then, we conduct a one-hot encoding to change

the ‘normal’ label to the number zero and the ‘attack’ label to the number one by using Sickit-Learn Python library. Annex C gives the source code for this process.

After the pre-processing stage is completed, the train-test split phase takes place. In this process, the pre-processed dataset is split into a training and testing set. 70% of the dataset is allocated for training and the rest 30% of the dataset is allocated for the testing set. The source code for this process is given in Annex D.

In the ‘Fit and Evaluate’ component of the proposed solution, we train adaptive boosting, gradient boosting, DT-based bagging, and KNN-DT- based stacking model. According to Xin et al. [60], KNN and DT are the most commonly used ML algorithms, and in the Scikit learn library DT is used as a default base learner for most ensemble-based classifiers. As a result, the KNN and the DT base learners are selected for this particular reason. As shown in Annex E, after we import the training data hyperparameters are used to tune the model and train it with a cross-validation of five folds. Table 5.1 shows the hyperparameters for the proposed ensemble-based algorithms.

The Adaptive Boosting algorithm is trained with 10,50, and 100 base estimators and 1,10, and 20 learning rates, and the best performance of the model is obtained with 10 estimators and a learning rate of One. The Gradient Boosting algorithm is also trained with a similar parameter to that of the Adaptive Boosting algorithm but with additional parameter called maximum depth. In this algorithm, the best model is obtained with 10 estimators, a learning rate of One, and Five maximum depth parameters. The Bagging ensemble-based technique used DT as a default base learner. In this algorithm, we use 10,50,100 estimators and 1,10,20 random state parameters. The best-performed model is obtained with 10 number of estimators and a random state parameter value of One. Finally, the fourth ensemble-based model that is designed in this experiment is the KNN-DT-based stacking model. This model is tuned with hyperparameters of several neighbors for the KNN model and the random state for the DT algorithm. The best performance of the model is obtained when the number of neighbors for the KNN algorithm is Five, and the random state for the DT algorithm is Three. For this ensemble-based algorithm, the LR is used as a final estimator.

Table 5.1: Hyperparameters

Ensemble-Algorithm	Parameters	Values	Best Values
Adaptive Boosting	n_estimators	[10,50,100]	10
	learning_rate	[1,10,20]	1
	Base estimator	DT	-
Gradient Boosting	n_estimators	[10,50,100]	10
	learning_rate	[1,10,20]	1
	max_depth	[10,50,100]	5
	Base estimator	DT	-
Bagging	n_estimators	[10,50,100]	10
	random_state	[1,10,20]	1
	Base estimator	DT	-
Stacking	knn__n_neighbors	[5,7]	5
	dt__random_state	[0,3]	3
	Base estimator	KNN and DT	-
	final_estimator	LR	-

5.4 Performance Evaluation Results

The proposed work develops four ensemble-based models to compare and contrast their performance. The adaptive boosting, gradient boosting, DT-based bagging, and KNN_DT-based stacking ensemble-based techniques are carefully crafted and experimented with using a testing dataset. Table 5.2 presents the metrics that are used for performance evaluation purpose of the proposed ensemble-based DDoS detection models for SDN networks, including accuracy, precision, recall, f1-score, and latency (prediction time).

The results show that the adaptive boosting model achives 100% accuracy, precision, reacall, and f1-score, with zero false positive and false negative values. The gradient boosting ensemble technique achives 99.99% accuracy, precision, and f1-score, and 100% recall. The DT-based bagging and DT_KNN-based stacking ensemble techniques also achive 99.99% accuracy, precision, recall, and f1-score. In terms of latency, the gradient boosting ensemble-based technique has the lowest prediction latency, with a value of 60.6 ms. On the other hand, the KNN_DT-based stacking model has the highest prediction time, with a value of 119,431.5 ms.

The performance of the models are also further evaluated using a confusion matrix. Figure 5.1-5.4 depict the confusion matrix results of each ensemble-based techniques on the testing set. The adaptive boosting ensemble technique accurately classifies all instances. The gradient boosting model misclassifies one instance out of 20,426 normal instances as an attack. For the DT-based bagging and DT-KNN-based stacking ensemble techniques, one instance is misclassified as an attack when it is actually normal, and one instance is misclassified as normal when it is actually an attack.

In summary, the ensemble-based models demonstrate strong performance in detecting DDoS attack in SDN networks, with the adaptive boosting model achieving perfect accuracy and the gradient boosting model exhibiting the lowest latency.

Table 5.2: Evaluation Metrics of the Proposed Ensemble-Based Models

Evaluation Metrix	Ensemble-Techniques			
	Adaptive Boosting	**Gradient Boosting**	**DT_Bagging**	**KNN_DT Stacking**
TN	20,426	20,426	20,426	20,426
FP	0	1	1	1
FN	0	0	1	1
TP	36,684	36,684	36,684	36,684
Accuracy	1.0	0.999982	0.999964	0.999964
Precision	1.0	0.999972	0.999972	0.999972
Recall	1.0	1.0	0.999972	0.999972
F1-Score	1.0	0.999985	0.999971	0.999971
Latency (Prediction Time)	501.6 ms	60.6 ms	145.5 ms	119, 431.5 ms

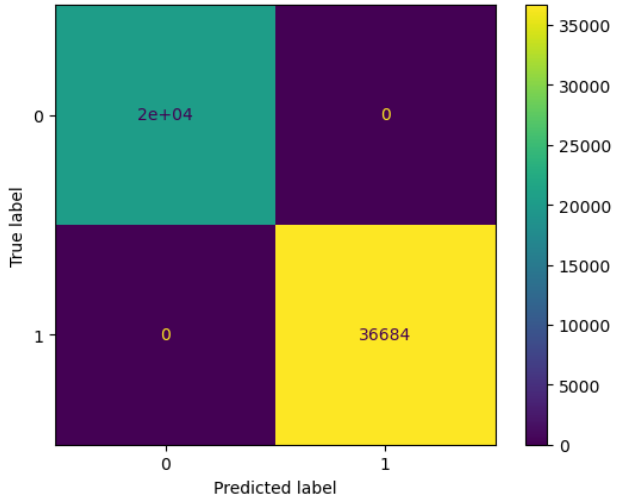

Figure 5.1: Confusion Matrix for Adaptive Boosting.

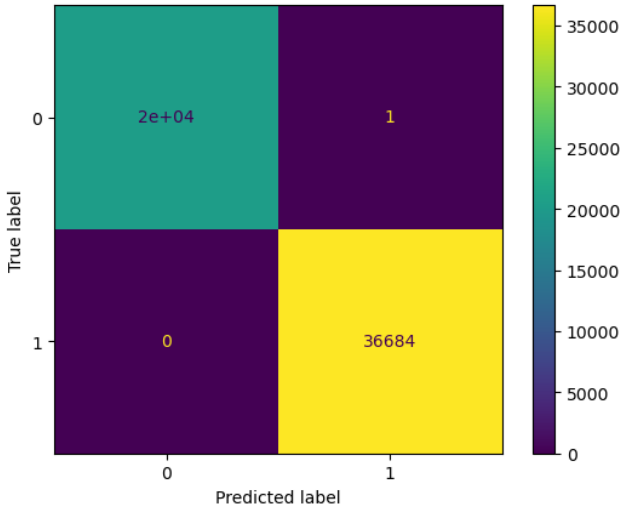

Figure 5.2:Confusion Matrix for Gradient Boosting.

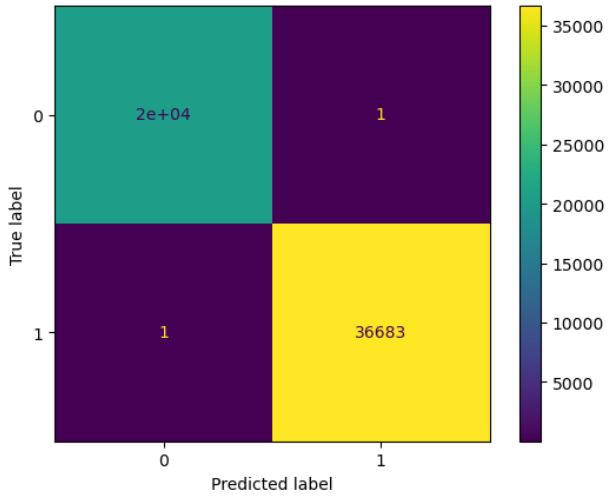

Figure 5.3: Confusion Matrix for Bagging Classifier.

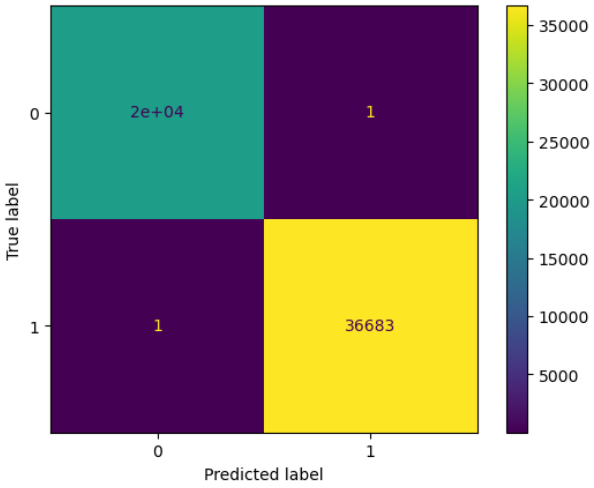

Figure 5.4: Confusion Matrix for Stacking Classifier.

5.5 Comparison

We will now proceed to compare our results with selected research works that serve as a frame of reference. To ensure a diverse range of techniques and results, we have chosen research works [20,53,54,56]. During the comparison and analysis of our work with these references, it is crucial to consider the identified research gaps. In light of this, we would like to highlight the following gaps identified in our proposed research:

1. Conventional ML techniques exhibit relatively lower performance, especially when trained with outdated and non-SDN-based or conventional network-based datasets. To address this, we propose ensemble-based techniques utilizing a state-of-the-art flow-based dataset specifically designed for SDN, known as InSDN.
2. State-of-the-art techniques like DL offer improved accuracy but come with increased computational costs. Therefore, we propose the use of ensemble-based techniques, which can yield highly accurate models while utilizing fewer computational resources.
3. Some research works have proposed ensemble-of-DL techniques, which are even more resource-intensive, as discussed in the related work section of our research.

Taking these points into consideration, let us now compare our work with research work [53]. Although research work [53] employs traditional ML techniques, the usage of representative datasets for SDN enables it to achieve over 99.99% precision, recall, and F1-Score. When

comparing our proposed work with research work [53], we observe that both achieve high performance due to the usage of representative SDN datasets. However, unlike research work [53], our work achieves 100% accuracy, precision, recall, and f1-score in the testing set for adaptive boosting ensemble technique.

Moving on, research work [54] introduces a DL-based model called DDoSNet, utilizing the CICDDoS2019 dataset with a total of 161,523 training samples and 46,150 validation samples. This research work achieves 99% precision, recall, F1-Score, and accuracy. On the other hand, research work [20] proposes a CNN-DNN-RNN-based DL ensemble technique utilizing the CIC-IDS2017 dataset with 45,149 samples, achieving an accuracy rate of 99.05%. Lastly, research work [56] proposes a boosting method for DDoS detection in SDN using the CICDDoS2019 dataset; however, the exact size of the dataset that is used for training, testing, and validation purpose is not mentioned. The performance of this research work is evaluated in a training set, evaluation set, and testing set, resulting in accuracies of 99.3%, 99.3%, and 99.4%, respectively. Therfore, with relative to research work [20,53,54,56], our work performs better.

In addition to the InSDN dataset, the performance of the proposed Four ensemble-based techniques was also evaluated using the CICIDS2017 conventional dataset that is generated by [62]. The CICIDS2017 dataset represents a traditional network, while the InSDN dataset is specific to SDN-based networks. When tested with the CICIDS2017 dataset, all four models that are trained with that same dataset achieved an accuracy of over 99.9%, indicating their effectiveness in detecting attacks in traditional networks. However, when tested with the InSDN dataset, the accuracy of the models dropped significantly. The Adaptive Boosting, Gradient Boosting, DT-based Bagging, and DT-KNN-based Stacking models achieved accuracies of 36%, 35.9%, 36%, and 35.1%, respectively. This decrease in accuracy suggests that the models may not be well-suited for the characteristics and complexities of SDN-based networks. The line graph in Figure 5.5 represents the accuracy of the ensemble models on the InSDN dataset. As Figure 5.5 shows, all four models exhibit similar accuracy levels, with slight variations. It is important to note that the low accuracy values indicate the challenges and limitations of the models in accurately detecting attacks in SDN-based networks.

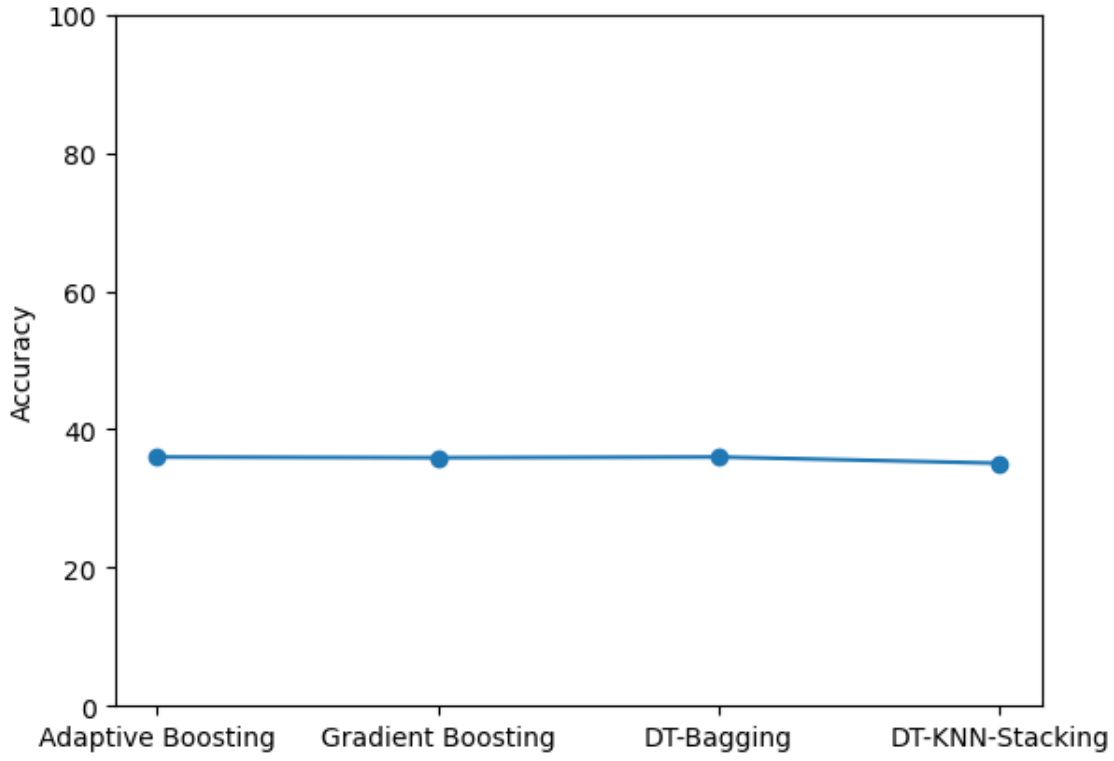

Figure 5.5: Accuracy of Models Trained with CICIDS2017 and tested with InSDN

5.6 System Prototype and Simulation

This section presents simulation tools, steps used to set up a simulation environment, usages of the tools, experimental topology, and simulation result discussion.

1. **Simulation Tools**: software requirements for the simulation are as follow.

- **Virtualbox:** it is a virtualization application that extends the capability of computers in such a way that multiple operating systems can run.
- **Ubuntu 20:** Linux-based open-source operating system.
- **Python3-pip:** python-based package manager. It is used to install and uninstall several Python packages.
- **Git:** open-source version control system. In this simulation, we used it to clone the cicflowmeter project from the respective git repository.
- **Mininet:** is a network emulator tool. it is possible to create virtual hosts, switches, controllers, and links with it.
- **Ryu SDN Controller:** SDN controller we used to create the simulated SDN environment.

- **Cicflowmeter:** open-source project for traffic flow generation. It generates bidirectional traffic flows. More than 80 statistical network traffic features are generated with these tools. We use these tools because the InSDN dataset is generated with this too as a result the DDoS detection model only knows features from this tool.
- **Tcpdump:** is used to capture and analyze network packets. It runs under a command line interface.
- **Wireshark:** is also a network analysis tool that is used to capture and filter networks for further analysis.
- **Hping3:** a tool used to generate a DDoS attack.

2. **Setting up the Simulation Environment**: the following steps are the required steps to set up Mininet and Ryu based SDN simulation environment.

- After installing the VitualBox and installing the Ubuntu operating system the Ryu controller and the Mininet SDN network simulation tool has to be installed as follow.
  - ✓ Pip3 install ryu
  - ✓ Sudo apt install mininet
- To install Sklearn use the following command
  - ✓ Pip3 install –U scikit-learn
- To install cicflowmeter use git to clone the cicflowmeter project from the following link and install as follow.
  - ✓ git clone https://gitlab.com/hieulw/cicflowmeter
  - ✓ cd cicflowmeter
  - ✓ sudo python3 setup.py install

3. **Usages of the tools:** A description of the use of the tools is provide as follow.
   - ✓ usages of cicflowmeter: The tcpdump tool captures and save the file as a pcap file format. As a result, we need to converet the pcap file to csv file by using the command below.
     - cicflowmeter –f example.pcap –c flow.csv
   - ✓ usage of Mininet: the following options are used under the Mininet command to create the SDN network topology.
     - -- controller: types of controller, local/remote and remote controller ip
     - -- mac: mac address start with 00:00:00:00:00:00
     - – i: ip subnet for the topology

- -- switch: switch type and OpenFlow version
- -- topo: topology type and parametrs

✓ Usage of Ryu controller: by the following command a simple switch application can be started. The simple switch application is the build in sample application by Ryu maintainer.

- ryu-manager ryu.app.simple_switch_13

✓ Usage of Hping3: the following command is used to generate volumetric based DDoS attack on the simulated environment.

- hping3 --flood --rand-source <target_ip>

4. **Experment Topology:** The topology of the Experiment SDN network is described and illustrated as follow. The topology is composed of Four hosts labeled from H1 to H4. These hosts are required to initiate attack and normal traffic in the SDN topology. A switch S1 and a controller C0 is also used as it is shown in the Figure 5.6.

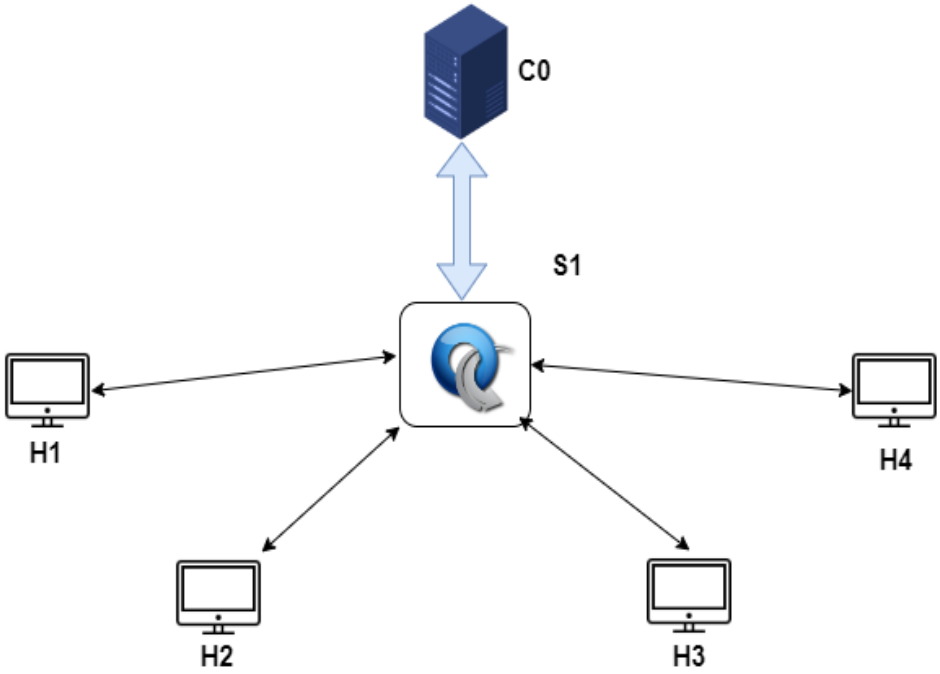

Figure 5.6: SDN Network Topology for Simulation

5. **Simulation:** The following steps can be applied to start the simulation on the proposed Mininet and Ryu controller-based SDN network.

- Create instance of simple_switch_13 ryu controller application as it is illustrated in Figure 5.7.

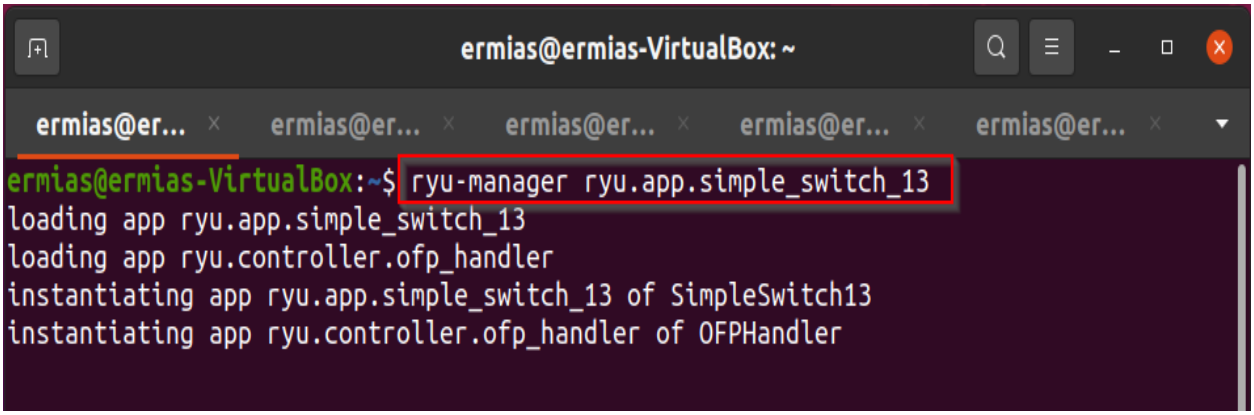

Figure 5.7: Ryu Application Initialization

- Create the SDN network in Mininet as it is illustrated in Figure 5.8.

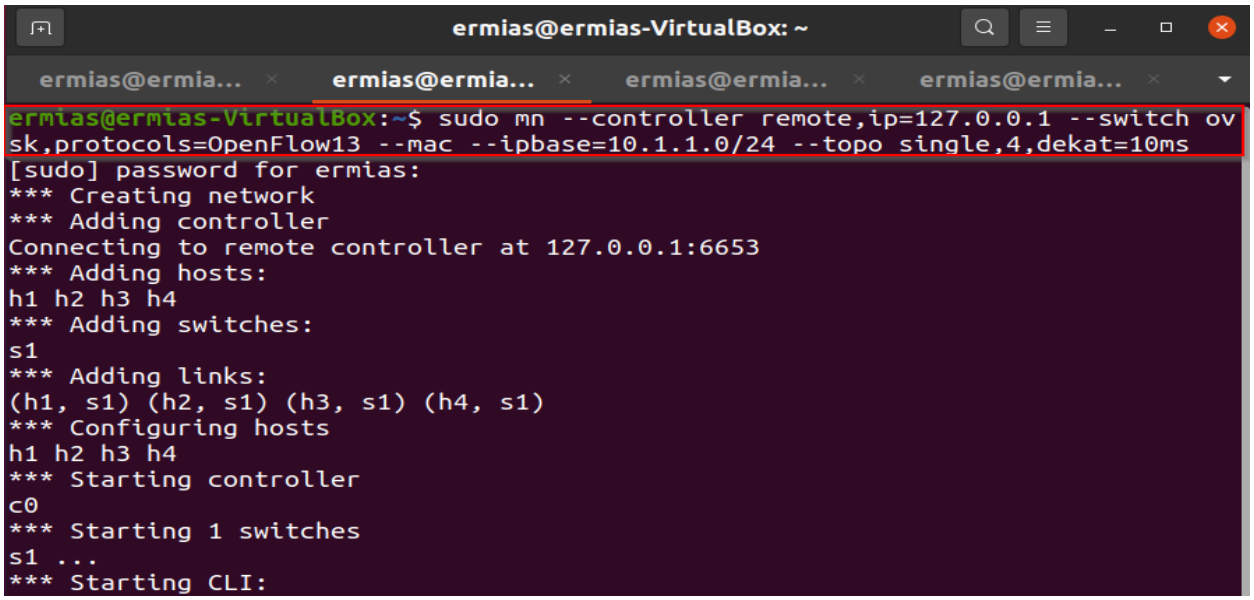

Figure 5.8: Mininet Application Initialization

- Figure 5.9 illustrates a first look on the OpenFlow Switch flow table.

✓ **Cookies**: This an opaque data value that is chosen by the controller to filter flow.

✓ **Priority:** Precedenc of the flow entry.

✓ **Match fields:** To match againes packets. These consists of the ingress port, frame and packet header fields, and optionally metadata specified by a previous table.

✓ **Action:** To modiy the action set or pipeline processing. Typically the output port associated with the flow.

✓ **Counters:** To update for maching packets.

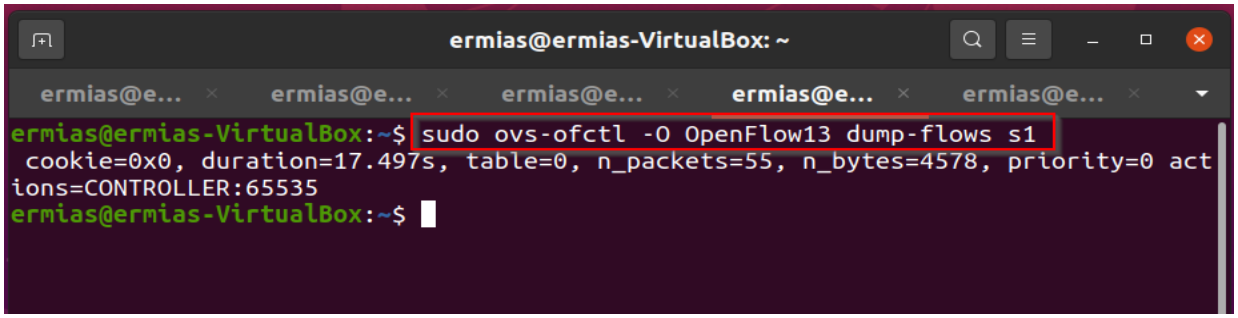

Figure 5.9: OpenFlow Switch flow table

- Generate normal and attack traffic with “ping” command and “hping3” tool, and capture the traffic with tcpdump to further process it with the cicflowmeter and wireshark tool. Here “H1” generates attack to “h4” traffic as it is shown in Figure 5.10.

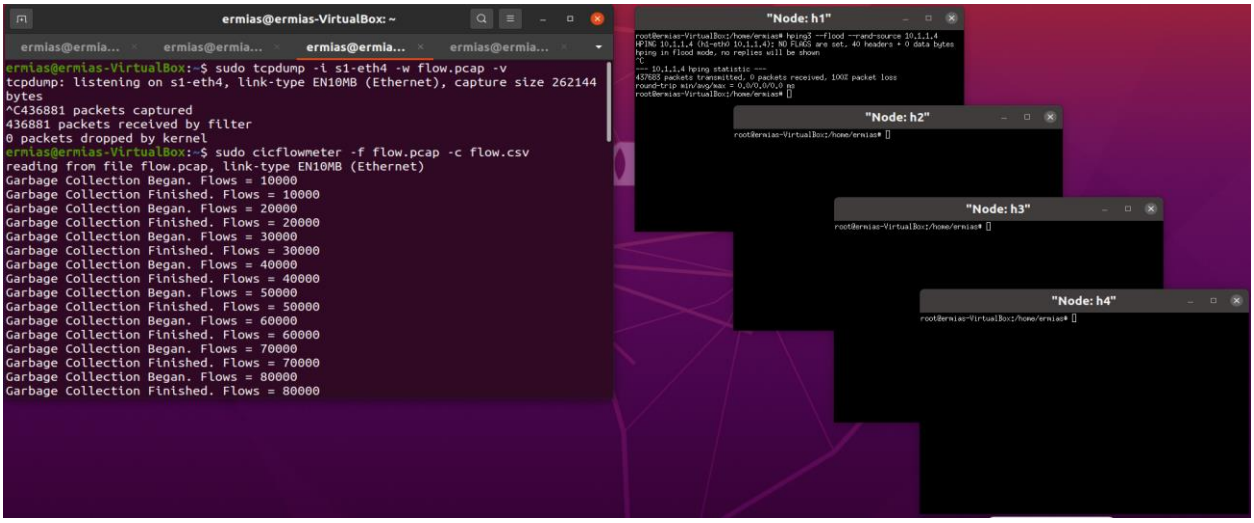

Figure 5.10: Performing DDoS attack and capture the traffic with tcpdump

- The python code at Annex G is used to detect the availability of DDoS in the SDN network using the DT-Bagging ensemble-based ML model. Using the above simulation after generating a flooding attack using “H1” the output of the source code correctly detects the availiability of DDoS as it is shown in Figure 5.11.

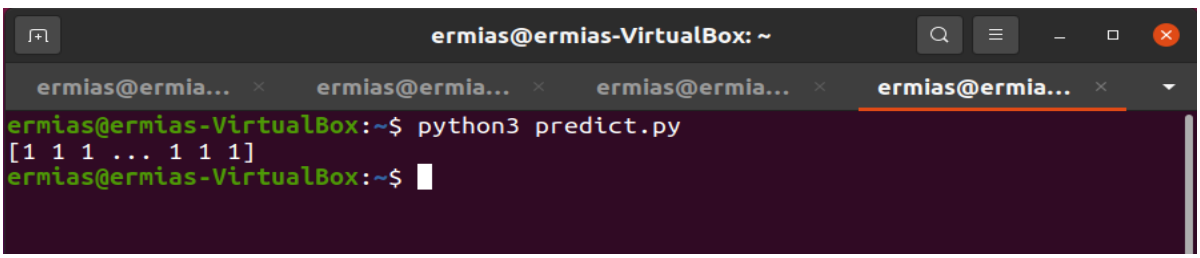

Figure 5.11: Simulation Results of the DDoS Attack

CHAPTER SIX: CONCLUSION AND FUTURE WORKS

6.1 Conclusion

The swift development of the Internet technologies and businesses that depends on it becomes more complicated, agile, and variable. Because most of the functionalities in traditional networking are implemented and configured with a huge involvement of human interaction, it becomes more difficult for this networking paradigm to handle this requirement of the modern era. In this perspective, SDN the new networking paradigm facilitates network management and provides programming capability. As a result, it supports today's complex and sophisticated networking infrastructures.

In spite of its advantages, the SDN networking paradigm is vulnerable to multiple attacks; mostly due to the inherent nature of the network architecture. Malicious applications, controllers' vulnerability, legitimacy, consistency of the flow rules, and vulnerable interface are some of the attack surfaces of SDN. Though there are many possible attacks on SDN, DDoS is the most common and devastating one. As a result, several studies applied several techniques to overcome these challenges. One of the approaches is using ML techniques and state-of-the-art techniques like DL.

Traditional ML techniques perform poorly to detect DDoS on SDN when compared with state-of-the-art techniques like DL. On the contrary, these state-of-the-art techniques are computationally complex and computationally inefficient. Therefore, the crucial objective of this research work is to find an optimal solution for this particular problem. In this perspective, the research work proposed ensemble-based ML techniques and utilized a state-of-the-art flow-based dataset. In addition, the boosting, bagging, and stacking ensemble techniques are developed and compared to detect DDoS on SDN.

The overall process of the proposed solution starts with preparing the InSDN dataset and cleaning it for testing purpose. Then, we create and train the ensemble models. These models are evaluated with several hyperparameters to get the best model performance. The result of each ensemble technique is compared to each other. In this regard, the adabtive boosting ensemble learining technique has the highest accuracy of 100%, while the gradient boosting has the lowest prediction time. Compared with traditional ML techniques the proposed technique performs extremely well.

Compared with state-of-the-art techniques like DL, the proposed techniques perform better without consuming high computational resources. From this, it is possible to conclude that the proposed solution by this particular research work has great accuracy and efficiency to detect DDoS in the SDN environment.

6.2 Contribution of the Study

The main contribution of the research work is measured with respect to the research gaps that are identified. More precisely, finding out what efficiency and accuracy the ensemble-based ML techniques specifically boosting, bagging, and stacking technique provides to detect DDoS on SDN by utilizing state-of-the-art flow-based features is the major research objective. Putting this context into consideration the following points are the major contribution of the research work.

- Using ensemble-based techniques to detect DDoS in SDN is further investigated and experimented with. The obtained result from the boosting, bagging, and stacking ensemble-based techniques is compared with each other and with results from other studies that are designed by considering both traditional ML techniques and state-of-the-art DL techniques.
- Flow-based features from a state-of-the-art dataset are utilized and the performance of the proposed ensemble-based model is enhanced. In this regard, it has been discovered that using ensemble-based techniques and flow-based features to detect DDoS on SDN performs better compared with DL thecniques that are proposed by other studies used as a reference in this research work comparison section.
- A trained ensemble-based DDoS detection model for SDN is used in a simulated environment to further experiment the performance of the proposed model in a simulated SDN environment.

6.3 Future Work

There are several research works that can be conducted as a future work

- The proposed research work has not considered slow-rate attacks in its scope. As a result, the effectiveness of the proposed solution can be tested in this particular perspective in future research works.
- The effectiveness and efficiency of applying a feature selection mechanism in the flow-based features to train a DDoS detection method can be considered as another future work perspective.

- Finally, since this particular research work covers only a DDoS detection mechanism, future research work can consider utilizing the proposed techniques for Intrusion Detection mechanisms for SDN.

REFERENCES

[1]. Z. AL-Badrany and Al-Somaidai, "Network Performance Evaluation of Distributed SDN using Floodlight Controllers," 2019 2nd International Conference on Electrical, Communication, Computer, Power and Control Engineering (ICECCPCE), 2019, pp. 34-39.

[2]. P. Zhai, Y. Song, X. Zhu, L. Cao, J. Zhang and C. Yang, "Distributed Denial of Service Defense in Software Defined Network Using OpenFlow," 2020 IEEE/CIC International Conference on Communications in China (ICCC), 2020, pp. 1274-1279.

[3]. A. T. Kyaw, M. Zin Oo, and C. S. Khin, "Machine-Learning Based DDOS Attack Classifier in Software Defined Network," 2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2020, pp. 431-434

[4]. R. D. Devapriya and S. I. Gandhi, "Enhanced Load Balancing and QoS Provisioning Algorithm for a Software Defined Network," 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), 2020, pp. 1-5.

[5]. M. H. H. Khairi et al., "Detection and Classification of Conflict Flows in SDN Using ML Algorithms," in IEEE Access, vol. 9, pp. 76024-76037, 2021.

[6]. K. Bavani, M. P. Ramkumar and E. Selvan G.S.R., "Statistical Approach Based Detection of Distributed Denial of Service Attack in a Software Defined Network," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020, pp. 380-385.

[7]. M. J. Anagha, R. Lepakshi, V. Goutham, V. Thavish and T. G. Keerthan Kumar, "Packet Injection and Dos Attack Controller Software(PDACS) Module to Handle Attacks in Software Defined Network," 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), 2020, pp. 966-970.

[8]. A. Mabayoje & Balogun, Abdullateef & Bajeh, Amos & Musa, Badamasi, “SOFTWARE DEFECT PREDICTION: EFFECT OF FEATURE SELECTION AND ENSEMBLE METHODS,” 2018, vol. 3, pp. 518-522.

[9]. K. Wang, X. Liu, J. Zhao, H. Gao and Z. Zhang, "Application Research of Ensemble Learning Frameworks," 2020 Chinese Automation Congress (CAC), 2020, pp. 5767-5772.

[10]. J. Bhayo, S. Hameed and S. A. Shah, "An Efficient Counter-Based DDoS Attack Detection Framework Leveraging Software Defined IoT (SD-IoT)," in IEEE Access, vol. 8, pp. 221612-221631, 2020.

[11]. S. Dong, K. Abbas and R. Jain, "A Survey on Distributed Denial of Service (DDoS) Attacks in SDN and Cloud Computing Environments," in IEEE Access, vol. 7, pp. 80813-80828, 2019.

[12]. R. Amin, E. Rojas, A. Aqdus, S. Ramzan, D. Casillas-Perez and J. M. Arco, "A Survey on ML Techniques for Routing Optimization in SDN," in IEEE Access, vol. 9, pp. 104582-104611, 2021.

[13]. K. S. Sahoo et al., "An Evolutionary SVM Model for DDOS Attack Detection in Software Defined Networks," in IEEE Access, vol. 8, pp. 132502-132513, 2020.

[14]. A. Ahalawat, S. S. Dash, A. Panda and K. S. Babu, "Entropy Based DDoS Detection and Mitigation in OpenFlow Enabled SDN," 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), Vellore, India, 2019, pp. 1-5.

[15]. P. T. Duy, D. T. T. Hien and V. -H. Pham, "A role-based statistical mechanism for DDoS attack detection in SDN," 2018 5th NAFOSTED Conference on Information and Computer Science (NICS), Ho Chi Minh City, Vietnam, 2018, pp. 177-182.

[16]. K. S. Sahoo, A. Iqbal, P. Maiti and B. Sahoo, "A Machine Learning Approach for Predicting DDoS Traffic in Software Defined Networks," 2018 International Conference on Information Technology (ICIT), Bhubaneswar, India, 2018, pp. 199-203.

[17]. M. W. Nadeem, H. G. Goh, Y. Aun and V. Ponnusamy, "Detecting and Mitigating Botnet Attacks in Software-Defined Networks Using Deep Learning Techniques," in IEEE Access, vol. 11, pp. 49153-49171, 2023.

[18]. S. Haider et al., "A Deep CNN Ensemble Framework for Efficient DDoS Attack Detection in Software Defined Networks," in IEEE Access, vol. 8, pp. 53972-53983, 2020.

[19]. Sadi Gadri, “Developing an Efficeint Predictive Model Based on ML and DL Approaches to Detect Diabetes.” Informatica, An International Journal of Computing and Informatics, Vol 45, No 3, pp. 433-440, 2021.

[20]. U. Mbasuva and G. -A. L. Zodi, "Designing Ensemble Deep Learning Intrusion Detection System for DDoS attacks in Software Defined Networks," 2022 16th International Conference on Ubiquitous Information Management and Communication (IMCOM), 2022, pp. 1-8.

[21]. T. -K. Luong, T. -D. Tran and G. -T. Le, "DDoS attack detection and defense in SDN based on ML," 2020 7th NAFOSTED Conference on Information and Computer Science (NICS), 2020, pp. 31-35.

[22]. A. Mubarakali and A. S. Alqahtani, "A Survey: Security Threats and Countermeasures in Software Defined Networking," 2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT), 2019, pp. 180-185.

[23]. Y. Liu, B. Zhao, P. Zhao, P. Fan and H. Liu, "A survey: Typical security issues of software-defined networking," in China Communications, vol. 16, no. 7, pp. 13-31, July 2019.

[24]. Y. He and X. Zhang, "A Survey on Network Measurement for Software-Defined Networks," 2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE), 2019, pp. 1534-1540.

[25]. P. Zhai, Y. Song, X. Zhu, L. Cao, J. Zhang and C. Yang, "Distributed Denial of Service Defense in Software Defined Network Using OpenFlow," 2020 IEEE/CIC International Conference on Communications in China (ICCC), 2020, pp. 1274-1279.

[26]. Y. Zhao, Y. Li, X. Zhang, G. Geng, W. Zhang and Y. Sun, "A Survey of Networking Applications Applying the Software Defined Networking Concept Based on Machine Learning," in IEEE Access, vol. 7, pp. 95397-95417, 2019.

[27]. N. Faujdar, A. Sinha, H. Sharma and E. Verma, "Network Security in Software defined Networks (SDN)," 2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE), 2020, pp. 377-380.

[28]. J. Xie et al., "A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): Research Issues and Challenges," in IEEE Communications Surveys & Tutorials, vol. 21, no. 1, pp. 393-430, Firstquarter 2019.

[29]. M. Alsaeedi, M. M. Mohamad and A. A. Al-Roubaiey, "Toward Adaptive and Scalable OpenFlow-SDN Flow Control: A Survey," in IEEE Access, vol. 7, pp. 107346-107379, 2019.

[30]. M. Parashar, A. Poonia and K. Satish, "A Survey of Attacks and their Mitigations in Software Defined Networks," 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2019, pp. 1-8.

[31]. Y. Al Mtawa, A. Haque and H. Lutfiyya, "Migrating From Legacy to Software Defined Networks: A Network Reliability Perspective," in IEEE Transactions on Reliability, vol. 70, no. 4, pp. 1525-1541, Dec. 2021.

[32]. Lubna Fayez Eliyan, Roberto Di Pietro, “DoS and DDoS attacks in Software Defined Networks: A survey of existing solutions and research challenges,” Future Generation Computer Systems, vol 122, 2021, pp. 149-171.

[33]. S. Singh and S. Prakash, "A Survey on Software Defined Network based on Architecture, Issues and Challenges," 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), 2019, pp. 568-573.

[34]. Jiesheng Zheng, Guangcai Wu, Bojian Wen, Yaosong Lu, and Ruigang Liang, “Research on SDN-Based Mimic Server Defense Technology”, 2019 International Conference on Artificial Intelligence and Computer Science (AICS), 2019 Association for Computing Machinery, New York, NY, USA, pp. 163–169.

[35]. B. Sokappadu, A. Hardin, A. Mungur and S. Armoogum, "Software Defined Networks: Issues and Challenges," 2019 Conference on Next Generation Computing Applications (NextComp), 2019, pp. 1-5.

[36]. S. -H. Tseng, A. Tang, G. L. Choudhury and S. Tse, "Routing Stability in Hybrid Software-Defined Networks," in IEEE/ACM Transactions on Networking, vol. 27, no. 2, pp. 790-804, April 2019.

[37]. R. D. Devapriya and S. I. Gandhi, "Enhanced Load Balancing and QoS Provisioning Algorithm for a Software Defined Network," 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), 2020, pp. 1-5.

[38]. Farah Chahlaoui, Mohammed Raiss El-Fenni, and Hamza Dahmouni, “Performance Analysis of Load Balancing Mechanisms in SDN Networks”, 2019 2nd International Conference on Networking, Information Systems & Security (NISS19), Association for Computing Machinery, New York, NY, USA, 2019, pp. 36, 1–8.

[39]. Open Networking Foundation, “OpenFlow Switch Specification: Version 1.5.1”, 2015, retrieved from https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf , Last accessed on Jan 03, 2023.

[40]. J. Liu and Q. Xu, "Machine Learning in Software Defined Network," 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), 2019, pp. 1114-1120.

[41]. I. D. Mienye and Y. Sun, "A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects," in IEEE Access, vol. 10, pp. 99129-99149, 2022.

[42]. Xibin DONG, Zhiwen YU, Wenming CAO, Yifan SHI, Qianli MA “A survey on ensemble learning”, Frontiers of Computer Science, Vol. 14, PP. 241-258, 2020.

[43]. Yang, Yongquan & Lv, Haijun & Chen, Ning, “A Survey on Ensemble Learning under the Era of Deep Learning”. Artificial Intelligence Review, 2022.

[44]. Fortmann-Roe S. “Understanding the Bias-Variance Tradeoff”. Retrived from http://scott.fortmann-roe.com/docs/BiasVariance.html. Date last accessed 18 Jun 2023.

[45]. T. N. Rincy and R. Gupta, "Ensemble Learning Techniques and its Efficiency in Machine Learning: A Survey," 2nd International Conference on Data, Engineering and Applications (IDEA), Bhopal, India, 2020, pp. 1-6.

[46]. S. K. Yadav, P. Suguna and R. L. Velusamy, "Entropy based mitigation of Distributed-Denial-of-Service (DDoS) attack on Control Plane in Software-Defined-Network (SDN)," 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, 2019, pp. 1-7.

[47]. T. Lotlikar and D. Shah, "A Defense Mechanism for DoS Attacks in SDN (Software Defined Network)," 2019 International Conference on Nascent Technologies in Engineering (ICNTE), Navi Mumbai, India, 2019, pp. 1-7.

[48]. A. Ahmad, E. Harjula, M. Ylianttila and I. Ahmad, "Evaluation of Machine Learning Techniques for Security in SDN," 2020 IEEE Globecom Workshops (GC Wkshps, Taipei, Taiwan, 2020, pp. 1-6.

[49]. J. Cui, J. Zhang, J. He, H. Zhong and Y. Lu, "DDoS detection and defense mechanism for SDN controllers with K-Means," 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC), Leicester, UK, 2020, pp. 394-401.

[50]. Polat, Huseyin, Onur Polat, and Aydin Cetin, "Detecting DDoS Attacks in Software-Defined Networks Through Feature Selection Methods and Machine Learning Models," *Sustainability*, Vol. 12(3), PP. 1-16, 2020.

[51]. N. Ahuja, G. Singal and D. Mukhopadhyay, "DLSDN: Deep Learning for DDOS attack detection in Software Defined Networking," 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2021, pp. 683-688.

[52]. L. Wang and Y. Liu, "A DDoS Attack Detection Method Based on Information Entropy and Deep Learning in SDN," 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chongqing, China, 2020, pp. 1084-1088.

[53]. M. S. Elsayed, N. -A. Le-Khac and A. D. Jurcut, "InSDN: A Novel SDN Intrusion Dataset," in IEEE Access, vol. 8, pp. 165263-165284, 2020.

[54]. M. S. Elsayed, N. -A. Le-Khac, S. Dev and A. D. Jurcut, "DDoSNet: A Deep-Learning Model for Detecting Network Attacks," 2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), Cork, Ireland, 2020, pp. 391-396.

[55]. V. Deepa, K. M. Sudar and P. Deepalakshmi, "Design of Ensemble Learning Methods for DDoS Detection in SDN Environment," 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), Vellore, India, 2019, pp. 1-6.

[56]. A. Shirmarz, A. Ghaffari, R. Mohammadi and S. Akleylek, "DDOS Attack Detection Accuracy Improvement in Software Defined Network (SDN) Using Ensemble Classification," 2021 International Conference on Information Security and Cryptology (ISCTURKEY), Ankara, Turkey, 2021, pp. 111-115.

[57]. D. Firdaus, R. Munadi and Y. Purwanto, "DDoS Attack Detection in Software Defined Network using Ensemble K-means++ and Random Forest," 2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), Yogyakarta, Indonesia, 2020, pp. 164-169.

[58]. V. Gulati and N. Raheja, "Efficiency Enhancement of Machine Learning Approaches through the Impact of Preprocessing Techniques," 2021 6th International Conference on Signal Processing, Computing and Control (ISPCC), Solan, India, 2021, pp. 191-196.

[59]. Qing Lyu and Xingjian Lu, "Effective Media Traffic Classification Using Deep Learning", 2019 3rd International Conference on Compute and Data Analysis (ICCDA) Association for Computing Machinery, New York, NY, USA, 2019, pp. 139–146.

[60]. Y. Xin et al., "Machine Learning and Deep Learning Methods for Cybersecurity," in IEEE Access, vol. 6, pp. 35365-35381, 2018.

[61]. Tigist Hidru, "Modeling Intrusion Detection System Based on Anomaly Approach Using Machine Learing Techniques", Unpublished Master Thesis, Department of Computer Science, Addis Ababa University, 2020.

[62]. Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018.

ANNEXE A: Dataset preparation source code

```
#importing important libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

#Normal and attack traffics are imported for dataset preparation processing.

normal = pd.read_csv('./../InSDN_DatasetCSV/Normal_data.csv')

metasploitable_attack = pd.read_csv('./../InSDN_DatasetCSV/metasploitable-2.csv')

ovs_attack = pd.read_csv('./../InSDN_DatasetCSV/OVS.csv')

#sample data from the normal group

normal.head()

#sample data from the metasploitable attack group

metasploitable_attack.head()

#sample data from the OVS attack group

ovs_attack.head()

#Selecting the DDoS attack from the metasploitable attack dataset.

metasploitable_ddos_attack = metasploitable_attack[metasploitable_attack['Label'] == 'DDoS']

metasploitable_ddos_attack.shape

#sample metasploitable DDoS attack information

metasploitable_ddos_attack.head()

#Reading the labels. As it is indicated in the output DDoS has a space at the end

ovs_attack['Label'].value_counts().index

#Removing the space for every entry of the OVS attack dataset

ovs_attack["Label"] = ovs_attack["Label"].map(str.strip)
```

#Now as it is demonstrated below unlike the above output now DDoS has no extra white space at the end.

ovs_attack['Label'].value_counts().index

#Selecting the DDoS attack from the general dataset for the ovs_attack dataset.

ovs_ddos_attack = ovs_attack[ovs_attack['Label'] == 'DDoS']

ovs_ddos_attack.shape

#sample OVS DDoS attack information

ovs_ddos_attack.head()

#combining the two DDoS attack datasets

ddos_attack = [metasploitable_ddos_attack,ovs_ddos_attack]

attackFrame = pd.concat(ddos_attack)

attackFrame.shape

#sample DDoS attack information from the combined dataset

attackFrame.head()

create a frame to hold the attack and normal dataset

frames = [normal, attackFrame]

prepared_dataset = pd.concat(frames)

prepared_dataset.shape

prepared_dataset['Label'].value_counts()

#drawing a bar graph distribution

prepared_dataset.head()

#Export the final prepared dataset

prepared_dataset.to_csv('./../prepared_dataset/prepared_dataset.csv', index=False)

ANNEXE B: Selected features

No	Features	No	Features
1	Flow Duration	25	Bwd IAT Tot
2	Tot Fwd Pkts	26	Bwd IAT Mean
3	Tot Bwd Pkts	27	Bwd IAT Std
4	TotLen Fwd Pkts	28	Bwd IAT Max
5	TotLen Bwd Pkts	29	Bwd IAT Min
6	Fwd Pkt Len Max	30	Fwd Header Len
7	Fwd Pkt Len Min	31	Bwd Header Len
8	Fwd Pkt Len Mean	32	Fwd Pkts/s
9	Fwd Pkt Len Std	33	Bwd Pkts/s
10	Bwd Pkt Len Max	34	Pkt Len Min
11	Bwd Pkt Len Min	35	Pkt Len Max
12	Bwd Pkt Len Mean	36	Pkt Len Mean
13	Bwd Pkt Len Std	37	Pkt Len Std
14	Flow Byts/s	38	Pkt Len Var
15	Flow Pkts/s	39	Pkt Size Avg
16	Flow IAT Mean	40	Active Mean
17	Flow IAT Std	41	Active Std
18	Flow IAT Max	42	Active Max
19	Flow IAT Min	43	Active Min
20	Fwd IAT Tot	44	Idle Mean
21	Fwd IAT Mean	45	Idle Std
22	Fwd IAT Std	46	Idle Max
23	Fwd IAT Max	47	Idle Min
24	Fwd IAT Min		

ANNEXE C: Source Code to Clean the Dataset

```
#import important libraries and packages

import pandas as pd

from sklearn import preprocessing

#Reading the dataset

df = pd.read_csv('./../prepared_dataset/prepared_dataset.csv')

#cheack if missing value is available

df.isnull().values.sum()

#Dropping irrelevant attribute

df.drop(['Flow ID', 'Src IP', 'Src Port', 'Dst IP', 'Dst Port', 'Protocol', 'Timestamp','Fwd PSH Flags', 'Bwd PSH Flags','Fwd URG Flags', 'Bwd URG Flags','FIN Flag Cnt','SYN Flag Cnt','RST Flag Cnt','PSH Flag Cnt','ACK Flag Cnt','URG Flag Cnt','CWE Flag Count','ECE Flag Cnt','Down/Up Ratio','Fwd Seg Size Avg','Bwd Seg Size Avg','Fwd Byts/b Avg','Fwd Pkts/b Avg','Fwd Blk Rate Avg','Bwd Byts/b Avg','Bwd Pkts/b Avg','Bwd Blk Rate Avg','Subflow Fwd Pkts','Subflow Fwd Byts','Subflow Bwd Pkts','Subflow Bwd Byts','Init Fwd Win Byts','Init Bwd Win Byts','Fwd Act Data Pkts','Fwd Seg Size Min'], axis=1, inplace=True)

#one-hot encoding

status_enum = {'Normal': 0, 'DDoS': 1}

df['Label'] = df['Label'].map(status_enum)

#write to a file

df.to_csv('./../clean_dataset/clean_dataset.csv', index=False)
```

ANNEXE D: Source Code for Train-Test split

```
#import important libraries and packages

import pandas as pd

from sklearn.model_selection import train_test_split

#import clean dataset

df = pd.read_csv('./../clean_dataset/clean_dataset.csv')

#split features and labels

features = df.drop('Label', axis=1) #all features in the dataset

labels = df['Label'] #Labels in the dataset

#split into a train, validation, and test set

X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.3, random_state =
42)

X_train.to_csv('./../train_dataset/train_features.csv', index=False)

X_test.to_csv('./../test_dataset/test_features.csv', index=False)


y_train.to_csv('./../train_dataset/train_labels.csv', index=False)

y_test.to_csv('./../test_dataset/test_labels.csv', index=False)
```

ANNEXE E: Sample Source Code to Train KNN_DT Based Stacking Ensemble Model

```
import joblib

import pandas as pd

from sklearn.ensemble import StackingClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import GridSearchCV

#import training features and labels

tr_features = pd.read_csv('./../../train_dataset/train_features.csv')

tr_labels = pd.read_csv('./../../train_dataset/train_labels.csv')

# Hyperparameter tuning

def print_results(results):

    print('BEST PARAMS: {}\n'.format(results.best_params_))

    means = results.cv_results_['mean_test_score']

    stds = results.cv_results_['std_test_score']

    for mean, std, params in zip(means, stds, results.cv_results_['params']):

        print('{} (+/-{}) for {}'.format(round(mean, 3), round(std * 2, 3), params))

base_learners = [('knn', KNeighborsClassifier()),('dt', DecisionTreeClassifier())]

# Initialize Stacking Classifier with the Meta Learner

clf = StackingClassifier(estimators=base_learners)

clf.get_params()

parameters = {'final_estimator': [LogisticRegression()],'knn__n_neighbors':
[0,3],'dt__random_state': [5,7]}
```

```
cv = GridSearchCV(clf,parameters,cv=5)

cv.fit(tr_features.values,tr_labels.values.ravel())

print_results(cv)

# return the best estimator

cv.best_estimator_

# Save the best estimator

joblib.dump(cv.best_estimator_, './../../models/Knn_dt_stacking.pkl')
```

ANNEXE F: Model Evaluation and Testing Source Code

```
#import important packages

import joblib

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.metrics import accuracy_score, precision_score,
recall_score,f1_score,confusion_matrix,ConfusionMatrixDisplay

from time import time


#reade testing dataset

te_features = pd.read_csv('./../../test_dataset/test_features.csv')

te_labels = pd.read_csv('./../../test_dataset/test_labels.csv')


#import models for testing

ab_mdl = joblib.load('./../../models/Addaptive_Boosting.pkl')

gb_mdl = joblib.load('./../../models/Gradient_Boosting.pkl')

bagging_mdl = joblib.load('./../../models/DT_Bagging.pkl')

stacking_mdl = joblib.load('./../../models/Knn_dt_stacking.pkl')


#model evaluation function

def evaluate_model(model, features, labels):

    start = time()

    pred = model.predict(features)

    end = time()

    accuracy = round(accuracy_score(labels, pred), 3)
```

```
    precision = round(precision_score(labels, pred), 3)
    recall = round(recall_score(labels, pred), 3)
    f1score = round(f1_score(labels, pred, average='macro'),3)
    confmatrix = confusion_matrix(labels, pred,labels=model.classes_)
    print('{} -- Accuracy: {} / Precision: {} / Recall: {} /F1-Score: {} / Latency:
{}ms'.format(str(model).split('(')[0],
                                                    accuracy,
                                                    precision,
                                                    recall,
                                                    f1score,
                                                    round((end - start)*1000, 1)))
    print('Confustion matrix: {}', confmatrix)
    disp = ConfusionMatrixDisplay(confusion_matrix=confmatrix,display_labels=model.classes_)
    disp.plot()
    plt.show()
#Evaluation models in the testing set
for mdl in [ab_mdl, gb_mdl, bagging_mdl,stacking_mdl]:
    evaluate_model(mdl, te_features, te_labels)
```

ANNEXE G: A Python Code to Monitor the SDN Network Using the ML Model in a Simulated Environment.

```
#import libraries

import joblib

import pandas as pd

#import traffic file

flow = pd.read_csv('./flow.csv')

#import model

model = joblib.load('./DT_Bagging.pkl')

#drop irrelevant features for normal sample

flow = flow.drop(['src_ip','src_mac','dst_mac','src_port', 'dst_ip', 'dst_port', 'protocol', 'timestamp','fwd_psh_flags', 'bwd_psh_flags','fwd_urg_flags', 'bwd_urg_flags','fin_flag_cnt','syn_flag_cnt','rst_flag_cnt','psh_flag_cnt','ack_flag_cnt','urg_flag_cnt','cwe_flag_count','ece_flag_cnt','down_up_ratio','fwd_seg_size_avg','bwd_seg_size_avg','fwd_byts_b_avg','fwd_pkts_b_avg','fwd_blk_rate_avg','bwd_byts_b_avg','bwd_pkts_b_avg','bwd_blk_rate_avg','subflow_fwd_pkts','subflow_fwd_byts','subflow_bwd_pkts','subflow_bwd_byts','init_fwd_win_byts','init_bwd_win_byts','fwd_act_data_pkts','fwd_seg_size_min'],axis=1)

#reorder the column values

flow=flow[['flow_duration','tot_fwd_pkts','tot_bwd_pkts','totlen_fwd_pkts','totlen_bwd_pkts','fwd_pkt_len_max','fwd_pkt_len_min','fwd_pkt_len_mean','fwd_pkt_len_std','bwd_pkt_len_max','bwd_pkt_len_min','bwd_pkt_len_mean','bwd_pkt_len_std','flow_byts_s','flow_pkts_s','flow_iat_mean','flow_iat_std','flow_iat_max','flow_iat_min','fwd_iat_tot','fwd_iat_mean','fwd_iat_std','fwd_iat_max','fwd_iat_min','bwd_iat_tot','bwd_iat_mean','bwd_iat_std','bwd_iat_max','bwd_iat_min','fwd_header_len','bwd_header_len','fwd_pkts_s','bwd_pkts_s','pkt_len_min','pkt_len_max','pkt_len_mean','pkt_len_std','pkt_len_var','pkt_size_avg','active_mean','active_std','active_max','active_min','idle_mean','idle_std','idle_max','idle_min']]

print(model.predict(flow.values))
```

Declaration

I, the undersigned, declare that this research is my original work and has not been presented for degree in any other university, and that all sources of materials used for the research have been acknowledged.

Declared by:

Name: Ermias Tsegu Gizaw

Signature: _____

Date: _____

Confirmed byadvisor:

Name: Ayalew Belay (PhD)

Signature: _____

Date: _____