AI-Driven Security in Software-Defined Networks: A Unified Framework for Intrusion Detection and Mitigation

by

Walid M. El Gadal

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

We acknowledge and respect the Lək̓ʷəŋən (Songhees and Xʷsepsəm/Esquimalt) Peoples on whose territory the university stands, and the Lək̓ʷəŋən and W̱SÁNEĆ Peoples whose historical relationships with the land continue to this day.

AI-Driven Security in Software-Defined Networks: A Unified Framework for Intrusion Detection and Mitigation

by

Walid M. El Gadal

Supervisory Committee

Dr. Sudhakar Ganti, Supervisor
(Department of Computer Science)

Dr. Hausi Müller, Departmental Member

(Department of Computer Science)

Dr. Issa Traoré, Outside Member

(Department of Electrical and Computer Engineering)

ABSTRACT

Over the past decade, data networks have evolved from static resource deployment to a more dynamic and adaptive paradigm. Software-Defined Networking (SDN) is one of the most creative network technologies where network control is separated from forwarding. It is directly programmable and has been proposed as a way to programmatically control networks, facilitating the deployment of new applications and services, as well as tuning network policies and performance. However, various challenges have hindered achieving strong cybersecurity within the dynamic network configurations of Software-Defined Networking. Traditional cybersecurity measures, especially in programmable and dynamic network infrastructures like SDNs, are not sufficient to mitigate cyber threats. This dissertation explores the capabilities of SDN and examines how AI-driven methods can enhance intrusion detection and mitigation. The study begins by providing a comprehensive introduction to SDN, outlining its fundamental capabilities and comparative advantages over traditional network architectures. In addition, it explores SDN vulnerabilities and addresses complex security challenges.

The objective of this thesis work is to improve the detection and mitigation of threats in SDN environments. For this, we first present a dynamic defense framework that includes Machine Learning and Deep Learning techniques for attack detection and mitigation. Furthermore, a novel hybrid Coot-Lyrebird optimization algorithm is developed to specifically choose the most impactful features in the network. The selected features are given to the proposed hybrid network that combines Convolutional Neural Network (CNN), SE-ResNeXt, and Long Short-Term Memory (LSTM) networks. Finally, the proposed Deep Q-Network (DQN) model performs attack mitigation measures. The results indicate that the proposed dynamic defense has an accuracy of 0.999571%. In addition, we extended our study to include more complex environments. Software-Defined Internet of Things (SD-IoT) networks enabled intelligent network management through their dynamic features, but expose centralized infrastructure to complex cyberattacks that put the system in great danger. In order to address this, a novel federated secure intelligent intrusion detection and mitigation framework with automated attack reporting for SD-IoT network is presented.

List of Abbreviations

SDN	Software-Defined Networking
SD-IoT	Software-Defined Internet-of-Things
AI	Artificial Intelligence
DDoS	Distributed Denial of Service
IoT	Internet of Things
NFV	Network Function Visualization
API	Application Programming Interface
ONF	Open Networking Foundation
LQR	Linear Quantile Regression
LSTM	Long Short-Term Memory
ML	Machine Learning
FNN	Feedforward Neural Network
FL	Federated Learning
LOA	Lyre-bird Optimization Algorithm
CBO	Coot Bird Optimization
DQN	Deep Q-Network
GRU	Gated Recurrent Unit
CNN	Convolutional Neural Network
ROC	Receiver Operating Characteristic
ViT	Vision Transformer
GraphSAGE	Graph Sample and Aggregation
MA-DQL	Multi-Agent Deep Q-Learning

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Dr. Sudhakar Ganti for his patience, for believing in me and pushing me to achieve my goals. The completion of this thesis would not have been possible without his encouragement and support. I am truly indebted to him for his insightful comments and feedback.

My colleagues, for the help, support and encouragement.

Libyan Ministry of Higher Education and Scientific Research, for sponsoring me with a Scholarship.

DEDICATION

This work is dedicated to my parents, whose wisdom, guidance, and sacrifices laid the foundation for my education and achievements. And to my wife, whose unwavering support, patience, and encouragement have been my greatest strength throughout this journey.

Chapter 1

Introduction

In recent years, having more things connected to the internet has resulted in more network traffic. Securing networks against threats is now an urgent need for many people and businesses [1-4]. Cyber-attackers use the popular method called Distributed Denial-of-Service (DDoS) to bring down the networks. In this type of attack, too much traffic is sent to a target's resource pool, thus hindering legitimate users from accessing important resources. The popularity of SDN for managing networks is due to its centralized control, programmability and agility which it offers. Nonetheless, there is a requirement for smart ways that are flexible enough to address promptly such emerging security challenges. DDoS attacks have become more sophisticated by utilizing amplification techniques as well as through coordination among botnets [5-6]. The usual defense methods based on thresholds or matching patterns may not be enough in SDN setups where traffic can change fast. Moreover, DDoS attacks in business can cause downtime, loss of money, and users dissatisfaction. This is why it is important to have efficient and reliable ways to detect and prevent DDoS attacks. SDN controllers and network devices do not have much computing power like traditional hardware-based solutions. Therefore, it is important to use their resources efficiently so we can quickly detect DDoS attacks without slowing down the network and keep the network running smoothly. If there are delays in stopping these attacks, it can cause longer downtimes and more harm. SDN networks are getting much bigger to handle more devices and users. To deal with this, DDoS defense systems also need to be efficient. They have to be able to manage large amounts of traffic and stop attacks happening across different parts of the network.

The rise of cyberattacks has become a critical concern in today's digital and networked world, requiring creative and flexible ways to protect the networks. As SDN

becomes more popular among businesses due to its programmability and speed, there is a growing need for enhanced security measures in these dynamic environments. The security and integrity of networked systems are constantly in danger due to the constantly changing environment of cyber threats. Modern attacks are more sophisticated, making it challenging for traditional security solutions and insufficient against the dynamic and adaptable nature of modern cyber threats.

Several solutions to these SDN security challenges have been proposed for enhanced network security [7-8]. Similarly, a number of references including a range from policy conflict resolution over controller replication schemes to authentication tools have been presented by the research community. The necessity for additional work to improve network detection of threats and create a secure SDN is obviously identified in most of the previous studies [5-9]. In this thesis, we are specifically targeting the following research questions:

- What are the SDN specific security vulnerabilities both on the data and control planes? In particular, what known or new vulnerabilities of SDN can be abused by attackers?
- How new vulnerabilities in SDN controllers can be exploited through threat vectors and possible solutions, improvements to address these problems?
- How to handle the issue of malicious applications being developed and deployed on SDN controllers?
- How the existing security techniques can be replaced with advanced AI-based approach for a dynamic, secure and intelligent security framework in SDN environment?

These research questions inspired us to look for tools and investigate how we could improve network detection and mitigation of threats in SDN environments.

1.1 Overview of Chapter 2

[1] In this chapter we start with a description of SDN, and how the separation of the control plane from the data plane serves as an effective network management

[1]This chapter is part of the work published in the Cyber Security in Networking Conference, 2024. Paper title: "Dynamic Defense Framework: A Unified Approach for Intrusion Detection and Mitigation in SDN". [49]

technique. This chapter will explain the SDN architecture, its essential components, and the implementation of OpenFlow protocol. The advantages of SDN over the legacy network architecture are vast. We are going to emphasize some of those in Section 2.2. Section 2.3 is dedicated to SDN and security threats. The review of the literature and the security challenges faced by the research community will be laid out in Section 2.4. In Section 2.5, we concluded the chapter with SDN Security and Research gap.

We present this chapter to give an overview of SDN and explains its role in developing new protocols, and control tools. This chapter establishes the foundation for the chapters that follow.

Chapter two contribution:

- Overview of SDN.
- Showing the power of SDN.
- Discussing research gap and the solutions.

1.2 Overview of Chapter 3

[2]

We take lessons learned in the previous chapter to improve security in SDN environment. While the previously discussed methods in chapter 2 offer valuable insights, they exhibit certain limitations that hinder their adaptability to emerging threats. To address these problems, machine learning and deep learning techniques are used to make networks more secure when these kinds of cyber-attacks occur.

In this chapter we developed a more efficient framework, thereby overcoming the common challenge posed by previous research limitations. We proposed a dynamic defense method which includes pre-processing of data, extracting features, filtering features, detecting the attacks and mitigating them. Furthermore, a novel hybrid Coot-Lyrebird optimization algorithm is developed to specifically choose the most impactful features. The selected features are given to the proposed hybrid network that combines Convolutional Neural Network (CNN), SE-ResNeXt, and Long Short-

[2]This chapter is part of the work published in the 12th IFIP International Conference on New Technologies, Mobility and Security. Paper title: "Federated Secure Intelligent Intrusion Detection and Mitigation Framework for SD-IoT Networks using ViT-GraphSAGE and Automated Attack Reporting". [136]

Term Memory (LSTM) networks. Finally, the proposed Deep Q-Network (DQN) model performs attack mitigation measures.

Chapter three contribution:

- The pre-processing, such as outlier detection and normalization, is employed to improve the data quality.
- For feature selection, we combined Lyrebird Optimization Algorithm (LOA) with Coot Bird Optimization (CBO) to identify optimal feature subsets. This improved model has performance with high detection accuracy.
- CNN, SE-ResNeXt, and LSTM are combined to improve the effectiveness and accuracy of intrusion detection system. This leads to better identification and classification of malicious activities and reduce false positives and negatives.
- Deep Q-Network (DQN) mitigation strategy is employed for minimizing the impact of attacks, maintaining service availability, and ensuring network reliability.

1.3 Overview of Chapter 4

In this chapter we take cyberattack detection a step further, where we are looking at more complex attacks in Software-Defined Internet-of-Things (SD-IoT) environment. SD-IoT networks enabled intelligent network management through their dynamic features but expose centralized infrastructure to complex cyberattacks that put the system in great danger.

In order to address this, a novel Federated Secure Intelligent Intrusion Detection and Mitigation framework with Automated Attack Reporting for SD-IoT network is proposed. The combination of Vision Transformer (ViT) and GraphSAGE architecture enables the model to process network relationships globally and locally which effectively increases intrusion detection performance. Besides, to dynamically reroute network traffic and isolate the compromised nodes, a Multi-Agent Deep Q-Learning (MA-DQL) based mitigation strategy is employed in real-time, which minimizes attack impact. For enabling collaborative and secure communication without centralized data exposure, the edge nodes are integrated with Federated Learning (FL) that ensures privacy-preserving and distributed model training. The proposed system also incorporates a Flan-T5 Transformer-based Automated Attack Reporting System

which develops comprehensive forensic reports that expose security threats and their corrective actions. Through this proposed framework, accurate threat detection of 98.06% with real-time adaptative operation and efficient attack containment is accomplished in parallel with reduced computational load which ensures secure operation of SD-IoT systems.

Chapter four contribution:

- Hybrid ViT-GraphSAGE: Intrusion Detection is introduced that combines Vision Transformer for global traffic analysis and GraphSAGE for local node interactions, resulting in improved threat detection accuracy.
- Multi-Agent DQL-Based Attack Mitigation: This enables real-time response by dynamically rerouting traffic and isolating compromised nodes thereby preventing large-scale disruptions.
- Federated Learning and Automated Attack Reporting: this proposed work uses FL for privacy-preserving intrusion detection and Flan-T5 Transformer for generating detailed forensic attack reports.

1.4 Overview of Chapter 5

Inspired by the previous work in Chapters 3, we included additional dataset to the experiments. Expanding the experimentation to include other intrusion datasets to solidify the claim of robustness across different network environments. In Section 5.2, we discussed the importance of features, and the potential limitation of the proposed models are presented in Section 5.3.

1.5 Overview of Chapter 6

As we come to the conclusion of this work, we conclude the dissertation and provide some suggestions for future works in this chapter.

Chapter 2

Software-Defined Networks (SDN)

This chapter provides a brief overview of Software-Defined Networks (SDN) and their functionality, one of the most creative network technologies where network control is separated from forwarding. SDN is directly programmable and has been proposed as a way to programmatically control networks, facilitating the deployment of new applications and services, as well as tuning network policies and performance. The basic idea of SDN is separating the forwarding plane (data plane) from the control plane which is responsible for making forwarding decisions and represents all the logic of the network, whereas traditional networks combine these two planes on the same devices, forcing each device to make the forwarding decision based on routing protocols.

SDN introduces new possibilities for network management and configuration methods which enable us to define and configure the network by programming or using software to define it. OpenFlow [10] is the most representative protocol implementing the SDN concept. It defines standard control interfaces, implement pre- programmed control policy, such as packet-forwarding rules in OpenFlow switches and handle data packet delivery. The programmable network interface gives us a great opportunity to define and configure network in an extremely flexible and efficient way. Thus, the SDN introduces an innovative approach to improve the network performance with both flexibility and compatibility. However, the characteristics of centralized control and programmability linked with the SDN technology present network security challenges. A major example of this challenges is a possible increase for Denial-of-Service (DoS) attacks due to the centralized controller. Another issue of concern linked to security is based on open programmability of the network which may effect the controllers and then the network devices. In addition, the dynamic and open pro-

grammable architecture of SDN provides convenient for attackers to perform effective attacks.

2.1 An overview of SDN architecture and OpenFlow protocol

Software defined networking (SDN) with OpenFlow protocol enables network controllers to determine the route of network packets across a network of switches. The centralized view of the network allows it to control, manage and treat network flows better than the traditional networks. The power is attributed to the programmability of the control plane enabling the application to control how the network behaves. The high level reference SDN architecture promoted by the Open Networking Foundation (ONF) is illustrated in figure 2.1 [10]. It contains three main layers: the application layer, control layer and the infrastructure layer which support the data plane operation. Additionally, the figure highlights two identified reference application programming interfaces, called Northbound API and Southbound API.

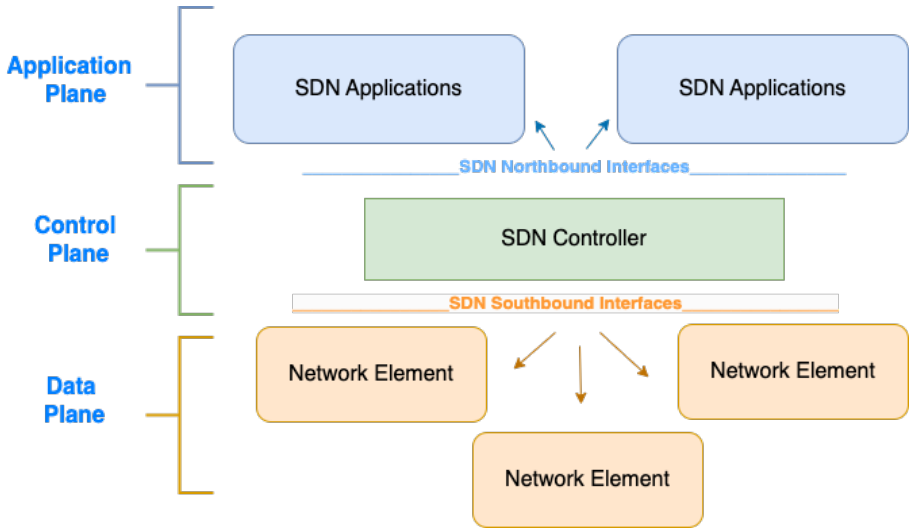


Figure 2.1: SDN Architecture

Unlike traditional networking equipment, SDN physically separates the controllers from the switches. This separation of the control from the forwarding allows for more high-level and advanced traffic management than is feasible using routing protocols or access control lists. The concept of southbound and northbound interfaces has been

introduced in several works, the northbound interface determines the network rules and operational tasks including the control or policy layer, whereas the southbound interface refers to the interface and protocol between programmable switches and the software controller. The primary motivation with this architecture is to simplify the forwarding devices and allow the networking software in the controller to evolve independently.

OpenFlow is one of the most common southbound SDN interface protocol that can remotely control and manages switches from different vendors. Figure 2.2 illustrates an OpenFlow switch that communicates with a controller over a secure connection using the OpenFlow protocol [12].

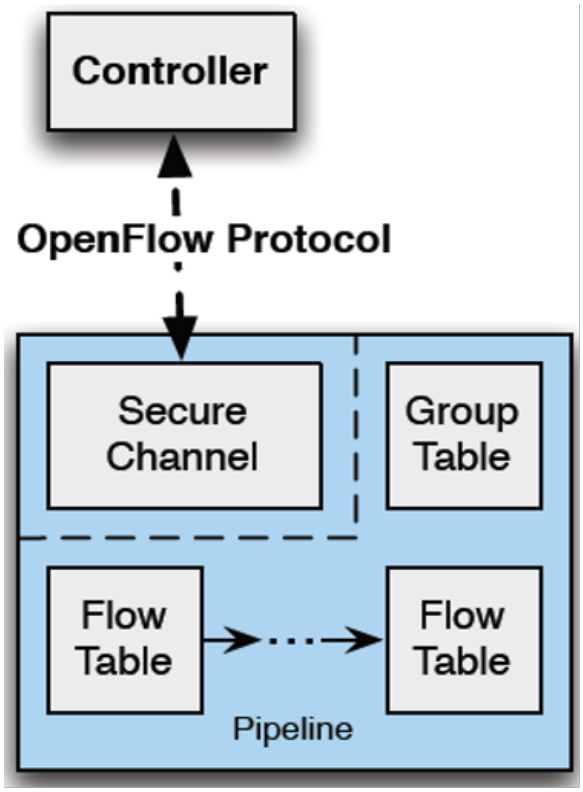


Figure 2.2: OpenFlow Switch [11]

The Open Networking Foundation (ONF) [10] is responsible for standardizing the OpenFlow protocol and describes it as the first standard communications interface which allows remote management of a layer 2 switch's packet forwarding tables, by adding, modifying and removing packet matching rules and actions both reactively and proactively. This way, routing decisions can be made periodically by the controller

and translated into rules and actions with a configurable period of time, which are then deployed to a switch's flow table, leaving the actual forwarding of matched packets to the switch for the duration of those rules. In Openflow, the table architecture is extended to include three major abstract types of information which are [13]:

- Matching rules: The received packet is matched against header fields. Priorities can be specified.
- Counters: When packets arrive at the switch and a match is found, the counters field is incremented. The counters field helps in monitoring and control of flows passed though the switch. Those counters can be received or sent packets, received or sent bytes, drops, errors, and many more.
- Action: This field is where the switch gets instructions on how to deal with flows matched.

Another powerful feature of Openflow is to make a separate table for groups of flows. Group tables are used to address and treat broadcast, multicast, and anycast easily. Those tables have their own parameters but are very similar to normal flows tables. The centralized architecture of Openflow makes it easy to program the controller with tools and algorithms that not only find the shortest path, but find the best path with the current resources. This is possible because the controller have the complete topology information of the network and has the monitoring capabilities to get the current link state. To avoid programming each functionality into the controller, most of the major implementations of the controller are using Applications Programmable Interfaces (API). This is an added layer where applications can configure and query the controller.

2.2 Advantages of SDN

The advantages of SDN over classical networks can be summarized as follows:

- Programmability: The whole idea of bringing the control to software was proposed to enable program controlled networks. This coupled with complete network information, guarantees that decisions are made with the needed knowledge and full control.

- Innovation: As mentioned in previous sections, the research community had a lot of difficulties in testing their new ideas because of the closed nature of vendor devices. With SDN, the devices are used only for forwarding while the controller is programmed separately. This opened up the door for innovation. Now it is easy to build a simple program to take over the network control or even send a simple API REST command to change the behavior of the network. Also, it enabled researchers and technology companies to experiment with new ideas and design the network to serve special needs. For example, Google implemented their SDN version to better control and monitor their WAN network and to cut cost on those links. This brings us to the next advantage of SDN.

- Efficiency: The general view of the SDN provides to the end application establishes a powerful tool to manage the network with good knowledge of its state. When the controller is consulted about a new flow, it checks the current flows it has already installed in the switch, and from that it can decide on which path to take. As a result, the controller logic can build a network with more tolerance (e.g., load balancing), robustness, and security.

- Security and Privacy: In classical networks, if an administrator wants to implement security they need to install a device between two network segments. Doing this isolates the two segments and allows traffic flow between them only if the administrator allows it. In SDN, rules can be implemented in any of the SDN enabled devices since flows can be allowed or dropped. It is easy to see that every device in an SDN network can be treated as a forwarding and security device. This also enables a more in depth and detailed policy enforcement of the intended security model. Another aspect is the separation of traffic, where different users can have different views of the network and its services.

- Management and Monitoring: Network administrators have been using various tools for management and monitoring. Most of them were expensive and hard to learn. Even devices from the same vendor can differ on how they are configured. Also, monitoring needed to enable certain protocols that can take more device resources. With SDN, The protocol is designed with generalized view and control which makes the management process easier and have a broader look. In addition, the counters field incorporated in the flow tables can give abstracted monitoring parameters. The next chapter will investigate how an

SDN network can provide a powerful tool enabling us to measure the latency between communicating parties.

- Virtualization: In the past VLAN and MPLS tried to introduce the concept of multi-tenets networks, but it had been a tough task to automate. With SDN it is easy to create multiple virtual networks on top of one physical network. To take it even a step further one physical network can have multiple controllers on the same network.

- Quality of Service: having a more general view of the network and the ability to provision resources are key aspects in making QoS feasible in SDN. Most of QoS implementations in classical networks are either done per device or by end to end reservation. Both of those techniques have their weaknesses that don't guarantee overall network efficiency. However, in SDN, the full information on network resources can be abstracted to find best path to take where parameters of the application, links, and devices can be taken into consideration.

2.3 SDN and Security Threats

By provisioning highly reactive security monitoring, analysis and response system, the SDN architecture can be enhanced with robust network security. The key to this scheme is the central controller. The data generated from traffic analysis or anomaly detection methods can be regularly transferred to a central controller. Based on the analysis, new or updated security policy can be propagated across the network in the form of flow rules. For instance, it may be possible to use SDN techniques to construct a data plane security solution that is able to coordinate both network and security devices to detect and react to attacks in a more flexible way. This combined approach can efficiently enhance the control and containment of network security threats.

On other hand, a new class of threats arise because a software-defined network explicitly offers programmatic access to the controller plane or applications entities. However, the same attributes of centralized control and programmability associated with the SDN platform introduce network security challenges. An increased potential for attacks and vulnerabilities in SDN include unauthorized access, data modification, data leakage, compromised or malicious applications, denial-of-service (DoS) and configuration issues due to centralized controller and flow-table limitation in network devices is a prime example [14-16]. Concretely, this dissertation aims at raising

awareness about potential vulnerabilities of SDN against a variety of attacks that we can summarize in three main categories:

- Attacks from the controller system: These attacks may be initiated from the system hosting the controller through the exploitation of software vulnerabilities. Such vulnerabilities could enable an attacker to execute arbitrary code or potentially obtain administrative access to the system.
- Attacks from SDN applications: The lack of a trust mechanism between the controller and SDN applications poses a critical security concern, as malicious applications could exploit northbound interfaces to execute unauthorized commands within the network.
- Attacks from hosts: Malicious hosts can launch two primary types of attacks: denial-of-service (DoS) and network topology poisoning, the latter occurring through the compromise of network switches.

A range of authentication mechanisms to controller replication schemes through policy conflict resolution have been proposed in the literature as a solution to these SDN security challenges. Similarly, a number of proposals have been made for enhanced network security. The implementation of SDN networks presents inherent security challenges, which can be summarized in the following key points:

- Controller or Switch vulnerabilities.
- Control channel vulnerabilities.
- Trust between devices and faking of traffic flows.

An analysis of the security challenges of SDN is presented next in this chapter.

2.4 Literature Review

In order to provide a network protected from malicious attack or unintentional damage, we must provide the basic properties of a secure communications network, such as: integrity, confidentiality, authentication and availability of information. There are competing approaches to figure out how SDN security should be deployed, managed, and controlled in SDN environment. Some believe security is best embedded

in servers and other computing devices, while others feel that we should integrate security middle-boxes into SDN and exploit the benefit of programmability to redirect selected network traffic through the middle-box. Regardless, the solutions need to be designed to create an environment that is more efficient, scalable and secure. Moreover, it must be simple and cost-effective to ensure it can be deployed everywhere. It is clear that SDN can implement the security rules very easily by implementing them in the controller. In most cases, it is enough to block certain IP addresses or port access on the network or a portion of the network. So Firewalls and Intrusion Detection Systems (IDSs) can be all implemented on the controller. In addition, the security issues that exist in legacy networks still persist on SDN networks. Despite that, SDN provides powerful tools and views of the network allowing innovation in security generally.

Monitoring and detection systems are essential to protect the network from an attack. In [17], the authors introduced a Distributed Denial-of-Service (DDoS) detection technique based on traffic flow features. This system uses Self Organizing Maps at regular intervals and monitors NOX switches to detect anomalous flows. In an another approach [18], which automatically directs the network packets to be inspected by pre-installed network security devices, OpenSAFE uses a policy language to manage the routing of traffic through network monitoring devices. A similar idea was presented by Shin and Guin by focusing on SDN in the cloud. CloudWatcher [19] controls network flows to guarantee that each essential packet flow is inspected by security devices. In addition, IDS (monitoring system) has been the focus of a number of SDN solutions [20]. Skowyra et al. [20] propose a learning IDS, which utilizes the SDN architecture to both detect and respond to network attacks in embedded mobile devices.

Classic networks use middle-boxes to add and enhance network security functions. Some previous works suggest to integrate security middle-boxes into SDN and get the advantage of programmability environment to redirect selected network traffic through the middle-box. For instance, the Slick architecture [21] proposes a centralized controller, which is responsible for installing and migrating functions onto custom middle-boxes. Applications can then direct the Slick controller to build the needed functions for routing specific flows based on security requirements.

In [22], Shin et al. note the useful features of OpenFlow protocol, especially the visibility that is given to the controller and the possibility of creating applications that run on the top of the controllers which may result in some security vulnerabil-

ity. Therefore, they present a framework particularly for the development of security applications for OpenFlow called FRESCO. The framework is intended to address several key issues that can accelerate the composition of new OpenFlow-enabled security services. FRESCO which itself is an OpenFlow application, offers a programming framework that enables security researchers to implement, share and compose together many different security detection and mitigation modules and then exports a scripting API that enables security professionals to code security monitoring and threat detection logic as modular libraries. These modular libraries represent elementary processing units in FRESCO and also allows for non-OpenFlow application to access the FRESCO abstraction layer. The system also resolves conflicts between different applications based on user privilege by implementing a security enforcement kernel which ensures that non-security applications do not override security applications.

Elubeyd and Yiltas-Kaplan [24] combined a Gated Recurrent Unit (GRU), 1D Convolutional Neural Network, and a Dense Neural Network to automatically detect DoS and DDoS attacks in SDN. This hybrid model helps to spot unusual traffic patterns that signal DoS/DDoS attacks efficiently. Hnamte and Hussain [25] presented Convolutional Neural Network (CNN) to DDoS attack detection in SDN. Employing a CNN, they analyzed network traffic data to discern atypical patterns indicative of potential DDoS attacks. Bhayo et al. [26] developed a SDN-WISE system to identify DDoS attacks in SD-IoT networks by employing machine learning techniques like Decision Tree, Naive Bayes, and Support Vector Machine (SVM) algorithms. Gebremeskel et al. [27] introduced a hybrid Long Short-Term Memory (LSTM) system to detect and classify attacks like DDoS in multicontroller SDN environments.

In [28], Yegench and Ganjali, present another distributed controller compatible with OpenFlow and propose Kandoo which is a framework for preserving scalability without changing switches. This work differs from others in its use of the controller hierarchy with two layers. The bottom layer is a group of controllers with no interconnection and no knowledge of the network-wide state, and responsible only for local decisions and defer any decisions outside of their scope. The top layer is a logically centralized controller that maintains the network-wide state and responsible for the entire network. These controllers handle most of the frequent events and effectively shield the top layer which is useful for defenses against certain types of attacks on the control plane. Kandoo's design allows network operators to replicate local controllers on request and reduce the load on the top layer, which is the only possible

bottle-neck in terms of scalability. It should be noted that security was not one of the main objectives of this work. In [29], a distributed SDN controller that is compatible with OpenFlow and address the issues of scalability and reliability that a centralized controller suffers from is presented. This work makes an important use of OpenFlow specific mechanisms, especially OpenFlow switch management functions. The most important part of the distributed controllers is that the mapping between a switch and a controller is statically configured, which may result in uneven load distribution among the controllers. The researchers propose an elastic distributed controller architecture to address the problem in which the controller pool is dynamically grown according to traffic conditions and the load is dynamically shifted across controllers. Therefore, they have proposed a novel switch migration protocol for allowing such load shifting, which conforms with the Openflow standard. This work along with Kandoo [30] and other distributed controllers can help mitigate certain types of attacks, such as Denial of service attacks (DoS) on the controller plane. However, the prime goal of this technology is to increase scalability and fault tolerance of the control plane rather than to address security issues.

Merlin [29] is one of the first examples of combined framework for controlling different network components. In addition, authors in [30] proposed Frenetic, which provides an interface to query traffic information and can also be used to create a policy to react to network events. Voellmy et al. [31] present Maple to simplify SDN programming by letting a programmer to use a programming language to design an arbitrary centralized algorithm for every packet entering the network. This work consists of two key components, namely an "optimizer" and a "scheduler". The optimizer generalizes rules in the flow table of specific switches by using the data structure to record the invocation of the programmer supplied algorithm on an exact packet. In general, the optimizer uses many methods to enhance the efficiency, including rule priority minimization, data structure compression and augmentation. Moreover, Voellmy et al. [32-34] present Nettle to enhance the reaction to network changes by using a reactive language to describe policies. Nettle merges functional reactive programming (FRP) with domain-specific languages (DSLs). A part of Nettle platform takes network events as input and then generates rule updates for specific usage, for instance, new user detection and user authentication.

SDN controller is responsible for generating packet forwarding rules defining the actions to be taken by installing them into appropriate switching devices. Furthermore, as a result of configuration changes and dynamic control, forwarding rules in

switching devices need to be updated at the same time to keep up with events such as, traffic rerouting for load balancing or network recovery after unexpected failure. However, flexibility is an essential factor that rule updates should preserve to guarantee decent active network operation. In this category, Reitblatt et al. propose a strict consistency implementation that links versioning with rule timeouts [34-35]. The basic idea is to mark each packet with a version number at its access switch stating which rule set should be applied. Then, the packet will be handled depending on the version number, and thus the next packets will be marked to act on the updated rule set. Thus, after a period of time, no more packets will act on the original rule set and then will be expired and removed. In [37] McGeer et al. focus to preserve switch memory space by ensuring that only a specific set of rules are present in a switching device at any time. When a new policy is about to be executed, each switching device is notified to direct targeted packets to a controller. Then the controller based on the policy, generates new packet forwarding rules and substitutes rules in the switching devices with these new rules. The affected packets will be buffered in the controller until the rule-replacements are finished and then sent back to switching devices for processing. In [38] authors proposed LiveSec, which is an OpenFlow-based agile architecture for network security management in production networks. The LiveSec framework can enable event replay besides application-aware monitoring and interactive policy enforcement by visualizing the environment of the network.

Active networking was proposed to enable programmability of switches through user injected programs [39-41]. In active networks, switches do a customized calculation on the payload that permits through them. Hence, the switches can be tailored to function according to user or application requirements. A main threat for active networks was to secure active switches from malicious applications. As a result, active security and other security approaches were proposed to ensure security through authentication and authorization mechanisms. In [42] Greenberg et al, associate the fragile nature of communication networks to the complex nature of control and management planes in traditional networks. It is illustrated that the lack of coordination between routing and security mechanisms result in a fragile network and security lapses. From this viewpoint, a new approach is proposed called the 4D approach, which totally separates the network control from the forwarding substrate. The authors in this work propose that a network architecture should be based on three key basis, which are: network-level objectives, network-wide views, and direct

control. In [43] Hartman et al. pointed out three main classification of applications that can affect network security in SDN. First, applications that provide inspector container or intrusion detection services, such as firewall or access control. Second, network sensitive applications that need specific network characteristics such as path characteristics or cost of traffic flows. Finally, packaged network services that mix applications from the first and second classes. Therefore, a malicious application can avoid firewalls and access controls by using an instance of the first class application.

The main capabilities of effective security are built on five security scales and components. First, providing security against ordinary attack scenarios and building a context-aware security system which offers protection through configuration of the infrastructure using protection procedures. Second, notify the reactive security system by applying a detection system through different sources or sensors that perform some detection and monitoring such as IDS. Third, modify the network dynamically by using the SDN controller to better defend or monitor the network in future. Fourth, in order to understand the attack, a collection of attack statistics to perform forensics evidence gathering. Finally, close monitoring and attack observation, and then responding by consuming the attacker's resources and limit his ability to continue attacks. In [44] authors present OrchSec, an orchestrator-based architecture which aims to improve network security using network monitoring and SDN control functions. The authors of OrchSec have proposed decoupling of the control and monitoring functionalities in SDN. In OrchSec, the controllers are responsible only for issuing flow rules, while network monitors are responsible for doing monitoring functions. In a similar scheme [46], applications are not implemented inside the controllers but rather developed as Northbound applications to provide independence to applications from the core controller architectures. OrchSec based on network-wide security requirements provides monitoring process, such as; varying sampling rates or changing responsibilities between monitors and controllers.

Vinayakumar et al. [45] examined the application of neural network frameworks in traffic monitoring, detection of intrusions, and malware identification for Android in cybersecurity. showed that because of their resilience in hostile settings and capacity to acquire the best feature participation, these designs performed better in these domains than traditional machine learning methods. Nevertheless, non-uniformity and artificial artifacts were among the drawbacks of the KDD-Cup-99 challenge dataset, which was utilized to evaluate machine learning classifiers. The study classified and extracted features for intrusion detection using algorithms for deep learning such as

LSTM, recurrent neural networks, and sparse autoencoders. The study concluded that a voting mechanism might enhance performance and that actual time detection of intrusions is an important path for further investigation. Iqbal et al. [46] studied SDN security concerns and suggested ways to strengthen the controller's security architecture. Methods for protecting the GUI and SDN environment, including role-based authorization (FortNOX), SSL/TLS integration, logging/security audit services, and encryption (AES and DES). The precise setup and execution of the SDN surroundings, however, could have had an impact on how well these solutions worked. According to the study, these methods might improve the safety framework of the controllers in SDN systems. It would have been possible to do more study to see how well these methods work in practical deployments as well as look into other security measures to counter new threats and weaknesses. Providing an evaluation of security and suggesting potential fixes to enhance SDN security were the main points of emphasis.

Boukria and Guerroumi [47] proposed SDN security by identifying and averting bogus information insertion threats in the link that communicates between the SDN infrastructural layer and the SDN control layer. For flow categorization and intrusion detection, the suggested system employed a deep learning methodology that included a logarithm function, the min/max scalar technique, Relu, and softmax functions. The study also looked at multi-controllers and blockchain integration as additional security measures for the SDN network. The efficacy and efficiency of the suggested safety measure were shown by the assessment results on an experimental platform utilizing the CICIDS 2017 dataset.

Sivanathan et al. [48] investigated the use of characteristics in choice tree-based machine learning methods for IoT state classifiers, traffic categorization, and precise prediction. used the correlation-based feature subset (CFS) algorithm with a best-first searching strategy to identify key characteristics and the information gain (IG) approach to quantify attribute weight. Additionally, the research recommended developing more sophisticated IoT state classifiers and investigating IoT traffic categorization in more detail. The discoveries may have a number of uses for machine learning.

In summary, SDN mainly relies on several software tools in the application layer to interact with the underlying infrastructure. This surely has major security drawbacks as code vulnerabilities can have a serious security impact. Additionally, by making the controller as centralized for management and control would result in creating a point

of failure for the whole system. Controllers could be threatened from both interfaces: the southbound and the northbound interfaces are respectively the entrance doors to the control, which themselves might be an appealing target for attackers. The data plane is also a potential target for attackers. Because of the exchanges between the control and data planes, an attacker could disrupt the data flow by flooding the switches or tampering the communications.

2.5 SDN Security and Research Gap

Although Software-Defined Networking (SDN) has introduced significant advancements in network security, several critical research gaps remain unaddressed. Security challenges persist, particularly concerning the northbound and east-west interfaces, trust management among network entities, and the mitigation of threats posed by malicious SDN applications. The integration of artificial intelligence into SDN environments also faces limitations, notably in areas such as AI-based intrusion, federated learning, and the development of safe reinforcement learning strategies for dynamic and adaptive network control. The scalability of the control plane, optimal placement of controllers, and the resilience of SDN infrastructures against failures and disruptions also require further investigation. Additionally, as SDN expands into emerging domains such as the Software-Defined Internet of Things (SD-IoT), there is a growing need to design enhanced SDN frameworks capable of meeting stringent demands for scalability, security, low latency, and seamless adaptability.

Specific limitations and research gaps identified in the existing literature will be critically examined and discussed in Chapters 3 and 4.

Chapter 3

Dynamic Defense Framework: A Unified Approach for Intrusion Detection and Mitigation in SDN

[1]

3.1 Introduction

In recent years, having more things connected to the internet has resulted in more network traffic. Securing networks against threats is now an urgent need for many people and businesses. Cyber-attackers use the popular method called DDoS (Distributed Denial of Service) to bring down networks. In this type of attack, too much traffic is sent to a target's resource pool, thus hindering legitimate users from accessing important resources [49-53]. The popularity of SDN for managing networks is due to this centralized control, programmability and agility which it offers. Nonetheless, there is a requirement for smart ways that are flexible enough to address promptly such emerging security challenges. The usual defense methods based on thresholds or matching patterns may not be enough in SDN setups where traffic can change fast. Moreover, DDoS attacks in business can cause downtime, loss of money, and users dissatisfaction. This is why it is important to have efficient and reliable ways to detect and prevent DDoS attacks without slowing down the network and keep the

[1]This chapter is part of the work published in the CSNet24: Cyber Security in Networking Conference 2024 [49]

network running smoothly. To address these problems, deep learning technique is used to make networks more secure when these kinds of cyber-attacks occur. The main contributions of this work include:

- The pre-processing, such as outlier detection and normalization, is employed to improve the data quality.
- For feature selection, we combined Lyrebird Optimization Algorithm (LOA) with Coot Bird Optimization (CBO) to identify optimal feature subsets. This improved model has performance with high detection accuracy.
- CNN, SE-ResNeXt, and LSTM are combined to improve the effectiveness and accuracy of intrusion detection system. This leads to better identification and classification of malicious activities and reduce false positives and negatives.
- Deep Q-Network (DQN) mitigation strategy is employed for minimizing the impact of attacks, maintaining service availability, and ensuring network reliability.

The remaining sections are arranged as follows: section 2 provides the review of the existing techniques, section 3 explains the proposed methodology in detail, section 4 gives the experimental results, and section 5 concludes the chapter.

3.2 Related Work

Elubeyd and Yiltas-Kaplan [24] combined a Gated Recurrent Unit (GRU), 1D CNN, and a Dense Neural Network to automatically detect DoS and DDoS attacks in SDN. This hybrid model helps to spot unusual traffic patterns that signal DoS/DDoS attacks efficiently. The 1D CNN is good at grasping how different parts of network traffic relate. The GRU is great at understanding changes over time. Then, the DNN is employed as the final decision maker by putting together all this information for accurate predictions. By using these combinations of deep learning models, their method shows better improvements in detecting DDoS or DoS attacks in SDNs.

Hnamte and Hussain [25] presented a CNN to DDoS attack detection in SDN. Employing a CNN, they analyzed network traffic data to discern atypical patterns indicative of potential DDoS attacks. Bhayo et al. [26] developed a SDN-WISE system to identify DDoS attacks in SD-IoT networks by employing machine learning

techniques like Decision Tree, Naive Bayes, and Support Vector Machine (SVM) algorithm. Their module comprises of IoT nodes and it oversees incoming IoT traffic while acting as an intermediary between the source IoT node and the controller. It further includes the use of machine learning (ML) based detection model for classifying the kind of DDoS packets in the IoT node traffic.

Gebremeskel et al. [27] introduced a hybrid LSTM system to detect and classify attacks like DDoS in multicontroller SDN environments. They use multiple network features used in calculating Entropy and they are; Destination IP address, Source IP address, Port numbers. Therefore, anomalies are detected by checking the change in entropy values against pre-established criteria. This LSTM model achieved an accuracy of 99.42% on the CICDDoS2019 dataset.

According to Wang et al. [54], one way of dealing with controller congestion problems is by employing a switch migration module which makes the transmission of network flows smooth. In order to prevent detection module downtimes due to controller failure, a switch migration module is proposed. This ensured that CPU processing ability is provided for future operation of the detection module. Furthermore, an anomaly detection module employing a two-stage approach for detection was been developed to identify DDoS attack patterns. Decreasing mislabeled or missmarked data in SDN settings help improve the steadiness of detecting DDoS attacks with many steps. By working together amidst controllers, it can single out and block anomaly flows with the aim to conserve the network performance and safety. It also eliminates cross-domain flow exceptions in the face of an anomaly through the use of cooperating modules. Also, a blacklist for abnormal flows efficiently blocks known malicious traffic.

A conditional entropy based method was proposed by Tian and Miyata [55] to detect DDoS attacks by analyzing traffic in SDN. The method is based on the conditional entropy where different states are detected using the distribution of entropy values when non-attack periods are considered. Moreover, the performance is enhanced by placing entropy values in normal and abnormal traffic behavior categories. Their method works well, because it can detect even little changes in entropy of network operations, displaying those values that reflect data flow complexity and regularity in SDN. With accurately set thresholds derived from average entropy dispersion, it is more precise than others in recognizing deviations indicating DDoS assaults. This also accomplishes dependable classification between typical and extraordinary traffic modes through utilization of entropy.

According to a report from Chaganti et al. [56], a new method based on LSTM neural network has been developed to detect attacks within IoT network with the aid of SDN. By means of this new method, researchers suggested an LSTM-based model that can interpret data flows being transmitted through it, looking out for peculiarities that might suggest break-ins. This way they managed to train their proposed model with relevant information from the network to give a detection rate of up to 97.1% in security-sensitive operations associated with intrusions.

3.3 Methodology

The proposed methodology has several key stages. Initially, standard data preprocessing techniques such as missing data handling and outlier detection are performed. Feature extraction techniques are then applied to extract relevant information from the data. Then the Lyrebird Optimization Algorithm (LOA) is combined with Coot Bird Optimization (CBO) algorithm to identify optimal feature subsets. The selected features are given to the dynamic defense attack detection network, which is developed using Convolutional Neural Networks (CNNs), SE-ResNeXt, and Long Short-Term Memory (LSTM). This dynamic defense model enhances the accuracy and efficiency of intrusion detection systems by capturing complex patterns and anomalies in network traffic. Additionally, the methodology includes strategies for dynamic attack mitigation to minimize the impact of cyber threats on SDN networks.

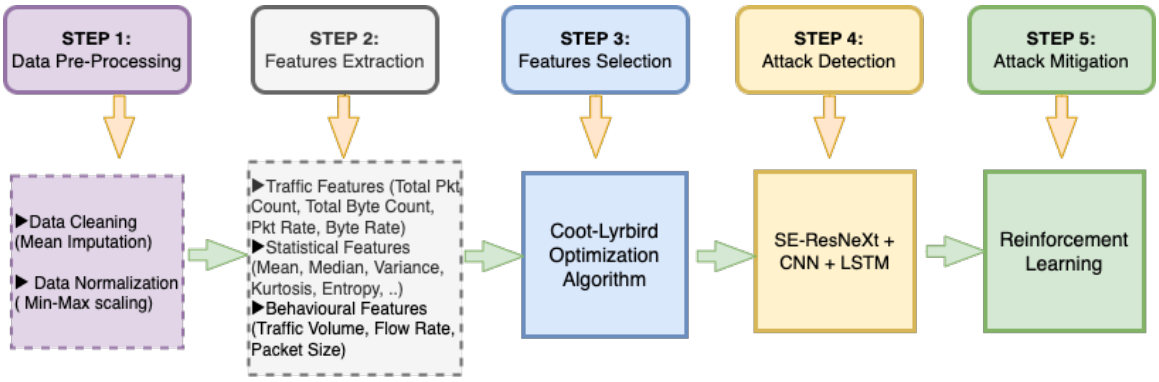


Figure 3.1: Block diagram of the proposed Dynamic Defense method

Implementing AI-driven security into a Software-Defined Networking (SDN) controller involves integrating machine learning (ML) and artificial intelligence (AI) techniques to enhance network monitoring, intrusion detection, and automated threat

response. The step-by-step methodology used in this study is depicted in Figure 3.1. As a first step, we address issues in the dataset, such as missing values, noise, and feature scaling, thereby improving model accuracy, generalization, and computational efficiency. As a second step, we implemented feature extraction, which enables more efficient and effective learning by extracting relevant features from the pre-processed data. In the next step, we combined Lyre-bird Optimization Algorithm (LOA) with Coot Bird Optimization (CBO) for feature selection. The combination of these two algorithms brings together their respective strategies to speed up the feature selection process, while also significantly improving the probability of efficiently identifying optimal feature subsets and detecting the most impactful features. In Step 4, we develop the Hybrid Deep Learning model for attack detection. This model is developed to detect intrusion in SDN environments by classifying the incoming flows in real time with labels (such as; normal flow, attack or malicious traffic). If an attack is detected, the Deep Q-Network (DQN) agent will carry out the attack mitigation process as explained in Step 5. This comprehensive framework will be explained in depth in this chapter.

3.3.1 Data Pre-Processing

This is the initial stage of the method that refers to the transformation that dataset undergoes before being fed to next step. Data pre-processing is very important as it will help us getting a cleaner data to improve the training and increases the accuracy and efficiency of machine learning models. Machine learning algorithms rely on high-quality data for effective learning and generalization. Raw data collected from real-world sources often contain inconsistencies, missing values, noise, and irrelevant attributes that can adversely affect model performance. In this work, the data pre-processing involves the following steps:

Data Cleaning

Missing Data Handling (Mean Imputation): The missing values are replaced with the mean or median of the feature.

$$\text{Mean } = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.1}$$

where x_i signifies non-missing values, and n indicates the sum of non-missing values.

For Outlier Detection and Removal, we use a statistical method called Interquartile Range or IQR. Using the IQR, the difference between the 75th (Q_1) and 25th percentiles (Q_3) is determined and data points outside the specified range of the dataset are removed as outliers.

$$IQR = Q_3 - Q_1 \tag{3.2}$$

$$UB = Q_3 + (1.5 \times IQR) \tag{3.3}$$

$$LB = Q_1 - (1.5 \times IQR) \tag{3.4}$$

Data points outside the range of (LB, UB) are considered outliers and are removed.

Data Normalization

By using Min-Max Scaling technique, feature values are normalized to a common range $[0, 1]$ to ensure consistent analysis across features.

$$\text{Scaled value } = \frac{x - \text{Min}(x)}{\text{Max}(x) - \text{Min}(x)} \tag{3.5}$$

3.3.2 Feature Extraction

From the pre-processed data, we implement Step 2 (Feature Extraction) which involve recognizing and extracting relevant features from the pre-processed dataset. These features offers greater informative data. The following are some of the features that are extracted.

Traffic Features

- Total Packet Count: This refers to the total number of packets transmitted within a specific time interval. It is an important metric for assessing network activity and performance.
- Total Byte Count: It shows how much information (in bytes) that goes through within particular period of time. Monitoring this measure will help us know how much data is moving around in the network.
- Packet Rate: The rate at which packets are transmitted (*packets per sec*).
- Byte Rate: The rate at which data is transmitted (*bytes per sec*).

Statistical Features

- Mean: It is computed by adding up all the values in a dataset and then dividing this sum by the total number of values.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.6}$$

  where n denotes the number of data points and x_i denotes each individual data point.

- Median: The median is the central value in a set of data when organized in ascending order. If n is odd:

$$M = x_{(n+1)/2} \tag{3.7}$$

  If n is even:

$$M = \frac{x_{(n/2)} + x_{((n/2)+1)}}{2} \tag{3.8}$$

- Standard Deviation: Measuring the standard deviation helps to assess how widely data points are spread around the mean.

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (x_i - \mu)^2} \tag{3.9}$$

  where μ represents the mean.

- Variance: The variance signifies the mean of the squared deviations from the average.

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu)^2 \tag{3.10}$$

- Skewness: The skewness shows whether data points in a given distribution are distributed symmetrically around the average. If there are too many data points that pile to the left of the average, it means the skewness is negative. If there are too many data points which pile on the right of the average then it means the skewness is positive.

$$\text{skewness} = \frac{\frac{1}{n}\sum_{i=1}^{n} (x_i - \mu)^3}{\sigma^3} \tag{3.11}$$

- Kurtosis: Kurtosis tells us about the shape/symmetry of the data distribution. Higher kurtosis means that the average of the fall is larger than that of a normal distribution.

$$\text{kurtosis } = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^4}{\sigma^4} - 3 \tag{3.12}$$

- Entropy: The degree of uncertainty or disorder in a set of information is a measure called entropy. Increased entropy means more disorder or randomness that helps detect unnatural or anomalous patterns.

$$\text{Entropy } = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i) \tag{3.13}$$

Behavioral Features

- Traffic volume: The measurement of traffic volume consists of examining variations in the basic rates during the given time intervals and abrupt changes may indicate unusual activity.

- Flow Rate Analysis: Flow rate analysis involves observing the time it takes for data traffic to flow through the network.

- Packet Size: When we focus on packet size, we get to know more about how different sizes are distributed across all packet sent on the internet. DDoS attacks involve abnormal packet size distributions.

3.3.3 Feature Selection

Feature selection is a fundamental aspect that plays a critical role in enhancing model performance and improving interpretability. It involves identifying and selecting the most relevant features from a dataset while discarding redundant or irrelevant ones. This process is particularly important and crucial for the proposed framework, where from the extracted features, the most significant ones are selected by the proposed novel hybrid Coot-Lyrebird optimization Algorithm, which will be presented and discussed in detail next in this chapter.

Coot Bird Optimization (CBO)

The COOT algorithm has both random exploration and focused exploitation within the feature space [57]. In this algorithm, the leaders emerge based on successful feature combinations and guide others towards better areas. Successful features attract other candidates which lead to convergence on promising combinations. Through repetition and adjustment, this method uses cooperative dynamics to effectively navigate the feature space. Therefore, the exploration and exploitation is balanced to improve the performance of the detection model. Next, for clarity, we are going to refer to the feature as (individual).

Population size (N): The number of coots in the population directly influences the complexity. If there are N individuals, initializing their positions and fitness values requires $O(N)$ time.

Feature space (d): The number of features or decision variables, denoted as d, affects the initialization process since each individual has to be initialized with a position in d-dimensional space, which costs $O(N \times d)$.

In Eqn (3.14), the subscript i represents the current individual being considered. The term CP_i refers to the position of this individual. The optimization problem has d variables, each with lower and upper bounds denoted by L_b and U_b. The fitness of each individual in the optimization problem is computed using Eqn. (3.15).

$$CP_i = Rd(1, Dv) * (U_b - L_b) + L_b \tag{3.14}$$

$$Fit = min(Error) \tag{3.15}$$

- *Random movement:*

During the optimization process, a random spot called RP is chosen within a specific range. This range is set by the U_b and L_b , as described in Eqn. (3.16). The location RP is then used to adjust the position of the individual currently being optimized.

$$RP = Rd(1, Dv) * (U_b - L_b) + L_b \tag{3.16}$$

By using random walks to adjust the current position, the algorithm can escape from local minima. The position update equation, given in Eqn. (3.17), determines the

new location of the individual in the search space.

$$CP_i = CP_i + B \times Rd_2 \times (RP - CP_i) \tag{3.17}$$

Eqn. (3.17) includes a random value Rd_2 between 0 and 1.

$$B = 1 - I_{\text{cur}} \times \left(\frac{1}{I_{\max}}\right) \tag{3.18}$$

Using Eqn. (3.18), a numerical value is computed that is influenced by the present iteration I_{cur} and the highest possible number of iterations I_{max}.

During the random movement phase, each individual randomly updates its position in the feature space. The update involves computing new positions using random values. This operation requires $O(1)$ time per individual. Therefore, for N individuals, the complexity is $O(N)$.

Fitness Evaluation: The fitness of each individual needs to be evaluated after each movement, which depends on the complexity of the objective function. If evaluating the fitness function takes $O(f(d))$, where $f(d)$ is the complexity of evaluating the objective function, the total complexity for N individuals will be $O(N \times f(d))$.

- *Chain movement:*

When employing chain movement, the distance vector between two individuals is determined. Consequently, the first individual moves towards the second by approximately half the distance. Using the chain movement method, the new location for the current individual is calculated using Eqn. (3.19).

$$CP_i = 0.5 \times (CP_{i-1}) + CP_i \tag{3.19}$$

Typically, within a group, a small number of individuals modify their locations to approach desirable areas. The remaining group members then alter their own positions based on the movements of these leaders. The adjustment procedure used in the COOT technique is defined by Eqn, (3.20).

$$L = 1 + (iMODT_{ld}) \tag{3.20}$$

In chain movement, the distance vector between two individuals is calculated, and one individual moves towards another. This requires computing the position updates

using Eqn. (3.19) and Eqn. (3.20). Since these involve basic arithmetic operations, the time complexity for each individual is $O(1)$, and for N individuals, the overall complexity remains $O(N)$.

In the COOT technique, there are N coots or individuals, and the leader is identified by the number $T_l d$. Each coot uses information from its assigned leader, L to determine its new position. The coot's new location is influenced by the location of the leader l.

$$CP_i = LP_l + 2 \times Rd_3 \times \cos(2\pi Rd)\,(LP_l - CP_i) \tag{3.21}$$

where l represents the current leader's identification number, LP indicates the position (location) of the chosen leader, Rd is a random number between 0.5 and 1.

- *Leading Group:*

In optimization, individuals in the population must stay within the search area. If any individual's dimension falls outside this area, it must be adjusted to bring it back within the boundaries. In the COOT method, individuals maintain their position based on the location of group leaders. So, group leaders must be within the search area. In COOT, group leaders update their position using Eqn. (3.22).

$$LP_i = \begin{cases} C \times Rd_4 \times \cos(2\pi Rd) \times (G_{\text{best}} - LP_i) + G_{\text{best}}\,, & Rd_5 < 0.5 \\ C \times Rd_4 \times \cos(2\pi Rd) \times (G_{\text{best}} - LP_i) - G_{\text{best}}\,, & Rd_5 >= 0.5 \end{cases} \tag{3.22}$$

The G_{best} point defines the location of the individual with the highest performance. Random numbers Rd_4 and Rd_5 both lie within the range of 0 to 1. The value of C is computed using the formula provided in Eqn. (3.23).

$$C = 2 - I_{\text{cur}} \times \left(\frac{1}{I_{\max}}\right) \tag{3.23}$$

Lyrebird Optimization Algorithm (LOA)

LOA approach is a population-based metaheuristic algorithm where lyrebirds are used as the population, and it can efficiently solve optimization problems using an iterative approach [58]. Inspired by the unique survival strategies of the lyrebird, LOA balances exploration and exploitation through two key mechanisms: escape and hiding strategies [58]. As a matter of fact, each lyrebird serves as an individual member in LOA hence decides various decision variables depending on their position within

problem solving space. Optimization algorithms play a crucial role in solving problems by identifying the best possible solutions in a given search space. LOA leverages two fundamental strategies to achieve an optimal balance between exploration and exploitation, which will be presented next.

- *Escaping Strategy (Exploration):*

For optimal feature selection using the LOA, the update phase adjusts the position of population members based on the behavior of a lyrebird escaping from threat to secure areas. This procedure copies the algorithm in global search as it extensively travels through different parts of the problem-space. In feature selection problems that employ LOA method, every population individual identifies areas occupied by other members with better objective function values as safe areas. These safe areas represent regions in the feature space where promising feature subsets or configurations exist. It is mathematically defined using Eqn. (3.24).

$$SA_i = \{Y_l, \quad F_l < F_i \text{ and } l \in \{1, 2, \ldots, M\}\} \tag{3.24}$$

Here, we denote orientation area (SA_i) that is safe for the i^{th} lyrebird and Y_l is the l^{th} row of the matrix Y with a better performance quality (better objective function value) F_i than the i^{th} LOA. It is hypothesized that the lyrebird flies at random into any of these safe areas in the LoA patterns. Therefore, the new location is discovered using Eqn. (3.25) based on this stage's lyrebird dislocation model. After that, if the fitness is improved, the new location is substituted for the old location of the same agent as per Equation (3.26).

$$y_{i,j}^{P1} = y_{i,j} + s_{i,j} \cdot (SSA_{i,j} - J_{i,j} \cdot y_{i,j}) \tag{3.25}$$

$$Y_i = \begin{cases} Y_i^{P1}, & F_i^{P1} \leq F_i \\ Y_i, & \text{else} \end{cases} \tag{3.26}$$

The chosen secure area for i^{th} lyrebird is known as $SSA_{i,j}$, with the j^{th} dimension, with Y_i^{P1} representing the fresh position computed for this bird. Using the technique of evasion from LOA as the realizing strategy, $y_{i,j}^{P1}$ denotes its j^{th} dimension, and F_i^{P1} stands for the objective function value, and the symbols $s_{i,j}$ are random numbers sampled from an interval ranging between zero percent (inclusive) and one hundred

cumulative percent (exclusive), while those numerals denoted by $J_{i,j}$ are either the first or second digits randomly chosen.

The escaping strategy involves updating the positions of the lyrebirds based on a random movement to a safe area. This can be computed in $O(1)$ per individual, and thus for N individuals, the complexity is $O(N)$. Fitness evaluation for the new positions also adds $O(N \times f(d))$.

- *Hiding (Exploitation):*

The Lyrebird updates its location in search space similar to how lyrebirds crawl into their habitats by carefully studying surroundings and moving a few steps at a time to get to a point to conceal itself, causes the changes to lyrebird's locations to be small when applied to local search by LOA demonstrating that it exhibits exploitation ability.

A new location for each Lyrebird is calculated by equation (3.27), based on the lyrebird's movement model towards the near-optimal hiding area, according to LOA design. This replaces the old position of that member if it improved the objective function value, as per equation (3.28).

$$y_{i,j}^{P2} = y_{i,j} + (1 - 2s_{i,j}) \cdot \frac{UB_j - LB_j}{I} \tag{3.27}$$

$$Y_i = \begin{cases} Y_i^{P2}, & F_i^{P2} \leq F_i \\ Y_i, & \text{else} \end{cases} \tag{3.28}$$

Here, Y_i^{P1} which refers to the new location computed for the i^{th} lyrebird using the hiding principles given by suggested LOA, $y_{i,j}^{P2}$ is the j^{th} dimension of Y_i^{P2} , while F_i^{P2} represents its function value, and $s_{i,j}$ denotes randomly generated numbers from the range [0,1], with I being the iteration counter. The hiding strategy also involves updating the position of each individual based on small steps toward better positions. The time complexity is similar to the escaping strategy, $O(N)$ for updating the positions, and $O(N \times f(d))$ for evaluating the fitness.

As mentioned in the introduction section, for feature selection, we combined Lyrebird Optimization Algorithm (LOA) with Coot Bird Optimization (CBO). The combination of these two algorithms brings together their respective strategies to speed up the feature selection process while also significantly improving the probability of

efficiently identifying the optimal feature subsets and detecting the most impactful features. This adaptive approach maximizes predictive accuracy and model performance. This novel hybrid Coot-Lyrebird Optimization algorithm structure is shown below:

Algorithm 1 Hybrid Coot-Lyrebird Optimization

1: **Initialization:** Initialize the population with Cootbird individuals.
2: **Exploration (Cootbird):** For each Cootbird individual, perform random movement using Eqn. (3.14) to (3.18) to explore the feature space. Apply chain movement based on Eqn. (3.19) to (3.21) to adjust the position towards promising areas.
3: **Exploitation (Lyrebird):** For each Cootbird individual: Use Eqn. (3.27) to compute the new position based on the hiding strategy from Lyrebird Optimization Algorithm. If the new position improves the objective function value, update the position using Eqn. (3.23).
4: **Fitness Evaluation:** Determine the fitness of each individual using Eqn. (3.15).
5: **Selection:** Select individuals based on their fitness values to form the next generation.
6: **Termination:** Repeat steps 2 to 5 until a termination condition is met.

For each individual in both COOT and Lyrebird methods, the position is adjusted based on the leader or the best individual in the population. This requires identifying the leader (which involves scanning the population for the best fitness value), which has a time complexity of $O(N)$. Position updates based on the leader will take $O(1)$ per individual. The optimization process repeats for T iterations. During each iteration, several operations are performed on all individuals, including position updates and fitness evaluations. Thus, the time complexity for each iteration is $O(N \times f(d))$. Therefore, the total time complexity of the hybrid Coot-Lyrebird algorithm for TTT iterations is $O(T \times N \times f(d))$. Combining all the operations, the overall computational complexity of the hybrid Coot Lyrebird Optimization algorithm is $O(T \times N \times f(d))$, where, T is the number of iterations, N is the population size (number of coots and lyrebirds), $f(d)$ is the complexity of evaluating the fitness function with respect to the number of features d. This time complexity reflects the number of iterations, the size of the population, and the complexity of the objective function evaluation. The effectiveness of the algorithm depends on balancing the population size, the number of iterations, and the computational cost of the fitness function.

3.3.4 Attack Detection - Hybrid Deep Learning

For attack detection, CNNs, SE-ResNeXt, and LSTM networks are combined. This combination employs the benefits of each component for spatial feature extraction, attention-based feature enhancement, and temporal modeling, respectively.

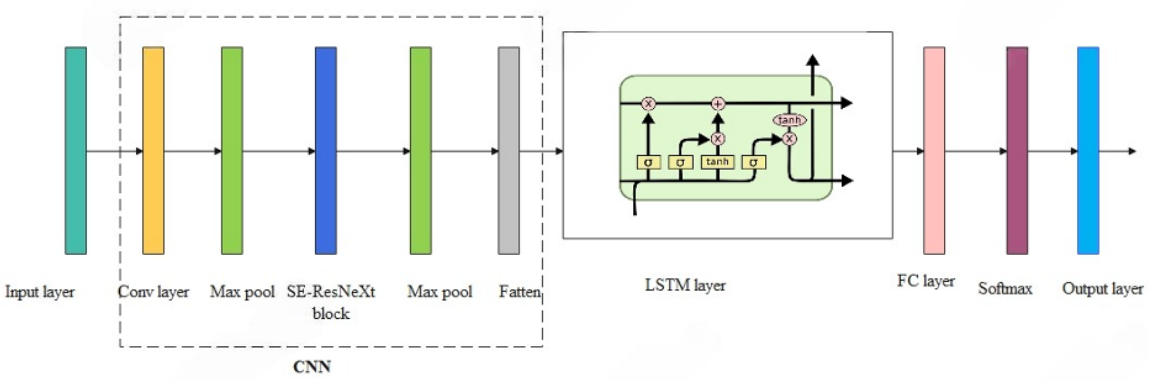


Figure 3.2: Architecture of the Proposed model

Input Layer

The input layer receives raw network traffic data or preprocessed features as input to the model.

Convolutional Layer (Conv)

Convolutional layers are effective for spatial feature extraction from the input data. They capture local patterns and spatial relationships in the network traffic data.

$$\text{Conv}(X) = \sigma(f * X + b) \tag{3.29}$$

where X represents the input feature map, f signifies the set of learnable convolution filters, $*$ denotes the convolution operation, b denotes the bias term, and σ represents the Rectified Linear Unit (ReLU) activation function. ReLU activation function operates element by element, which is fast and straightforward to implement, and leads to faster training times compared to other kinds of activation functions.

Max Pooling Layer

Max pooling layer downsamples the spatial dimensions of the feature maps obtained from the convolutional layers. It reduces the computational complexity and extracts the dominant features while preserving spatial information.

$$\text{MaxPool}(X) = max(X_{y,j}) \tag{3.30}$$

where $X_{y,j}$ denotes the elements in the pooling window.

SE-ResNeXt Block

The SE-ResNeXt block [80] enhances feature representation through channel-wise attention mechanisms. It recalibrates channel-wise feature responses, focusing on informative features and suppressing noise.

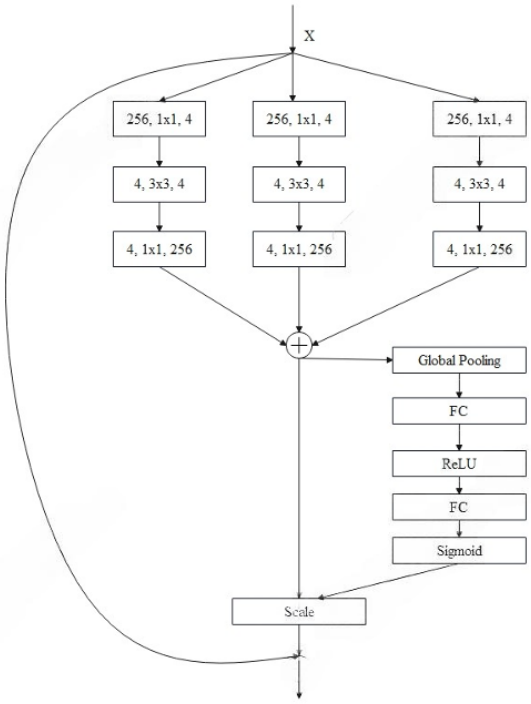


Figure 3.3: Architecture of the SE-ResNeXt Block

Max Pooling Layer

Additional max pooling further downsamples feature maps and reduces dimensionality before subsequent layers. This helps to capture more salient features and reduce the computational burden.

Flatten Layer

The flatten layer reshapes the output from the convolutional layers into a vector format. It prepares the feature representation for input into subsequent layers, such as LSTM and fully connected layers.

$$\text{Flatten}(X) = reshape(X, shape = (batch - size, -1)) \tag{3.31}$$

This operation reshapes the output of the previous layer into a 1D vector while maintaining the values.

LSTM Layer

LSTM layer model temporal dependencies in sequential data, such as network traffic. They capture long-range dependencies and temporal patterns, which are crucial for intrusion detection. LSTM cell state (c_t) plays a key role in preserving information throughout the time steps, and it consists of several gates and operations:

- Forget gate:

$$f_t = \sigma\left([h_{t-1}, X_t] \cdot W_f + b_f\right) \tag{3.32}$$

  The forget gate determines how much of the previous cell state c_{t-1} should be forgotten based on the current input X_t and previous hidden state h_{t-1}.

- Input gate:

$$i_t = \sigma\left([h_{t-1}, X_t] \cdot W_i + b_i\right) \tag{3.33}$$

$$\widetilde{c}_t = \tanh\left([h_{t-1}, X_t] \cdot W_c + b_c\right) \tag{3.34}$$

  where $\widetilde{c}$ represents the new candidate cell state.

- Update cell state:

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \widetilde{c}_t \tag{3.35}$$

- Output gate:

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, X_t] + b_o\right) \tag{3.36}$$

$$h_t = o_t \cdot \tanh\left(c_t\right) \tag{3.37}$$

Where σ indicates the sigmoid activation function, W_f, W_i, W_c, W_o signifies weight matrices corresponding to the forget gate, input gate, candidate cell state, and output gate, respectively. b_f, b_i, b_c, b_o are bias vectors.

Fully Connected Layer

Fully connected layers process the output from the LSTM layer, performing feature transformation and abstraction. They learn complex relationships between features and prepare them for the final output layer.

$$FC(X) = \sigma\left(W_{FC}X + b_{FC}\right) \tag{3.38}$$

where W_{FC} is the weight matrix for the fully connected layer, b_{FC} is the bias vector, σ represents the activation function.

Output Layer with Softmax Function

The Softmax function is employed as an activation function in the output layer, which was chosen for its outstanding performance as a classifier. The output layer provides the classification results regarding the presence of intrusions based on the input data and the model's learned representations.

$$\text{Output}(X) = \textit{softmax}\left(W_{\text{out}}\, X + b_{\text{out}}\right) \tag{3.39}$$

This proposed hybrid approach uses the processing abilities of CNNs for extraction of spatial features, SE-ResNeXt for attention-based enhancement, and LSTM networks for modeling time integration. Therefore, the model could appropriately capture the two dimensional and one dimensional natures of network traffic data, thus being potentially useful for detecting intrusion in SDN environments. Suppose as the attacks are happening, the DQN agent (SDN Controller) would react in time. This as a result of Step 4 (trained model) will classify the incoming flows in real time with labels (such as; normal flow, attack, or malicious traffic). If an attack is detected, the DQN agent will take the action required as explained in Section 3.3.5.

3.3.5 Attack Mitigation

In this section, we propose a Deep Q-Network (DQN) to mitigate the attacks. Deep Q-Network (DQN) [58] combines Q-Learning and a reinforcement learning algorithm, with deep neural network to handle large and complex state spaces. The DQN agent acquires decision-making skills through its interaction with the network environment. It observes the current state, takes actions, and receives rewards based on the consequences of those actions.

State: Packet rate, flow duration, protocol usage, as well as source and destination IP addresses, are all representations of the current state of the network and its activity at those levels.

$$S_t = [TV_t, FR_t, AS_t, PSD_t] \tag{3.40}$$

where TV_t denotes traffic volume, FR_t denotes flow rate, AS_t, represents anomaly score, and PSD_t signifies packet size distribution.

Action: The DQN agent's potential actions related to traffic management are to apply rate limiting to specific traffic flows A_1, prioritizing traffic for certain applications A_2, blocking suspicious IP addresses A_3, redirecting traffic to honeypots A_4, and dynamically adjusting firewall rules A_4.

Reward Function: Positive reward is given for reduced malicious traffic detection and negative reward for increased packet loss or latency.

Training: The DQN agent is trained using a deep neural network to approximate the Q-value function, which estimates the expected reward for taking an action in a given state. The network is trained using experience replay, where past experiences (state, action, reward, next state) are stored and sampled randomly to break the correlation between consecutive experiences.

The loss function for training the DQN is based on the temporal difference (TD) error.

$$L(\theta) = \mathrm{E}\left[\left(r + \gamma \max Q\left(s', a'; \theta^{-}\right) - Q(s, a; \theta)\right)^2\right] \tag{3.41}$$

Where θ are the parameters of the Q-network, θ^{-} are the parameters of the target network, s and s' are the current and next states, γ denotes the discount factor, s and a' are actions.

After training, the decision-making process of the SDN controller is taken over by a DQN agent. It interacts with the network constantly as it keeps updating its policy according to new data and change in threats. DQN agents react in time when there

are modifications in traffic patterns or attacks, and automatically adjust SDN flow rules to isolate threats.

3.4 Experimental Results

In this section, results from our various experiments are discussed. We set up the experimental environment using Python to implement, evaluate, and compare the proposed methodology with three other existing methods. We tried to find dataset that contains the record of the real-time traffic that has been captured on a daily basis. For that we chose a dataset for SDN intrusion detection which is available on Kaggle.com, and contains the records of DDoS, XSS Intrusion, Brute Force Intrusion, SQL Injection and Bening traffic [59]. Then, we implemented the steps mentioned on the methodology section.

We randomly divided the dataset into training and testing sets as 70/30 and 80/20. This will create random sampling, which involves randomly selecting a subset of data for the training set. Also, we implemented a Cross-Validation to provide a more reliable estimate of model performance.

For performance evaluation, we used precision, accuracy, sensitivity, specification, F-Measure, FPR and FNR. The results of the experiment are shown below in different graphs and tables.

The histogram of features is provided in Figure 3.4. which is helpful in understanding the distribution of features. The normlized output is shown in Figure 3.5.

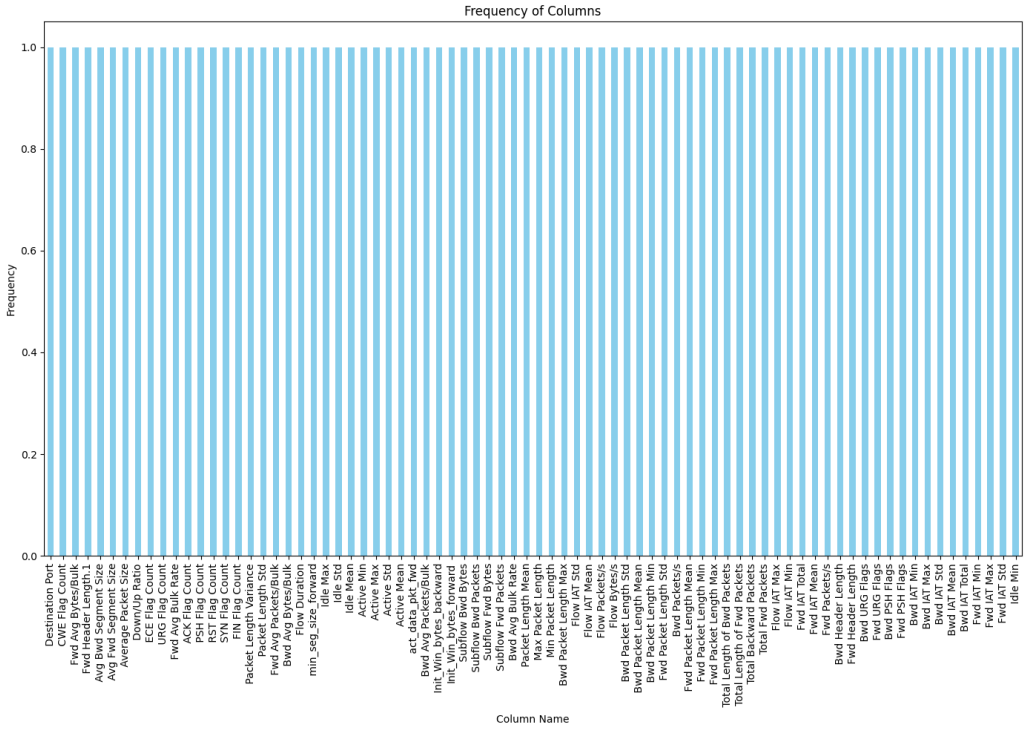


Figure 3.4: Histogram of features

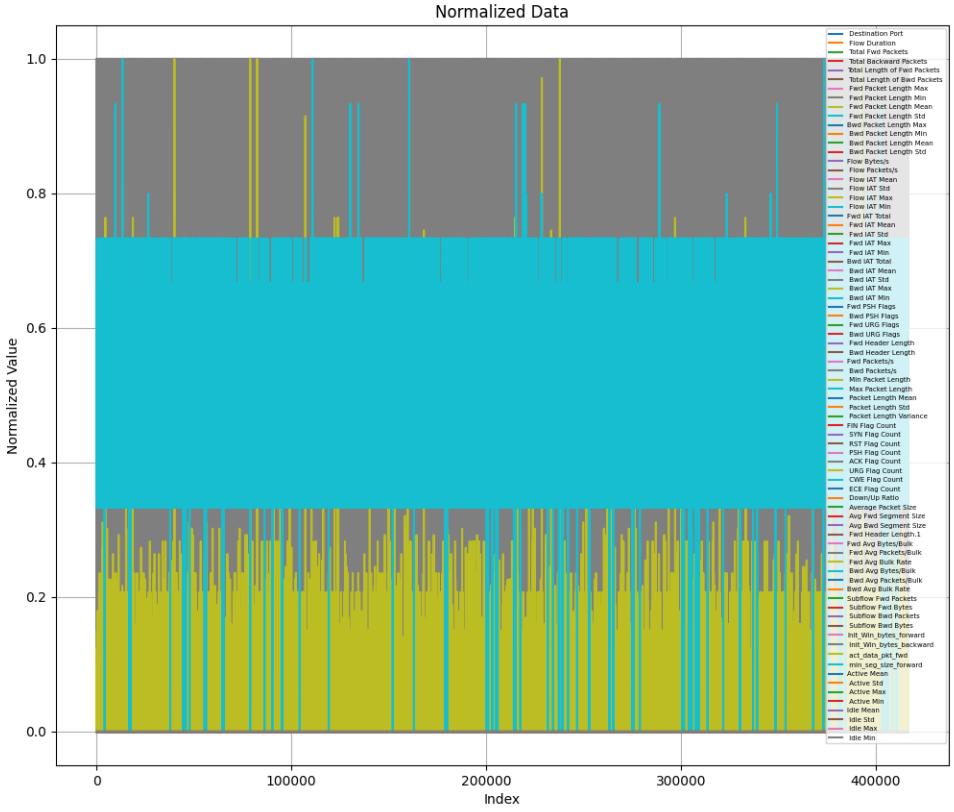


Figure 3.5: Normalized Output

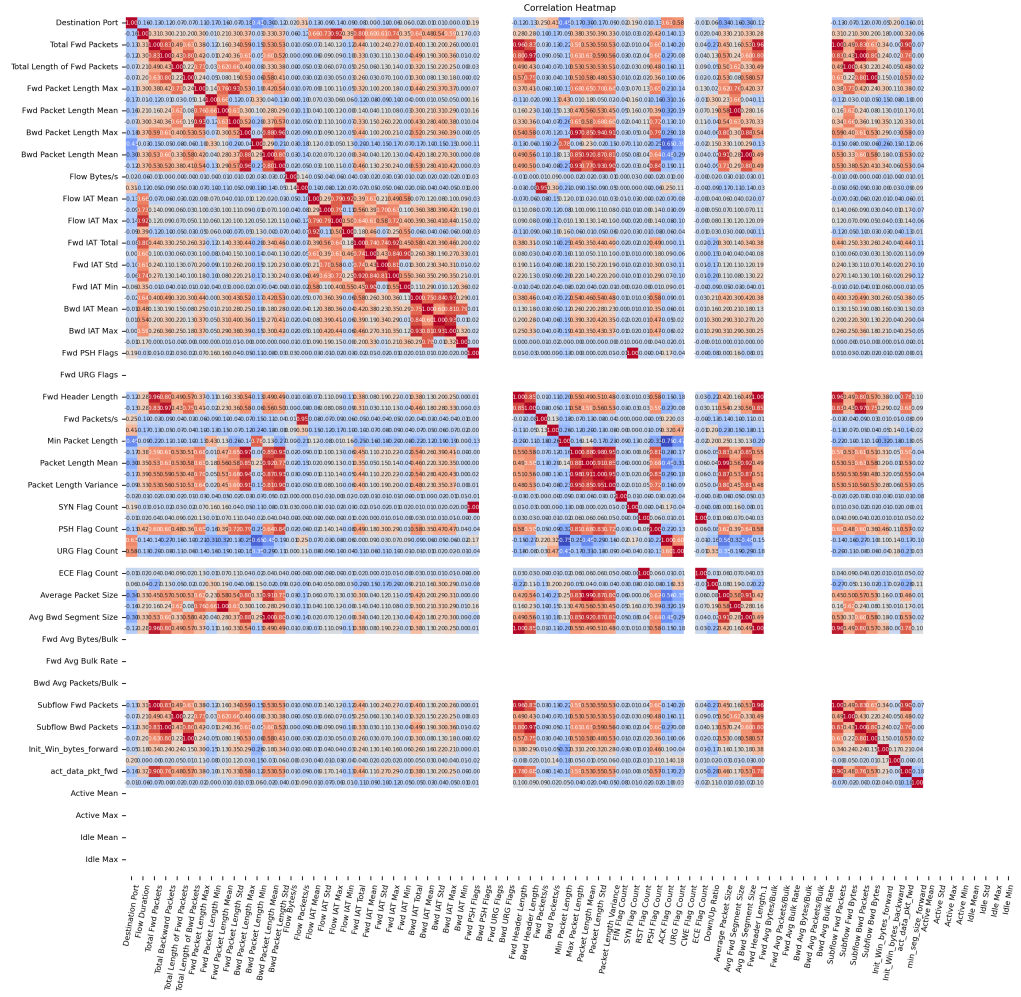


Figure 3.6: Correlation Heatmap

Figure 3.6 illustrates the correlation heatmap where darker colors indicate a stronger correlation matrix between multiple network flow features contained in the dataset for more advanced analysis. The intensity of colour spectrum runs from deep blue for negative correlations through deep red for positive correlations.

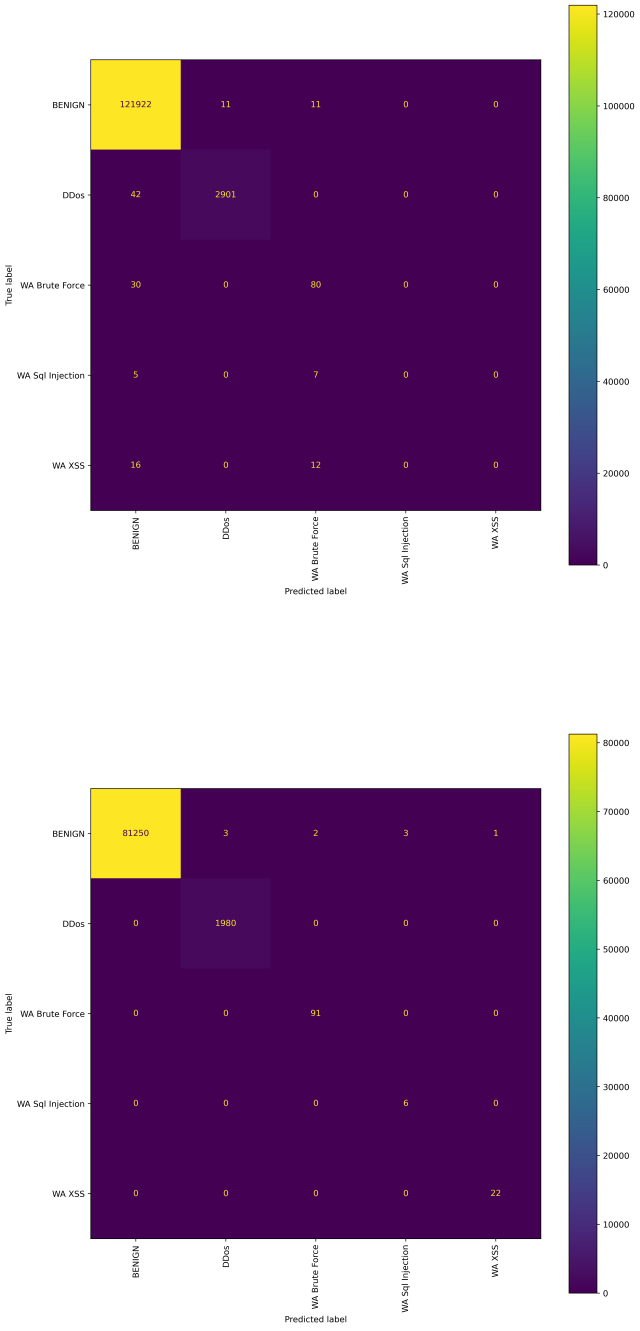


Figure 3.7: Confusion Matrix for 70% training data and 80% training data

Figure 3.7 shows the confusion matrix which is a performance evaluation tool in Machine Learning that represents the prediction summary in matrix form and indi-

cates how accurately the model performs. It provides a comprehensive breakdown of correct and incorrect predictions, facilitating a deeper understanding of an algorithm's performance. The results indicate a better and higher accuracy (less error) with 80% training data.

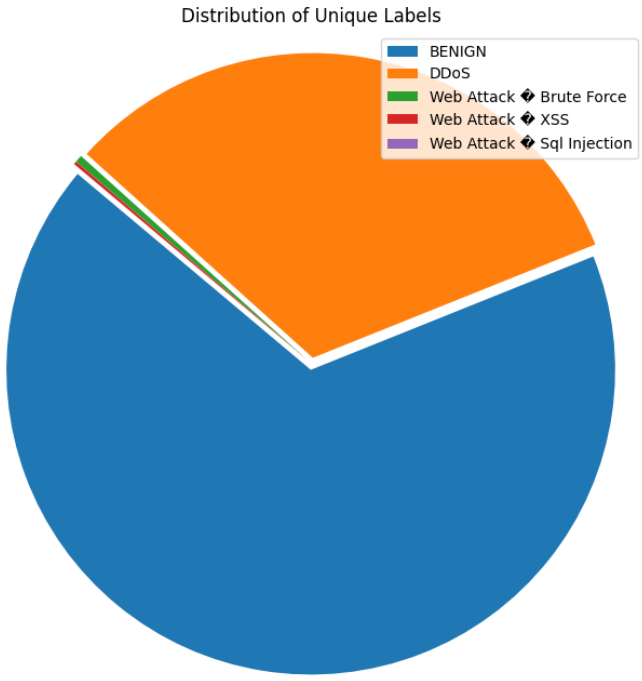


Figure 3.8: Distribution of Labels

Figure 3.8 shows the distribution of various attacks such as DDoS, Brute force, XSS attack etc.

In Figure 3.9, for 70% training data, the proposed model has obtained high specificity score of 0.999732. This is higher than GRU, which scores 0.960347. Additionally, it is better than CNN with a specificity of 0.976319, SVM at 0.990486, and LSTM at 0.947431. For 80% training data, the proposed model has a specificity of 0.999973. This shows its enhancement over GRU, which has a specificity of 0.970635. It also exceeds CNN's score of 0.984841, SVM's 0.992143, and LSTM's 0.951508.

Figure 3.9 gives the sensitivity analysis, for the proposed and the existing methods. For 70% training data, the proposed model has a sensitivity of 0.998928. On the

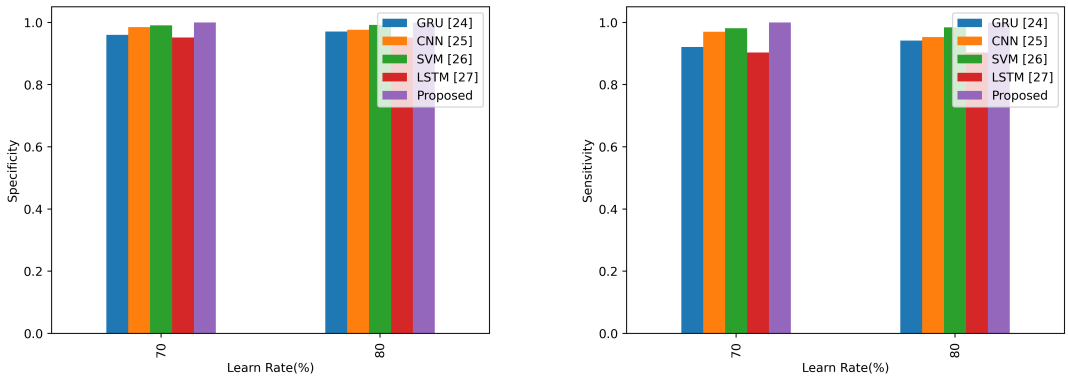


Figure 3.9: Specificity and Sensitivity analysis

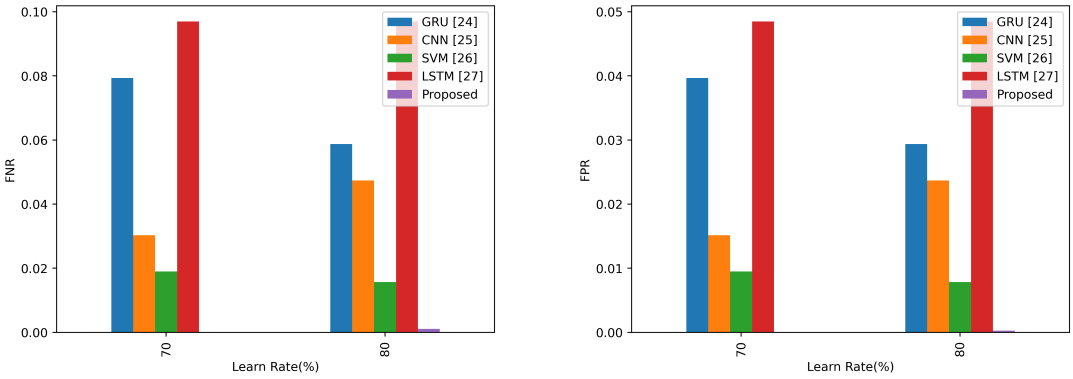


Figure 3.10: FPR and FNR analysis

other hand, GRU's sensitivity is 0.920694, CNN with 0.952639, SVM with 0.980972, and LSTM with 0.894861. This indicates that the proposed model is able to detect malware and reduce false negatives. Similarly, for 80% training data, the proposed has a sensitivity of 0.999892, which is higher than GRU's sensitivity of 0.94127. The sensitivity of the CNN is 0.969683, SVM is 0.984286, and LSTM is 0.903016. The high sensitivity scores reveal the reliability of the proposed method across different training data proportions for identifying the attacks accurately.

Figure 3.10 gives the False Positive Rate (FPR) and False Negative Rate (FNR) analysis. Our proposed model has the lowest FPR of 0.000268. This is much less compared to GRU of 0.039653, CNN of 0.023681, SVM of 0.009514, and LSTM of 0.052569. Additionally, our FNR remains below at 0.001072 compared to GRU (0.079306), CNN (0.047361), SVM (0.019028), and LSTM (0.105139). With 80%

training data, the proposed model has an even lower FPR of 0.000027 compared to GRU (0.029365), CNN (0.015159), SVM (0.007857), and LSTM (0.048492). The FNR of the proposed model is also low at 0.000108.

A comparative analysis with different matrices and methods are illustrated in Tables 1 and 2.

Table 3.1: COMPARATIVE ANALYSIS WITH 70% TRAINING DATA

Metrics/Method	**GRU[24]**	**CNN[25]**	**SVM[26]**	**LSTM[27]**	**Proposed**
Accuracy	0.94713	0.968426	0.987315	0.929907	0.999571
Precision	0.920694	0.952639	0.980972	0.894861	0.997698
Sensitivity	0.920694	0.952639	0.980972	0.894861	0.998928
Specificity	0.960347	0.976319	0.990486	0.947431	0.999732
F-Measure	0.920694	0.952639	0.980972	0.894861	0.998928
MCC	0.881042	0.928958	0.971458	0.842292	0.998501
NPV	0.960347	0.976319	0.990486	0.947431	0.999257
FPR	0.039653	0.023681	0.009514	0.052569	0.000268
FNR	0.079306	0.047361	0.019028	0.105139	0.001072

Table 3.2: COMPARATIVE ANALYSIS WITH 80% TRAINING DATA

Metrics/Method	**GRU [24]**	**CNN [25]**	**SVM [26]**	**LSTM [27]**	**Proposed**
Accuracy	0.960847	0.979788	0.989524	0.935344	1
Precision	0.94127	0.969683	0.984286	0.903016	0.9999
Sensitivity	0.94127	0.969683	0.984286	0.903016	0.999892
Specificity	0.970635	0.984841	0.992143	0.951508	0.999973
F-Measure	0.94127	0.969683	0.984286	0.903016	0.999892
MCC	0.911905	0.954524	0.976429	0.854524	0.999852
NPV	0.970635	0.984841	0.992143	0.951508	0.999991
FPR	0.029365	0.015159	0.007857	0.048492	2.7E-05
FNR	0.05873	0.030317	0.015714	0.096984	0.000108

In Figure 3.11, the Receiver Operating Characteristic (ROC) analysis is shown.

This analysis measures how well the proposed method discriminating an object using false positive and true positive rates against various threshold values. A larger area under ROC curve implies greater division capacity of this model.

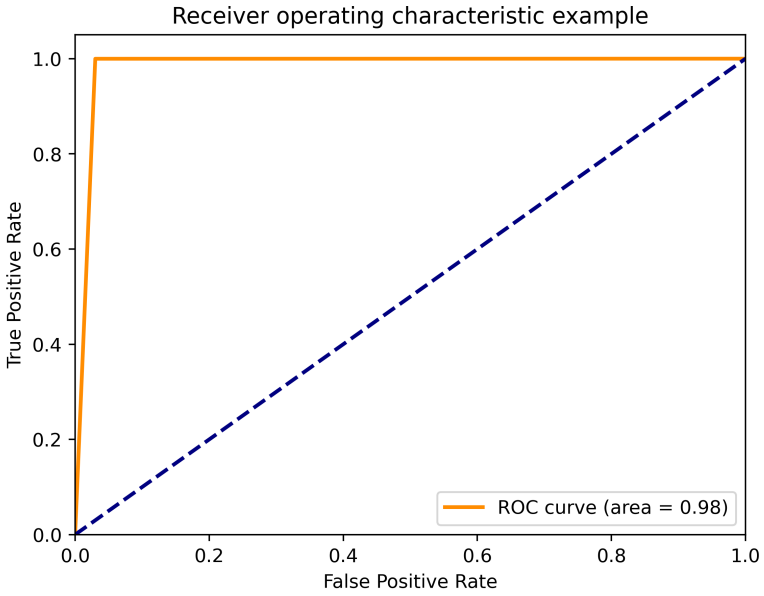


Figure 3.11: ROC analysis

Figure 3.12 presents the Density plot analysis for different training data proportions. Subfigure (a) illustrates the density plot analysis for the 70% training data, while subfigure (b) depicts the same analysis for the 80% training data.

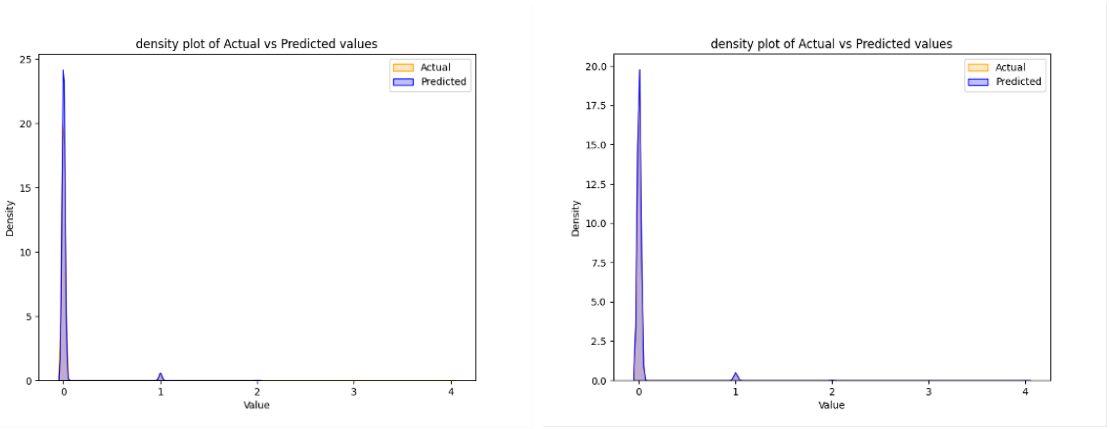


Figure 3.12: Density plot analysis (a) For 70% training data (b) For 80% training data

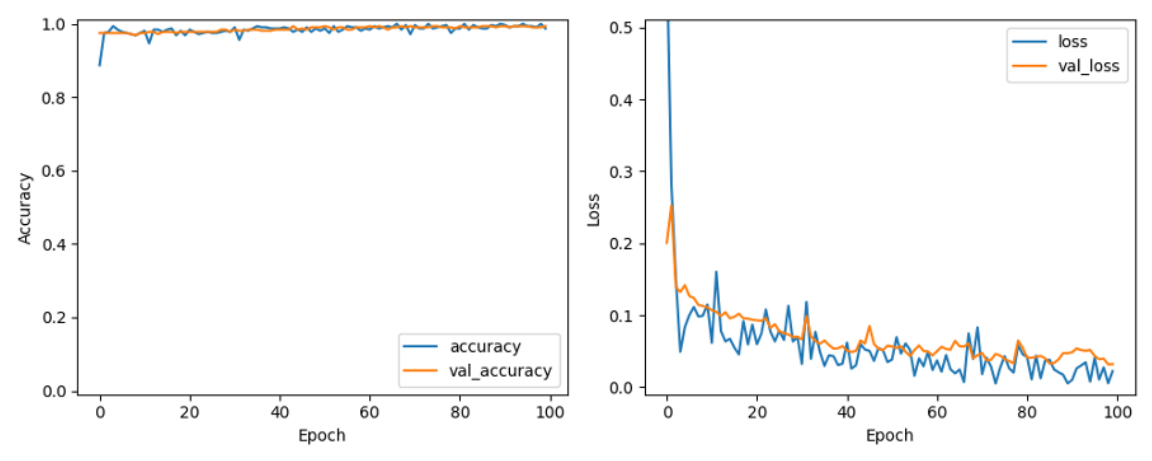


Figure 3.13: Accuracy and Loss analysis 70% training data

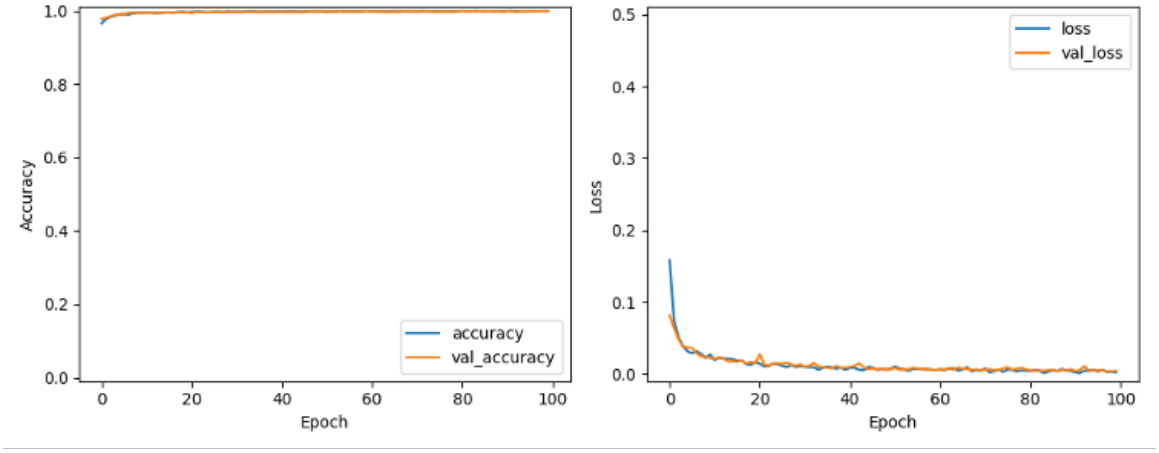


Figure 3.14: Accuracy and Loss analysis 80% training data

In Figure 3.13 and Figure 3.14, the proposed model is evaluated based on two performance indicators, namely Accuracy and Loss. The analysis is undertaken separately for training data sizes of 70% (Fig. 3.13) and 80% (Fig. 3.14). These curves track the learning time error rate as well as classification accuracy at each iteration.

3.5 Conclusions

This chapter presented a Dynamic Defense approach to enhance intrusion detection and mitigation capabilities in SDN environments. By using data pre-processing, feature extraction, and selection, coupled with innovative algorithms like the LOA and

CBO algorithms, the optimal feature subsets are identified. This helped to improve the accuracy of intrusion detection systems. The integration of CNNs, SE-ResNeXt, and LSTM architecture further improved the ability to detect and classify complex cyber threats. Additionally, DQN based dynamic attack mitigation strategy was employed to minimize the impact of attacks and take necessary actions. Future studies should make the optimization better, expand the experiments to include other intrusion datasets, and take some other add-ons in order to make the security of the network tighter to resist the constantly changing cyber threats.

Chapter 4

Federated Intelligent Intrusion Detection and Mitigation Framework for SD-IoT Networks using ViT-GraphSAGE and Automated Attack Reporting

[1]

4.1 Introduction

Many domains such as smart cities, industrial automation, healthcare and intelligent transportation systems have seen advanced transformations due to the proliferation of Internet-of-Things (IoT), that enables unified and continuous interconnectivity of devices and data-driven automation. To deploy IoT on a grand scale proves exceedingly difficult because it needs proper network administration together with resource management and security protocols. In order to resolve these complications, Software-Defined IoT (SD-IoT) emerged as a promising solution that leverages the manipulation of Software-Defined Networking (SDN) for IoT infrastructure to bring in a centralized, flexible and programmable network architecture [60, 62]. SD-IoT splits

[1]This chapter is part of the work accepted in the NTMS'25: 12th IFIP International Conference on New Technologies, Mobility and Security, 2025. [136]

the control plane functions from data plane operations which delivers instant network adjustments, security monitoring capabilities and traffic response control. Network administrators gain control through this split structure because controllers can implement better policies for anomaly detection and threat response with enhanced efficiency. The continued IoT device connectivity creates serious security problems which requires the creation of advanced intrusion detection and mitigation solutions to protect SD-IoT networks from new cyber threats. Central control systems in SD-IoT networks, being such major targets for cyber threats like DDoS attacks, spoofing, and unauthorized access, require safeguards and countermeasures to mitigate those threats [44-46]. IoT devices with limited resources are subject to hackers who can take over the devices and use them as part of botnets to launch big attacks, preventing services on networks from running and causing major operational breakdowns. This is further complicated due to a large number of IoT devices, and it is difficult to detect harmful activities. In addition, IoT devices generate an enormous amount of data, which is time sensitive, thus complicating the detection and handling of threats [16].

The common practice of security approaches such as firewalls and rule based anomaly detection are incapable of securing the SD-IoT networks due to their static nature and lack of ability to respond to newly emerging attack patterns. Various intrusion detection and mitigation methods ranging from traditional signature-based detection to advanced machine learning (ML) and deep learning models are being studied and developed to counter these security threats. However, conventional methods of intrusion detection systems (IDSs), based on pre-defined attack signatures, are infeasible for zero day attacks as they are not adaptable. In response, there has been development of ML based models to analyse the network traffic patterns and find anomalies with better detection capability. In more recent times, Graph Neural Networks (GNNs) and Transformer models have achieved good performance for intrusion detection by exhibiting the capability to identify both local and global dependencies in network traffic. Additionally, the security threats are dynamically contained by considering these various attack mitigation approaches including SDN-based traffic rerouting, automated response mechanisms, and reinforcement learning based decision making. SD-IoT networks face many challenges with current intrusion detection techniques. The typical deployment of Machine Learning (ML) and Deep Learning (DL) models leads to excessive identification of false threats that produces incorrect threat assessment and unnecessary actions to investigate these potential threats [24].

A major drawback of deep learning intrusion detection is its heavy requirement for processing power because small IoT devices often lack sufficient computational resources [25-26]. The existing security issues demand a combined security system that can grow and identify threats in real-time. A security system must combine federated learning (FL) with spread-out security, deep learning for adapting to new threats and reinforcement learning as a live decision maker to stop attacks quickly. By decentralizing the learning process, FL offers a robust, scalable, and privacy-preserving approach to securing sensitive data against evolving cyber threats. The main contributions of this chapter are as follows;

- Hybrid ViT-GraphSAGE: Intrusion Detection is introduced that combines Vision Transformer for global traffic analysis and GraphSAGE for local node interactions, resulting in improved threat detection accuracy.
- Multi-Agent DQL-Based Attack Mitigation: This enables real-time response by dynamically rerouting traffic and isolating compromised nodes thereby preventing large-scale disruptions.
- Federated Learning and Automated Attack Reporting: this proposed work uses FL for privacy-preserving intrusion detection and Flan-T5 Transformer for generating detailed forensic attack reports.

This chapter continues with the following organization: Section 4.2 presents related work; Section 4.3 explains the technical aspects and the components of the proposed approach; Section 4.4 discusses the implementation and performance results of the proposed approach and finally, Section 4.5 concludes the chapter.

4.2 Related Work

Jehad Ali et al [61] presented a two-step method for TinyML algorithm selection through the implementation of Hybrid Analytical Network Process (HANP). The preliminary phase included examining different machine learning algorithms through qualitative comparison to determine which ones fit TinyML requirements in SD-IoT devices. A performance analysis followed to evaluate chosen machine learning algorithms before using validation methods to show the effectiveness of proposed model selection approaches for TinyML. The main drawback of this investigation was its

failure to examine real-time intrusion detection plus mitigation procedures for SD-IoT networks. Though TinyML algorithm enhanced the performance across various applications, it failed to solve security risks which emerge as cyber threats continue to change dynamically.

The two-step intrusion detection method for SD-IoT networks developed by Tian et al. [62] contains features intended for attack detection enhancement. The authors added differential evolution mutation algorithm to firefly algorithm to overcome its limitations with complex applications while increasing its speed and improving avoidance of local optimum traps. The enhancement produced better accuracy as well as efficiency in selecting features. The most essential features were obtained through a wrapper-based feature selection process prior to their evaluation through a unique ensemble classification system. An ensemble model used the C4.5 decision tree, multilayer perceptron and instance-based learning which provided a diverse robust classification approach. The system used weighted voting to establish abnormal behavior in network traffic which resulted in better intrusion detection outcomes. The method lacked preventive countermeasures to prevent security violations in SD-IoT systems because no attack mitigation solutions were implemented.

Sayed et al. [63] designed MP-GUARD as a new framework which uses Software-Defined Networking and machine learning together with a multi-controller architecture to boost IoT security features. The framework operates as three parallel components to detect intrusion in real time and monitor combined traffic with multiple defenses against attack scenarios. The MP-GUARD framework included P4-Assisted Cooperative Traffic Monitoring (CTM-P4) as its initial module along with Multi-Pronged Intrusion Detection and Mitigation (MPIDM) as its secondary component. Through the CTM-P4 module, multiple SDN controllers could establish real-time communications that enabled P4-enabled switch state tables to interact dynamically for feature extraction. SDN controllers cooperated through this method to share data which increased efficient traffic monitoring and abnormality detection capabilities. MP-GUARD suffered from two main drawbacks since it depended on traffic analysis through rules and static Machine Learning-based detectors that proved ineffective against changing cyber threat patterns.

Shaji et al [64] proposed SD-IIDS as an Intelligent Intrusion Detection System for Software-Defined Networks that specialized in Distributed Denial-of-Service attack detection in SDN networks. The system developed two ensemble ML classification models that could operate either as binary or multi-class classifiers. SVC served

as the Support Vector Classifier component in the first ensemble model linked with SVC-RF through bagged Random Forest while the second model joined RF with Logistic Regression as RF-LR. The developed models served to boost classification accuracy while making the SDN network intrusion detection process more reliable within dynamic environments. However, it resulted in a reactive security approach as this framework did not incorporate any proactive attack mitigation strategies.

Prabhat et al. [65] developed DLTHF as a Threat Hunting Framework with Deep Learning capabilities which detects isolated and multiple attack patterns to improve SD-IoT security. The framework included an automatic features extraction module which used data perturbation encoding and normalization scaling to prepare network traffic information. A Long Short-Term Memory Contractive Sparse AutoEncoder (LSTMCSAE) method transformed data values into protected format while creating robust feature representation. The Threat Detection System (TDS) incorporated MhSaBiGRNN which is a Multi-head Self-attention-based Bidirectional Recurrent Neural Networks to detect cyber threats alongside their attack types effectively. The proposed threat detection system evaluated network traffic instances with self-learned weight values to measure their relevance levels thus enhancing intrusion detection accuracy.

The research team led by Mahmoud introduced XI2S-IDS [32] which served as an Explainable Intelligent 2-Stage Intrusion Detection System that addressed detection accuracy and interpretation for low-frequency attacks on SD-IoT networks. This framework executed a dual-step procedure starting with binary separation between regular and damaging network movements and then performing multi-class categorization for specific attack types. The SHAP (SHapley Additive exPlanations) values system revealed the most crucial features for attack detection to security analysts making decisions more transparent. The explainability features delivered better prediction understanding to analysts who wanted to trust automated intrusion detection and take more informed security decisions. However, this method mainly aimed to boost interpretability without optimizing attack prevention which resulted in a deficiency regarding reactive security action capabilities.

Research by George et al. [66] developed an IDS for IoT devices through the implementation of unsupervised and supervised deep learning models that underwent training using Federated Learning. The researchers compared FL-trained models against traditional models which did not use FL by implementing the N-BaIoT dataset containing nine IoT device network traffic data. A randomized search hyperparam-

eter optimization technique improved detection accuracy through the adjustment of important system parameters. However, the FL introduces computational overhead difficulties mainly because of the regular data exchange activities between edge nodes and the central server which results in latency problems in resource-restricted IoT environments.

Generative Adversarial Networks (GANs) enabled Khatami et al. [67] to carry out anomaly detection for IoT systems. Researchers improved optimization through the implementation of five-dimensional Gray Wolf Optimizer (5DGWO) as a hyper-parameter tuning system. The 5DGWO framework used gamma wolves along with theta wolves to both enhance exploration and exploitation while reducing chances of local minimum trapping during the optimization process. The proposed system architecture contains four distinct stages starting from preprocessing and proceeding to generative model training and auto-encoder (AE) training and finishing with predictive model training which enables an efficient anomaly detection system. The experimental tests proved that the optimized GAN framework successfully identified anomalies within Internet-of-Things systems. Logically the GAN training requires extensive computational resources while requiring repeated iterations from 5DGWO which limited its feasibility for IoT device deployment.

Toony et al. [68] developed MULTI-BLOCK as a multi-module framework to boost IoT network management based on machine learning methods and P4 stateful processing along with an SDN multi-controller architecture. The designed framework serves multiple important security goals including synchronized communication alongside traffic monitoring capabilities and intrusion detection together with attack mitigation functions. The primary elements of MULTI-BLOCK formed a four-part system. The installment of MULTI-BLOCK established the Pyramidal Conceptually Decentralized Multi-Controller Structure (PCDMCS) with Decentralized Control Interfaces (DCIs) providing the Decentralized Warning Conduit (DWC) for real-time threat detection. However, the implementation of multiple controllers for threat mitigation created delay problems that could affect urgent response to security threats.

The research team of Kong et al [69] developed iDetector as a real-time intrusion detection system for IoT networks which combines simplicity in structure along with easy replication features. The framework used a moving sampling window technique to observe network dialogs in real time while producing traffic samples which joined various network features. Using the obtained samples researchers could easily represent different traffic behavioral patterns. By considering classification performance

enhancement, the Nonlinear Feature Transformation (NFT) algorithm transformed prior traffic feature distributions for increasing information entropy in sample data. The developers built EdgeNet as a lightweight deep neural network design which could execute on edge gateways. However, the system faced challenges because of its static feature transformation process to identify new attack forms dynamically, thus achieving reduced accuracy in detecting modern cyber threats.

4.3 Motivation and The Proposed Methodology

The Software-Defined IoT (SD-IoT) networks are creating significant new problems for intrusion detection system due to their substantial growth of dynamic, resource-constrained and heterogeneous networks in nature. The existing intrusion detection systems frequently fail to adjust to the evolving cyber threats in SD-IoT environments, primarily due to two key limitations: *1)* The ineffective feature extraction and selection in high-dimensional, noisy network traffic data, and *2)* Inadequate attack mitigation strategies that are failing to predict the evolution of sophisticated cyber-attacks. Generally, some crucial things, such as handling diverse and multimodal network traffic, including time-series anomalies and noise originating from heterogeneous IoT devices are the main requirements of an intrusion detection in SD-IoT networks. Hence, the conventional anomaly detection methods struggle to capture both local interaction of nodes and global traffic patterns in the networks as they are generally based on statistical modeling leading to poor generalization and high false positive rates.

In our proposed model, the hybrid Vision Transformer (ViT) and GraphSAGE are utilized that address this challenge by integrating global context-aware analysis with local node-wise intrusion detection. In addition to this, feature selection is presented as another fundamental process in ensuring real-time and efficient intrusion detection. However, some redundant and irrelevant network traffic features in the IoT data lead to increased computational overhead and degraded performance of the model. In order to overcome this issue, in the proposed approach, a Sparrow Search Optimized CapsNet is introduced in which the dynamic routing capabilities of a capsule neural network (CapsNet) is utilized to capture hierarchical feature relationships while using swarm intelligence-based optimization for selecting the most discriminative traffic features.

Along with the intrusion detection system, an effective attack mitigation frame-

work is necessary for the complete protection of the IoT system data. However, the commonly used existing attack mitigation frameworks in SD-IoT networks generally exhibit latency even when they respond to sophisticated attacks like the network spoofing and Distributed Denial of Service (DDoS) as they rely on reactive rule-based strategies. Besides, cascading failures are being noticed in critical IoT infrastructures because of the lack of predictive mitigation models that further aggravate the vulnerability of SD-IoT networks. Hence, in order to bridge this gap, this research work proposes a Multi-Agent Deep Q-Learning (MA-DQL) framework in which each SDN controller acts as an autonomous agent to make intelligent mitigation decisions such as device isolation and traffic rerouting. In addition, to enhance the intrusion detection, the existing approaches use Reinforcement-Learning based predictive mitigation strategies. However, they make the SD-IoT system ineffective against stealthy and evolving network attacks as they lack proactive mitigation capabilities. Therefore, to address this, a Temporal Convolutional Network (TCN)-based Predictive Mitigation Module is integrated into the proposed model to address these issues by forecasting attack progression and taking preventive actions thereby minimizing network disruptions.

The overall structure of the proposed model is depicted in Figure 4.1. Furthermore, in this proposed model, Federated Learning (FL) is incorporated to enhance collaborative security intelligence. This integrated FL allows multiple edge nodes to participate in distributed intrusion detection without compromising data privacy. The existing centralized IDS approaches always suffer from single points of failure and excessive communication overhead, while FL leverages decentralized training to enhance model generalization across diverse attack patterns. Finally, not much research has been done on post-mitigation forensic analysis hence it still remains a neglected area in SD-IoT security. Focusing solely on the available logging mechanisms, they do not have the relevant infrastructure to generate machine-automated structured attack reports that are critical for the sharing of intelligence. Hence, a Flan-T5-based Transformer-driven Attack Reporting System is embedded in this proposed methodology supporting the generation of automated documentation of attack patterns, affected nodes, and the mitigation actions that enables post-incident analysis and ensures reinforcement of security throughout the entire framework.

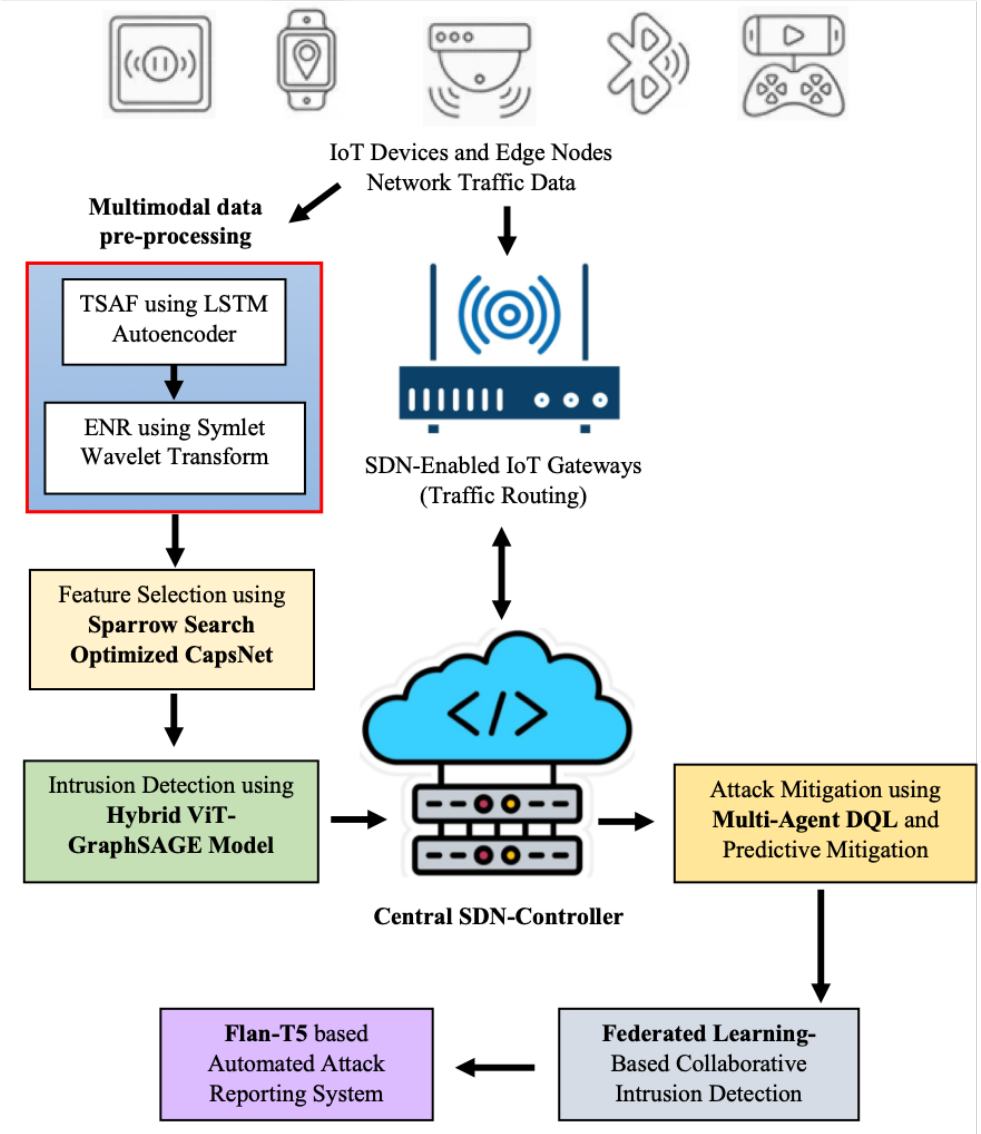


Figure 4.1: Overall architecture of the proposed method

Each component of the proposed innovative framework is explained in detail in the upcoming sections one by one with their technical significance.

4.3.1 Multimodal Data Pre-Processing

Due to the heterogeneity of various devices used in the IoT network, the original traffic data is often noisy, irregular, and contains missing values. Hence, in the pre-processing stage, the following two key techniques are employed to ensure high-quality

data: Time-series anomaly filtering (TSAF) and Edge Noise Reduction (ENR) which are explained below.

Time-Series Anomaly Filtering using LSTM Autoencoder

The first step of preprocessing involves detecting and correcting anomalies in time-series network traffic data using a Long Short-Term Memory (LSTM) Autoencoder which is employed to learn a compact representation of normal traffic patterns and identify deviations as anomalies. The LSTM encoder [70] initially compresses the sequential input traffic data $X = \{x_1, x_2, \ldots, x_T\}$ into a latent representation, while the decoder then reconstructs the input from this compressed encoding. The reconstruction error between the original and reconstructed sequence is computed using the Mean Squared Error (MSE) loss function as given by equation (4.1);

$$L_{TSAF} = \frac{1}{T} \sum_{t=1}^{T} \|x_t - \hat{x}_t\|^2 \tag{4.1}$$

where, the actual traffic data is represented by x_t, reconstructed value is represented by $\hat{x}_t$ and the total number of time steps are denoted by T. In this $TSAF$, a predefined anomaly threshold δ is used to classify anomalous traffic points thereby ensuring that outliers in network behaviour are effectively detected and corrections are applied before further processing. The input traffic data after being corrected for time series inconsistencies at this stage are represented as X'.

Edge Noise Reduction using Symlet Wavelet Transform

The accuracy of intrusion detection models would get degraded due to the environmental factors and device fluctuations, causing the IoT network traffic data to be affected by high-frequency noise, leading to false alarms, adversarial attacks, and data corruption. To mitigate this, Symlet Wavelet Transform (SWT) is employed in the second stage of pre-processing to effectively isolate noise from meaningful traffic patterns by decomposing the signal into approximation and detail coefficients. The wavelet decomposition applied on this given the traffic signal X' is mathematically represented by equation (4.2);

$$X' = A_k + D_k \tag{4.2}$$

Here, A_k signifies the approximation coefficients, capturing the low-frequency components, and D_k denotes the detail coefficients at level k, containing high-frequency

noise components. A thresholding mechanism is applied to D_k to suppress irrelevant fluctuations, ensuring that only the significant traffic features are retained for further analysis. This process enhances the robustness of intrusion detection by providing a cleaner and more reliable traffic data representation X_f as the input data for the upcoming crucial processes.

4.3.2 Feature Selection using Sparrow Search Optimized CapsNet

In the proposed intrusion detection system, feature selection is presented as a basis for reducing computational overhead and enhancing the performance of the model. For this, the proposed approach is integrated with Capsule Networks (CapsNet) for hierarchical feature extraction [75] with Sparrow Search Optimization (SSO) for optimal feature selection. Here, the CapsNet effectively captures spatial relationships between traffic patterns, while SSO ensures that only the most informative features are retained thereby reducing the redundancy and improving classification accuracy.

Feature Extraction Using CapsNet

CapsNet is employed to extract hierarchical and spatially correlated traffic features from the pre-processed network data X_f, ensuring that even a little smaller attack patterns are effectively captured. In the Capsule layer, the number of traffic samples are denoted by N and the feature dimensions are represented by d. The feature vectors V_i produced by this capsule layer, for an input feature matrix $X_f \in R^{N \times d}$, is mathematically represented by equation (4.3);

$$V_i = squash(W_{ij} X_f + b_{ij}) \tag{4.3}$$

In this equation (4.3), the trainable transformation matrix is represented as W_{ij}, bias term is denoted by b_{ij}, and $squash(.)$ represents the nonlinear activation function which ensures that the magnitude of V_i lies between 0 and 1. This transformation maintains the orientation based relationships between the traffic features by ensuring that capsules encode spatial hierarchies. CapsNet, unlike CNNs, provides better retention of both spatial location and contextual information which makes it acceptable for learning and detecting sophisticated dependencies in network traffic. The extracted feature vectors from CapsNet are given as inputs to the SSO module for

the optimal feature selection.

Optimal Feature Selection using Sparrow Search Optimization

Since CapsNet produces an exhaustive set of features, it becomes vital to choose the best subset of features F_s to improve model performance while addressing the risk of overfitting. Sparrow Search Optimization uses sparrow seeking behaviour to perform the feature selection optimization process [72]. The goal is to find those features in F_s that maintain strong classification capabilities alongside minimal redundant information and overlap. The process of optimization unfolds through implementation of fitness functions represented by equation (4.3);

$$J\left(F_s\right) = \sum_{i=1}^{m} \frac{InfoGain\left(f_i\right)}{|F_s|} \tag{4.4}$$

Here, in this equation (4.3), the information gain of feature f_i in distinguishing attack patterns is represented as $InfoGain(f_i)$. The gains of information from features directly affect the contribution strength to the classification operations. The SSO controls the selection dynamics of features through an exploration-exploitation process which allows leader sparrows to investigate candidate subsets while follower sparrows make updated selections. A dynamic adjustment mechanism keeps only the most important traffic characteristics denoted by X_{fs}, while maintaining accurate model performance and efficient computations. Hence, this approach significantly improves the accuracy and computational efficiency of the intrusion detection model thereby ensuring that only the most relevant traffic features are retained.

4.3.3 Intrusion Detection using Hybrid ViT - GraphSAGE Model

Following the optimal feature selection using Sparrow Search Optimized CapsNet, the resultant refined traffic feature dataset moves through a deep analysis for intrusion detection. For this, a Hybrid ViT-GraphSAGE Model is proposed in which the ViT module distinguishes normal traffic patterns from potential attack behaviors by modelling the overall network behavior, while GraphSAGE will be capturing the localized deviations to generate node embeddings based on network topology. Then, this network traffic data is classified into Normal, DDoS and Spoofing attacks through

a Multi-Layer Perceptron (MLP) classifier that receives the fused embeddings from these two models as its input.

Vision Transformer for global traffic pattern recognition

The ViT recognizes the global network traffic patterns [71] by analysing spatial dependencies in the selected subset of pre-processed feature space X_{fs}. This input traffic matrix X_{fs} is partitioned into P_i fixed-size patches and then according to equation (4.5), it is transformed into learnable embeddings E_i as given below;

$$E_i = W_p Pi + E_{pe} \tag{4.5}$$

In equation (4.5), the learnable patch embedding matrix is represented as W_p, and the positional encodings that retain sequential dependencies across network traffic patches is denoted by E_{pe}. Long-distance dependencies within the network structure are detected by multi-head self-attention layers by processing these embeddings. The ViT model develops capabilities to distinguish normal traffic from abnormal network wide behaviors through its learning process while building an extensive representation of global attack behaviour. The main attack detection limitation of ViT requires incorporating GraphSAGE to achieve better localized node-specific context understanding.

GraphSAGE for Local Node Interaction Aggregation

The ViT system recognizes universal network relationships but GraphSAGE chooses to process IoT device interactions through neighborhood information processing steps. Consider an IoT network graph $G(V, E)$, where V represents the set of nodes or devices and E denotes communication links. Here, the GraphSAGE aggregation function iteratively updates the representation [41] of each node. The updated feature embedding of node v at layer $k+1$ is shown in equation (4.6);

$$h_v^{k+1} = \sigma(W_k \cdot AGG\left(\left\{h_u^k \mid u \in N\left(v\right)\right\}\right)) \tag{4.6}$$

where, $N(v)$ signifies the neighbouring nodes, and the function $AGG(.)$ represents the mean pooling function that aggregates data from the connected devices. Through GraphSAGE, the proposed model efficiently identifies how DDoS attacks and spoofing incidents are spread among neighboring nodes, by effectively capturing localized

attack propagation patterns of nearby nodes. The classified intrusion output represented as X_a, is obtained from an MLP classifier which receives the unified feature representation of ViT and GraphSAGE outputs. The combined method of ViT and GraphSAGE enables creation of a systematic intrusion detection infrastructure that integrates global traffic analysis from ViT with nearby network pattern detection from GraphSAGE to substantially improve detection capability in SD-IoT.

4.3.4 Attack Mitigation using Multi-Agent DQL and Predictive Mitigation

An effective mitigation strategy is indispensable to prevent further network compromises. Hence, in the proposed SD-IoT model, succeeding the detection of malicious activities using the Hybrid ViT-GraphSAGE model, a MA-DQL is employed for decentralized, real-time decision-making. Here, the autonomous SDN controller agent enables localized mitigation actions such as traffic rerouting, isolation of malicious nodes, and adaptive filtering of identified anomalous traffic. As the real-time mitigation alone is inadequate for sophisticated and evolving cyber threats in SD-IoT, a Temporal Convolutional Network (TCN) based predictive mitigation mechanism is also integrated, which forecasts the progression of detected attacks and pre-emptively applies countermeasures. This combined processing of reactive (real-time) and proactive (predictive) mitigation ensures that the impact of attack is minimized and the network integrity is preserved.

Multi-Agent DQL for Localized Mitigation

In the Multi-Agent Deep Q-Learning framework, each SDN controller is represented as an independent reinforcement learning agent, which adapts its security policies dynamically based on network conditions. The system state is represented as s which denotes the current network status, for example, detected threats, traffic anomalies, etc. Based on this status, the agent selects an action a to maximize a cumulative reward r_q. The governing decision-making process is regulated by the Q-value function which is shown in equation (4.7);

$$Q(s, a) = r_q + \mu \max Q(s', a') \tag{4.7}$$

where, the next network state and the optimal future action are represented by s'

and a' respectively. The discount factor balancing the immediate and future rewards is denoted by μ.

The optimal attack mitigation policies are learnt by each SDN agent through continuous interaction with the network environment and a distributed attack containment is enabled with the incorporation of the multi-agent architecture, which together prevent the single-point failures and reduce response latency.

Predictive mitigation using temporal convolutional network

A TCN is applied to further enhance the mitigation process by predicting the future attack states for proactive intervention before security breaches intensify. With a sequence of past attack indicators X_a such as traffic anomalies and node compromise rates, the TCN model learns to estimate the probability of future attack occurrences $P(Y_a)$, conditioned on the observed historical patterns. This is shown in equation (4.8);

$$P(Y_a|X_{a-1}, X_{a-2}, ..., X_0) = f_{TCN}(X_a) \tag{4.8}$$

where, f_{TCN} represents the dilated causal convolutional function of the TCN model, that captures long-term dependencies in traffic behaviour. In this process, the parallel processing of temporal data is achieved by TCN, unlike traditional recurrent models thereby ensuring efficient learning of complex attack orders. Once an attack is predicted, this combination of MA-DQL and TCN in the proposed approach triggers preventive mitigation actions, such as adaptive firewall rule updates or dynamic resource reallocation in order to minimize damage.

4.3.5 Federated Learning-Based Collaborative Intrusion Detection

For a strong and effective intrusion detection model that continuously adapts to the evolving cyber threats, succeeding the attacks mitigation stage is very much essential. Hence, to achieve this, a Federated Learning framework is integrated into this proposed approach ensuring collaborative model training across distributed edge nodes, which also preserves data privacy, integrity, and confidentiality. The edge nodes locally train the intrusion detection model using the attack detection data. Instead of raw network traffic data, the central SDN controller will receive only model parameters. Finally, the SDN controller uses the Federated Averaging (FedAvg)

algorithm [77] to aggregate model update from all edge nodes to have a globally optimized intrusion detection model which is mathematically expressed in equation (4.9);

$$M_{Global} = \sum_{i=1}^{n} \frac{w_i}{W} M_i \tag{4.9}$$

In this equation (4.9), the weight of each local model is represented as M_i, and the total weight sum across all edge nodes is denoted by W. Then, the updated global model is redistributed to the edge nodes which now are able to detect future intrusions without the need of data sharing over the centralization. The collaborative learning aspect of this scheme further ensures that the intrusion detection framework is scalable, decentralized, and the data is confidential.

4.3.6 Transformer-Based Automated Attack Reporting System

For future improvements in security, it is necessary to detect attacks, remedy them where possible, and also to observe during and after the attack for which the happenings occur, a structured reporting mechanism is essential for forensic analysis. In order to do this, a Flan-T5 Transformer [73] based Automated Attack Reporting System is deployed to aggregate detected attacks and score their impact, actionable conditions, and mitigation action performed. The transformer-based system is given an input dataset D_i containing traffic logs, attack classification, and security response details and makes a comprehensive textual report R_T based on this input dataset as shown in equation (4.10);

$$R_T = f_{T5}(D_i) \tag{4.10}$$

The fine-tuned Flan-T5 model is represented as f_{T5}, which has been trained for automated cybersecurity documentation. Reports generated by the system provide necessary details about the attack types including DDoS and spoofing attacks along with compromised devices and affected network segments as well as applied mitigation steps. The gathered reports get processed within an SDN forensic database that supports post-mitigation investigation in addition to regulatory compliance needs and security audits. The proposed procedure with automated attack reporting enables a proactive data-driven intrusion response which supports constant security updates in

Software Defined IoT systems.

4.4 Implementation and performance analysis

This section contains information about the dataset together with system specifications and implementation tools for this work. The section also presents performance evaluation results of the proposed method, while showing its effectiveness through complete comparison analysis. Some sample output (report) details of this model experiment are shown in Appendix.

4.4.1 Dataset Description and Tools Used

The proposed model is studied using two datasets, the SD-IoT Intrusion Detection Dataset [74] and the CSE-CIC-IDS2018 dataset [76]. SD-IoT dataset serves a purpose as a dedicated intrusion detection solution for software-defined IoT environments by providing normal and attack instance labels for network traffic data. Security models aimed at dynamic IoT environments need to be evaluated by using this dataset because it contains different attack types such as Distributed Denial-of-Service (DDoS), port scanning and botnet activities. CSE-CIC-IDS2018 dataset contains two categories of incoming traffic which are Benign and Malicious and supports detection of multiple attack types including DDoS, Botnet, Brute Force attacks and Infiltration attacks. These datasets contain numerous network flow features which include packet length along with flow duration and protocol-based attributes that help detect anomalous patterns. We used a three-way split into training, validation, and test sets. With a split ratio 60/20/20, where 60% is used for training, 20% for validation, and 20% for testing, which facilitates model learning, hyperparameter tuning, and unbiased performance evaluation respectively

Python provided the platform for implementing the proposed security framework of SD-IoT networks. PyTorch and TensorFlow were used to perform evaluations and training of the ViT-GraphSAGE intrusion detection model as well as the MA-DQL based attack mitigation system. The graph-based network analysis required NetworkX while and Scikit-learn along with NumPy were utilized to pre-process the data and extract features from it. The Flan-T5 Transformer-based Automated Attack Reporting System operated through implementation of the Hugging Face Transformers library to create real-time forensic reports.

4.4.2 Performance Results Evaluation and Discussion

The pie chart in figure 4.2 presents the distribution of data traffic labels so as to observe Benign and Bot flow percentages. This analysis shows that 72.7% of observed traffic belongs to the Benign category whereas Bot traffic makes up 27.3%.

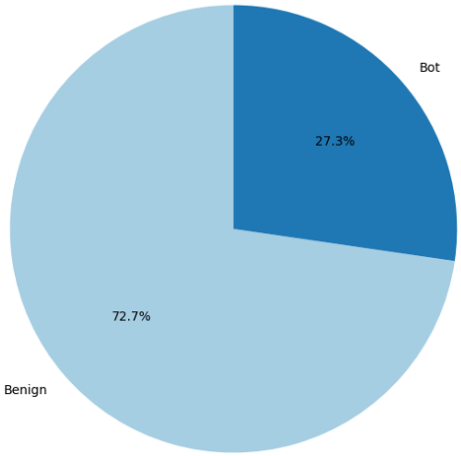


Figure 4.2: Label distribution

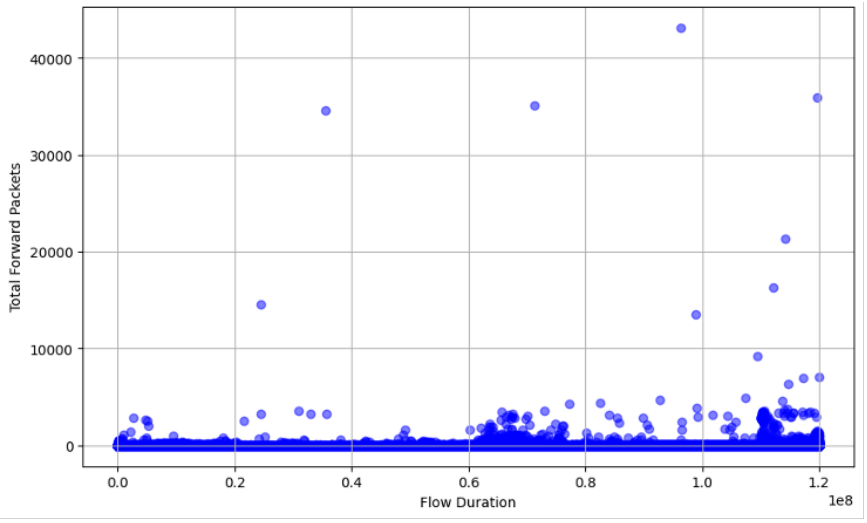


Figure 4.3: Scatter plot of flow duration vs. total forward packets

Figure 4.3 shows the scatter plot for the network traffic data. The Flow Duration relates to Total Forward Packets in the scatter plot. Most network flows concentrate near short Flow Duration values according to the data distribution patterns. A few anomalous observations show extended time-span lasting with elevated forward

packet transmission which denote extensive botnet operations or distributed assault activities.

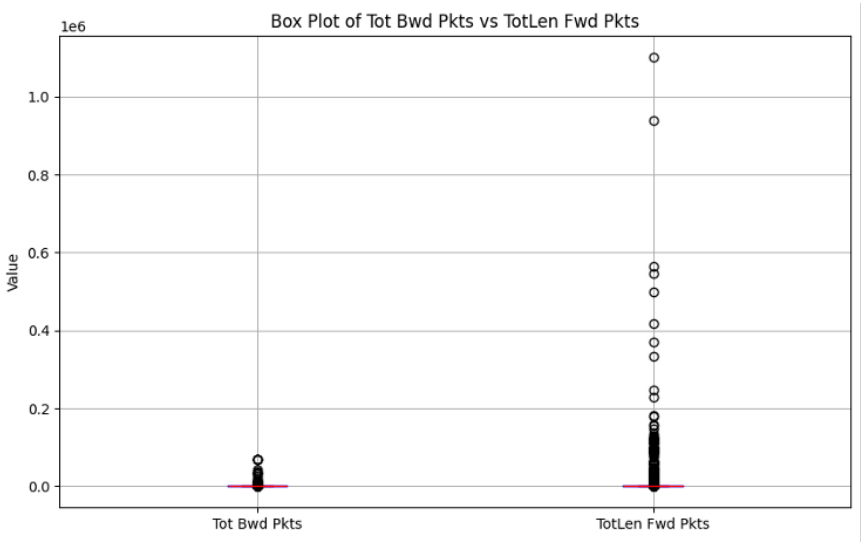


Figure 4.4: Box plot of total backward packets vs total forward packet length

The graphical representation given in Figure 4.4 shows how total backward packets and total forward packet length are distributed among different cases, which are commonly used in network traffic analysis, particularly in intrusion detection, anomaly detection, and network forensics. They describe packet flow characteristics between two communicating entities. Various departing packets from the network possess unusually large sizes according to the total forward packet length data which displays multiple outliers throughout the plot. Uncommon or unwanted network activities including data exfiltration attempts and significant network scanning actions can be detected through this deviation.

The correlation matrix presented in Figure 4.5 shows the visualization of interaction relationships between numerous network flow features contained in the dataset. The intensity of colour spectrum runs from deep blue for negative correlations through deep red for positive correlations to display the levels of feature association. The diagonal features demonstrate perfect correlation through self-comparison while displaying a value of 1. The data transmission patterns reveal the indirect dependencies between network attributes because their total forward packets and total forward bytes show strong positive correlations. Features that display negative correlation relationships show that their values maintain an inverse pattern when one variable goes up another decreases.

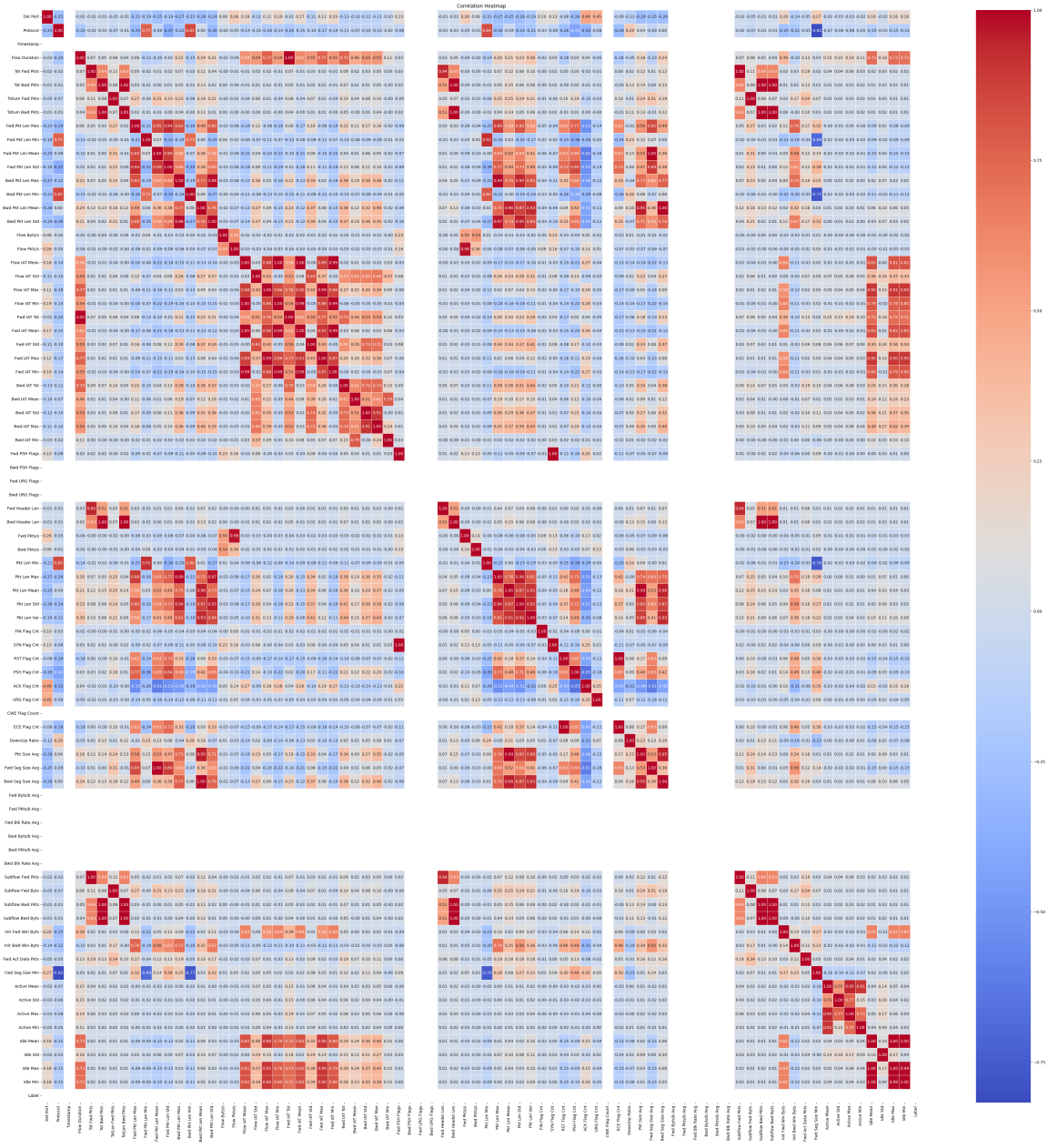


Figure 4.5: Correlation matrix

From Figure 4.6, it is understood that through the application of Hybrid ViT-GraphSAGE the system reached 98.06% accuracy with successful detection of SD-IoT network intrusions. The model exhibits high precision of 98.28% and recall value of 99.75% thus establishing its capability to accurately detect both normal and attack traffic while generating few false alarm alerts. The detection capability demonstrated by the F-measure score of 99.01% indicates its robustness against new attack patterns

because it shows balanced detection performance. The research proves the combined use of ViT and GraphSAGE networks which combine global traffic analysis from ViT with local node relationship analysis from GraphSAGE to boost intrusion detection performance.

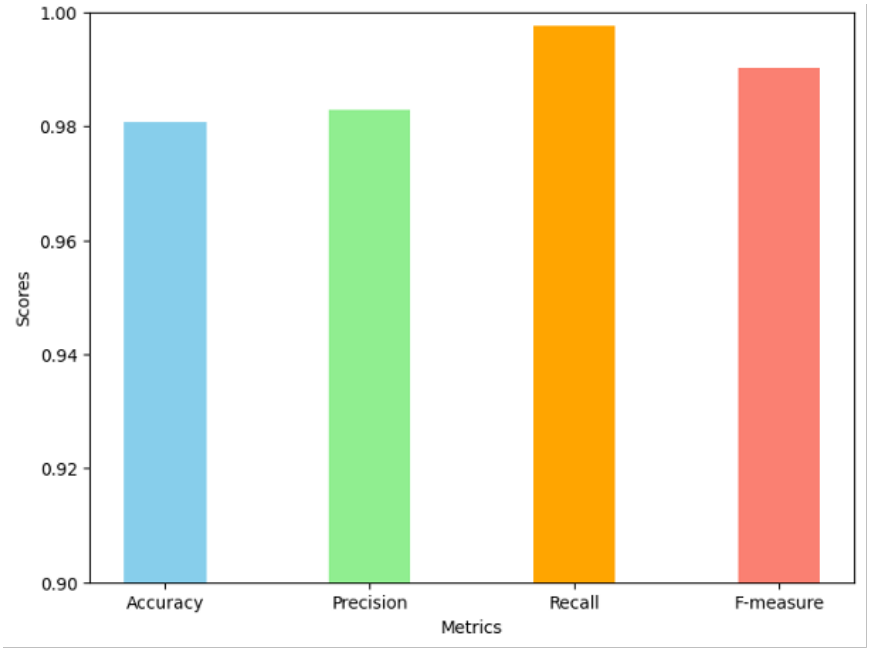


Figure 4.6: Performance metrics of Hybrid ViT-GraphSAGE model

Figure 4.7 shows that the Federated Learning method for the proposed collaborative intrusion detection systems achieved an accuracy level of 97.27%. Also, this FL model used in the proposed collaborative intrusion detection framework shows a high precision of 97.56% and a recall of 99.64% to detect intrusions without endangering data privacy between multiple edge nodes. Evaluation through the F-measure indicated the model performed with 98.59% reliability for network traffic classification. Through this FL, the devices operating in the SD-IoT space can work together to build collaborative intrusion detection models utilizing privacy-protected data without the need to concentrate sensitive information.

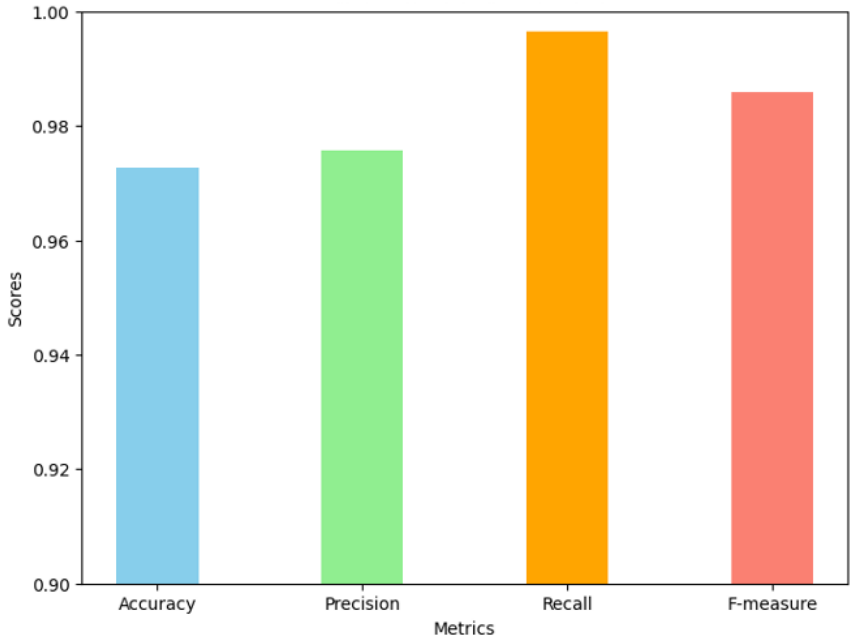


Figure 4.7: Performance metrics of Federated Learning-based collaborative intrusion detection

Figure 4.8, shows that the Hybrid ViT-GraphSAGE model achieved superior accuracy as well as precision results compared to Federated Learning-based approach indicating that centralized data training delivers better feature learning capabilities. The detection of malicious activities reaches higher recall scores in both methods indicating strong abilities to detect dangerous activities while reducing false negative outcomes. The diminished accuracy rate compared to the Hybrid ViT-GraphSAGE stands logical since both models use differing approaches in distributed network environments of federated learning. FL establishes an anonymous training environment for intrusion detection models that enables distributed devices to collaborate via data protection features without needing centralized storage of sensitive information. The implemented framework which merges ViT-GraphSAGE detection alongside distributed training through FL demonstrates its ability to deliver dependable security measures for SD-IoT systems.

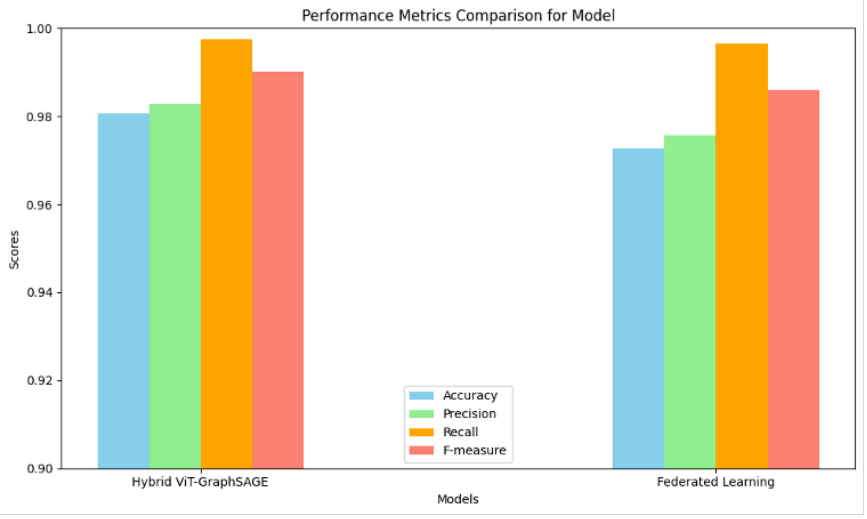


Figure 4.8: Comparison of models

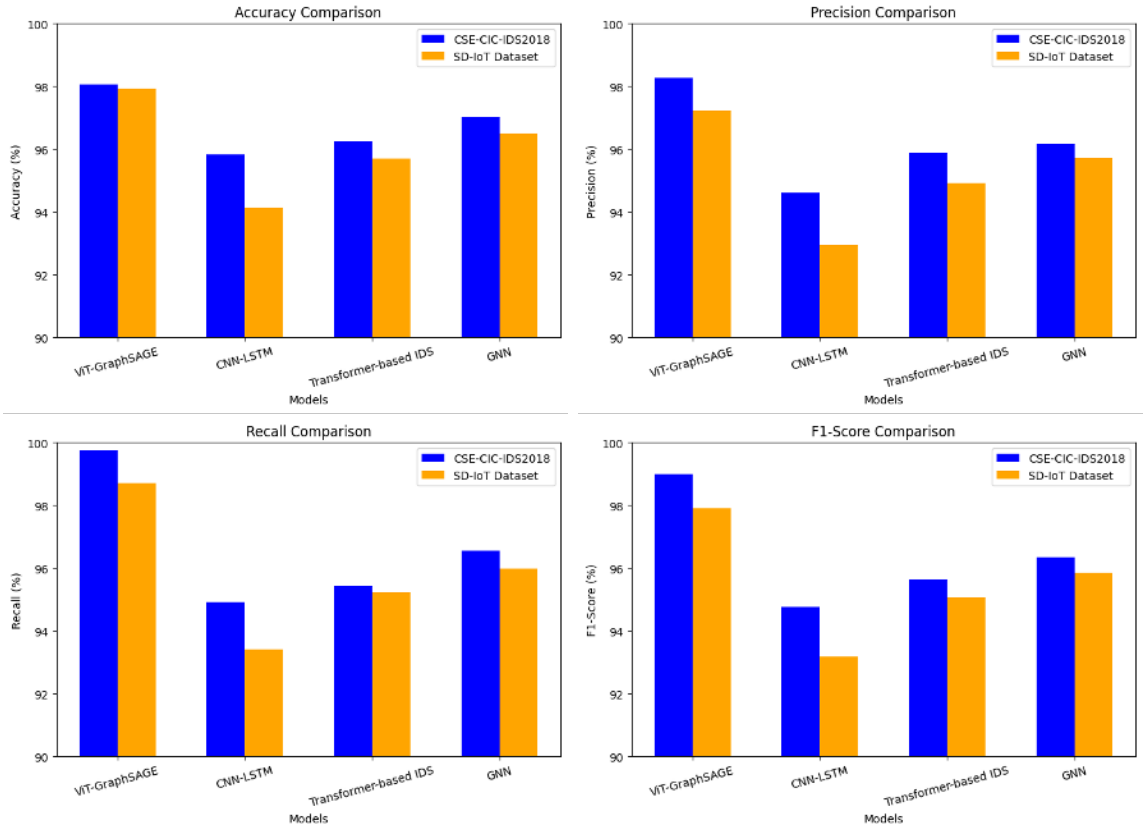


Figure 4.9: Performance Comparison on CSE-CIC-IDS2018 and SD-IoT Datasets

In figure 4.9, the performance comparison results of the proposed work and other

state of the arts methods such as Hybrid CNN-LSTM, Transformer based IDS, and GNN have been provided. These findings show that ViT-GraphSAGE performs better than other methodologies using both datasets through improved detection capabilities for intrusion methods. The intrusion detection capabilities of ViT-GraphSAGE surpass those of CNN-LSTM hybrid and transformer-based IDS models because the model detects minor and major attack patterns with superior performance metrics of F1-score and Recall. Implementation results with the SD-IoT dataset show that this solution works well with changing conditions in IoT systems and therefore is suitable for actual cybersecurity deployments. The incorporation of Flan-T5 Transformer and Deep Q-Learning approaches inside the model enhances effective decision-making during the analysis of sophisticated attack patterns. GNN achieved competitive results specifically in the CSE-CIC-IDS2018 experiments while ViT-GraphSAGE provided marginally better outcomes in the SD-IoT experiments. The integration of ViT with graph-based models shows higher accuracy because it contributes added functionality for recognizing both spatial and temporal patterns.

4.5 Conclusions

In this chapter, we proposed a comprehensive security framework for SD-IoT networks combining a hybrid ViT-GraphSAGE intrusion detection model, a Multi-Agent Deep Q-Learning (DQL)-based attack mitigation strategy and an automated threat reporting system using Flan-T5 Transformer. By using ViT-GraphSAGE, both global traffic patterns and local node interactions were captured, which improved the detection accuracy of 98.06%. The multi-agent DQL effectively rerouted the traffic in real-time and proactively mitigated attacks, thereby finally reducing the impact of intrusions. The Flan-T5 Transformer could provide automated and detailed forensic reports on detected threats, and improved the response and decision making. To improve scalability and privacy, this proposed model employed FL, so as to train the model across distributed edge nodes without collecting data in the centralized node with an accuracy of 97.27%.

Chapter 5

Discussion, Analysis and Comparisons

In this chapter, we will discuss the importance of features that offers a great informative data to our models. In addition, we will discuss the potential limitations of the proposed models in Section 5.2. Finally, we conclude this chapter with some analysis and comparisons in Section 5.3.

5.1 A discussion on features importance

In this implementation study, the network traffic features including flow duration and total forward and backward packets together with packet length statistics and entropy-based characteristics functioned as main indicators to differentiate malicious from benign traffic. Flow duration serves as a crucial element to differentiate between transient malicious connections because malignant traffic normally shows extended or unexpectedly short period lengths. The identification of potential botnet and DoS attacks depends on analyzing asymmetrical communication patterns using total forward and backward packet counts. The analysis of mean and maximum packet lengths gave analysts information about typical flow patterns to identify evasive behavior caused by attackers who use fragmented or oversized packets. In order to learn complex patterns and hierarchical relationships within the dataset, these features were used by the ViT-GraphSAGE model, combined with Flan-T5 Transformer and Deep Q-Learning. Specifically, the packet-based statistical attributes together with entropy-based attributes allowed detection of anomalous behavior while they

provided enhanced detection ability for sophisticated attacks by monitoring packet distribution patterns.

5.2 Potential Limitation of the Proposed Models

A key feature of the proposed secure and intelligent intrusion detection system shows competent performance for detecting SD-IoT cyber threats but it does possess known restrictions to note. Its performance depends strongly on the quality of input training data because insufficient attack variations may cause both overfitting and underperformance in operational networks. The effective spatial-temporal relationship extraction of ViT-GraphSAGE presents scalability issues for IoT environments that have resource limitations. The performance of this model could suffer degradation from adversarial attacks which include both evasion and poisoning tactics so robust defensive systems must be developed to protect against these threats. Real-time deployment needs optimized inference time for graph-based computations because these computations cause latency essential for maintaining detection accuracy. These limitations are to be addressed in a future work that incorporates adaptive learning techniques with lightweight model compression along with adversarial combative strategies thereby enhancing the overall resilience and efficiency of the proposed system.

5.3 Comparison Analysis Using Additional Dataset

In this section, we are adding an additional dataset to our experiments presented in Chapter 3. Expanding the experimentation to include other intrusion datasets would strengthen the claim of robustness across diverse network environments. For this, we have chosen a comprehensive realistic cyber security dataset called (Edge-IIoTset Cyber Security Dataset of IoT and IIoT) available online to the research community [74]. Specifically, the dataset organized into multiple layers, including Cloud Computing Layer, Network Functions Virtualization Layer, Fog Computing Layer, Software-Defined Networking Layer, and Edge Computing Layer. In this dataset, cyber attacks are categorized into five threats, including DoS/DDoS attacks, Information gathering, Man in the middle attacks, Injection attacks, and Malware attacks.

Table 5.1: COMPARATIVE ANALYSIS WITH 70% TRAINING DATA - Edge-IIoTset DATASET

Metrics/Method	**GRU[24]**	**CNN[25]**	**SVM[26]**	**LSTM[27]**	**Proposed**
Accuracy	0.934521	0.961235	0.97932	0.91743	0.999249
Precision	0.912314	0.948256	0.973264	0.883562	0.998184
Sensitivity	0.911233	0.949372	0.974142	0.883367	0.998674
Specificity	0.961522	0.979283	0.991812	0.953689	0.999851
F-Measure	0.911233	0.948256	0.973264	0.883367	0.998674
MCC	0.876395	0.926845	0.968525	0.823129	0.9980531
NPV	0.961522	0.979283	0.991812	0.953689	0.999612
FPR	0.038478	0.020717	0.008188	0.046311	0.000149
FNR	0.088767	0.050628	0.025858	0.116633	0.001326

The performance of several machine learning models (GRU, CNN, SVM, LSTM, and the proposed approach) utilizing 70% of the training data is compared in Table 5.1. The findings demonstrate that the suggested approach performs noticeably better than the alternative models on every criterion. In particular, the suggested approach attains the highest sensitivity (0.998674), specificity (0.999851), accuracy (0.999249), precision (0.998184), F-measure (0.998674), MCC (0.998053), and NPV (0.999612). It also performs better in identifying and reducing false positives and false negatives, as seen by its lowest FPR (0.000149) and FNR (0.001326). The success of the suggested approach in feature selection and detection tasks is demonstrated by the lower values across these metrics displayed by the other models (such as GRU, CNN, SVM, and LSTM) in comparison.

Table 5.2: COMPARATIVE ANALYSIS WITH 80% TRAINING DATA - Edge-IIoTset DATASET

Metrics/Method	**GRU [24]**	**CNN [25]**	**SVM [26]**	**LSTM [27]**	**Proposed**
Accuracy	0.951742	0.976542	0.986137	0.923076	0.999876
Precision	0.927843	0.961842	0.977922	0.881283	0.99954
Sensitivity	0.927843	0.961842	0.977922	0.881283	0.999462
Specificity	0.955874	0.974563	0.986522	0.93824	0.999953
F-Measure	0.927843	0.961842	0.977922	0.881283	0.999462
MCC	0.89942	0.94156	0.97054	0.830523	0.999845
NPV	0.955874	0.974563	0.986522	0.93824	0.99997
FPR	0.044126	0.025437	0.013478	0.06176	4.70E-05
FNR	0.072157	0.038158	0.022078	0.118717	0.000538

Table 5.2 compares the various models (GRU, CNN, SVM, LSTM, and the proposed approach) utilizing 80% of dataset training data. In every metric, the suggested approach outperforms all other models, exhibiting remarkable performance. Accuracy (0.999876), precision (0.99954), sensitivity (0.999462), specificity (0.999953), F-measure (0.999462), MCC (0.999845), and NPV (0.99997) are all at their highest scores. Furthermore, the suggested approach has the lowest FPR (4.70E-05) and FNR (0.000538), suggesting a low number of false positives and false negatives. When compared to other models like GRU, CNN, SVM, and LSTM, the suggested method consistently performs better in detecting and classifying instances while retaining a high degree of efficiency and reliability in its predictions.

5.4 Training Time and Inference Time Analysis

Across two datasets, Table 5.3 contrasts the training and inference times of several machine learning models (GRU, CNN, SVM, LSTM, and the proposed approach). The proposed approach consistently shows the quickest training and inference times for both datasets. The proposed method's training duration in Dataset 1 is 150 seconds, which is noticeably faster than that of the other models, LSTM's was the longest at 900 seconds. In a similar vein, the suggested method's inference time per sample is the shortest at 0.01 seconds when contrasted with other models such as CNN

and LSTM, which need 0.05 and 0.1 seconds, respectively. With a training time of 400 seconds and an inference time of 0.015 seconds, the suggested approach continues to perform better than the others for Dataset 2. This suggests that the suggested approach provides an effective way to train the model and provide predictions in real time.

Table 5.3: Analysis of Training Time and Inference Time

Dataset	**Metrics**	GRU [24]	CNN [25]	SVM [26]	LSTM [27]	Proposed
SDN Intrusion Detection	Training Time (seconds)	180	600	300	900	150
	Inference Time (seconds per sample)	0.02	0.05	0.03	0.1	0.01
Edge-IIoTset	Training Time (seconds)	250	900	600	1200	400
	Inference Time (seconds per sample)	0.03	0.05	0.05	0.12	0.015

Chapter 6

Conclusions and Future Work

In this dissertation, we went through the journey of exploring and applying the capabilities of artificial intelligence and central management of SDN into a dynamic, effective, and secure SDN and SD-IoT environment. We began by describing SDN, its advantages over traditional networks, and how it is implemented with the Openflow protocol. Also, we discussed SDN vulnerabilities and security. Since SDN have revolutionized network management by decoupling control and data planes, providing dynamic and programmable network infrastructure. However, with this flexibility comes the challenge of securing SDNs against cyber threats.

Conventional security measures often struggle to adapt to the evolving landscape of cyber attacks in SDN environments. The rapid growth of SDNs has necessitated the development of sophisticated and adaptive security mechanisms. Traditional methods, such as signature-based detection and rule-based mitigation, are often insufficient to counter the diversity and complexity of modern cyber attacks. There is a need for intelligent and automated approaches that can effectively detect and mitigate attacks in SDNs.

The main motivation of this work is to enhance the detection and mitigation of cyber attacks in SDNs by leveraging advanced technologies, specifically deep learning and reinforcement learning. The proposed approach involves a comprehensive framework that integrates data preprocessing, feature extraction, and two distinct phases of machine learning – deep learning for attack detection and reinforcement learning for dynamic attack mitigation. In addition, by using ViT-GraphSAGE, both global traffic patterns and local node interactions were captured, which improved the detection accuracy. The multi-agent DQL effectively rerouted the traffic in real-time and proactively mitigated attacks, thereby finally reducing the impact of intrusions.

The Flan-T5 Transformer provided an automated and detailed forensic reports on detected threats, and improved the response and decision making. To improve scalability and privacy, this proposed model employed Federated Learning FL, so as to train the model across distributed edge nodes without collecting data in the centralized node with an accuracy. Overall, this dissertation addressed the challenges of high false positive rate, adapted to evolving attacks, thereby being suitable to be a robust and scalable solution for SD-IoT environments.

As we come to the conclusion of this work, we suggest some extension of the work to improve the system model performance and efficiency. Apply ensemble learning methods to combine multiple AI models. This approach can mitigate the limitations of individual models and improve the overall resilience of the system in order to make the security of the network tighter to resist the constantly changing cyber threats. Future work should also expand the experiments to include other intrusion datasets to develop models that are more generalizable, robust, and capable of handling diverse scenarios.

Appendix

Figures 1, and 2 show a sample output while the Multi-Agent DQL preforming attack mitigation and predictive mitigation using Temporal Convolutional Network (TCN). The multi-agent DQL effectively rerouted the traffic in real-time and proactively mitigated attacks, thereby finally reducing the impact of intrusions.

Figures 3, and 4 show a sample output of model training. Such as; feature extraction using CapsNet and Hybrid ViT-GraphSAGE model training for Intrusion Detection.

In Figure 5, we see Federated Learning-Based Collaborative Intrusion Detection, and automated attack reporting system, which provide automated and detailed forensic reports on detected threats, and improved response and decision-making.

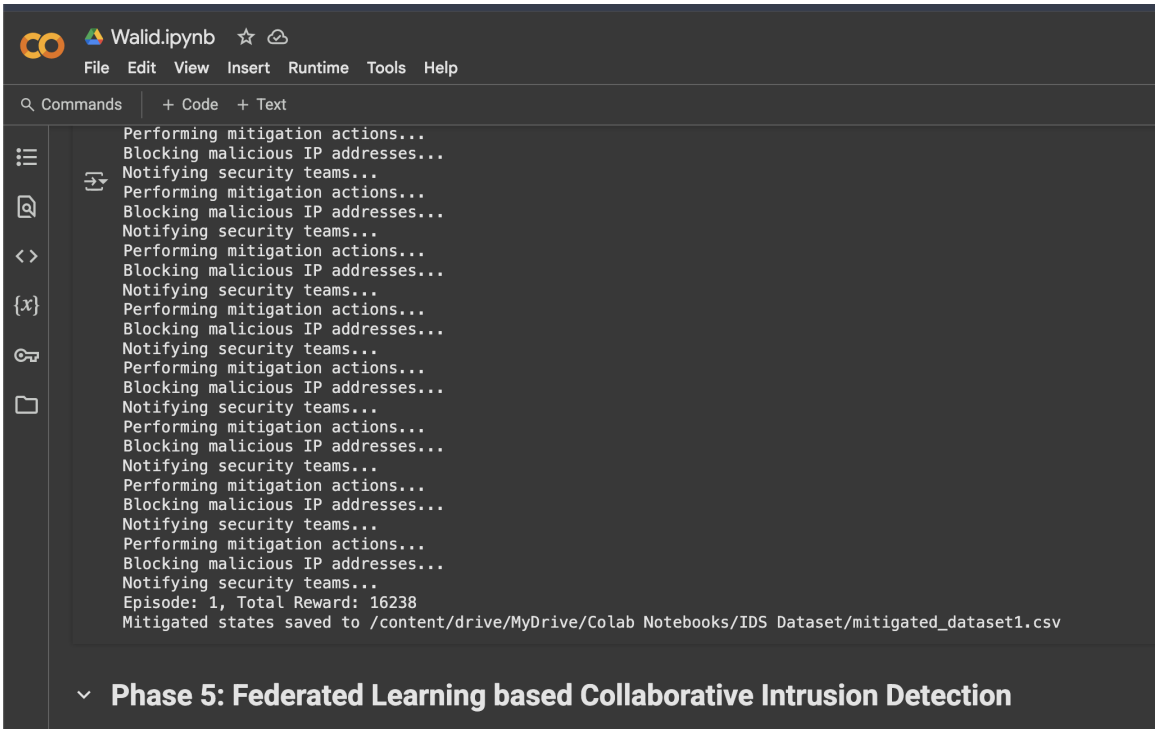
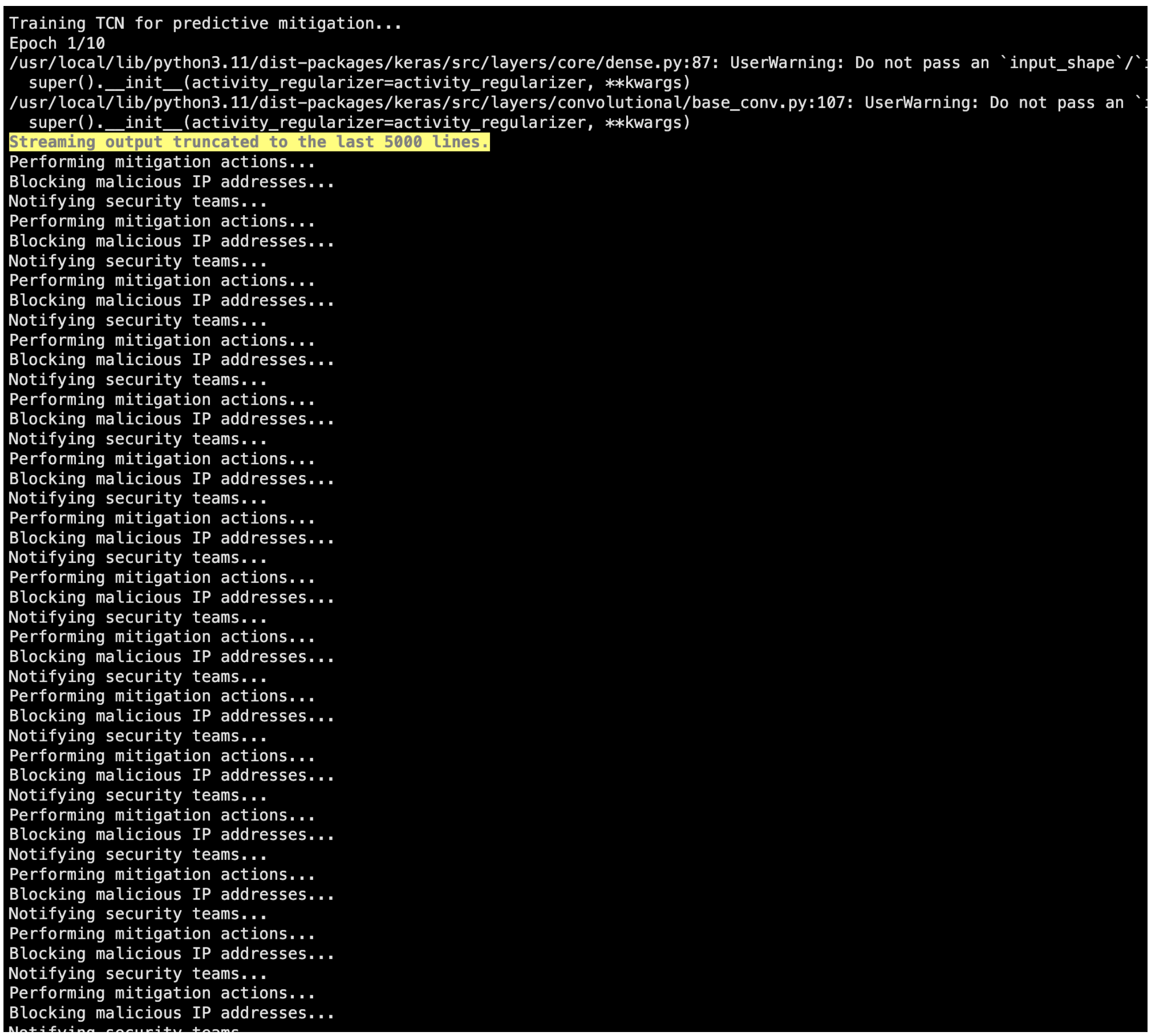
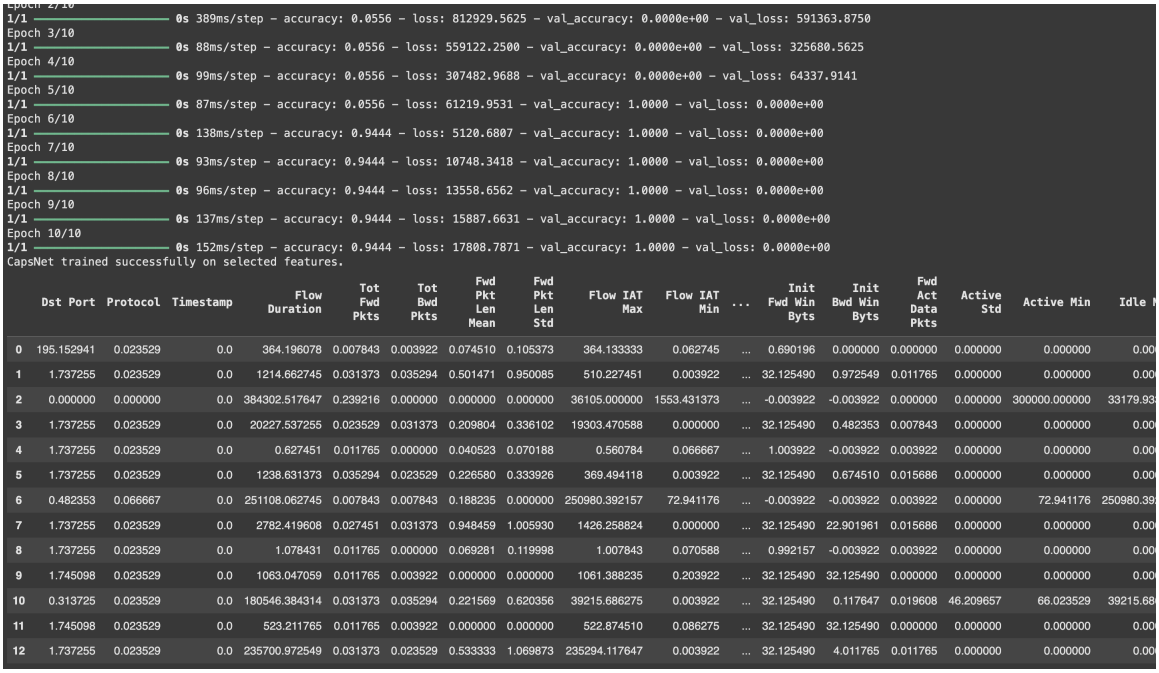


Figure 1: Multi-Agent DQL Preforming Attack Mitigation Actions



```
Training TCN for predictive mitigation...
Epoch 1/10
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Streaming output truncated to the last 5000 lines.
Performing mitigation actions...
Blocking malicious IP addresses...
Notifying security teams...
Performing mitigation actions...
Blocking malicious IP addresses...
Notifying security teams...
Performing mitigation actions...
Blocking malicious IP addresses...
Notifying security teams...
Performing mitigation actions...
Blocking malicious IP addresses...
Notifying security teams...
Performing mitigation actions...
Blocking malicious IP addresses...
Notifying security teams...
Performing mitigation actions...
Blocking malicious IP addresses...
Notifying security teams...
Performing mitigation actions...
Blocking malicious IP addresses...
Notifying security teams...
Performing mitigation actions...
Blocking malicious IP addresses...
Notifying security teams...
Performing mitigation actions...
Blocking malicious IP addresses...
Notifying security teams...
Performing mitigation actions...
Blocking malicious IP addresses...
Notifying security teams...
Performing mitigation actions...
Blocking malicious IP addresses...
Notifying security teams...
Performing mitigation actions...
Blocking malicious IP addresses...
Notifying security teams...
Performing mitigation actions...
Blocking malicious IP addresses...
Notifying security teams...
Performing mitigation actions...
Blocking malicious IP addresses...
Notifying security teams...
Performing mitigation actions...
Blocking malicious IP addresses...
```

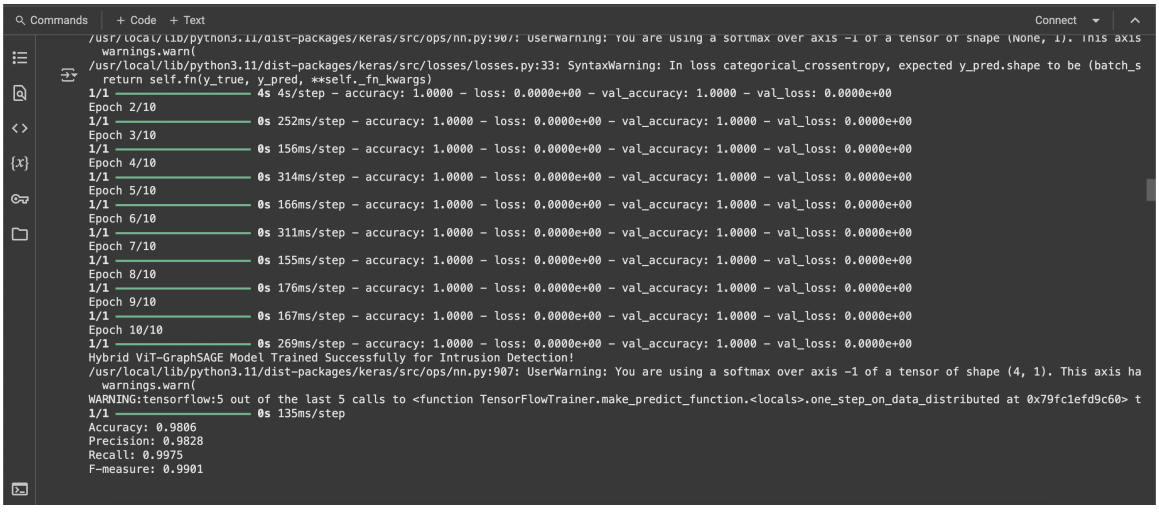
Figure 2: Predictive Mitigation using Temporal Convolutional Network (TCN)



```
Epoch 2/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 389ms/step - accuracy: 0.0556 - loss: 812929.5625 - val_accuracy: 0.0000e+00 - val_loss: 591363.8750
Epoch 3/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step - accuracy: 0.0556 - loss: 559122.2500 - val_accuracy: 0.0000e+00 - val_loss: 325680.5625
Epoch 4/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 99ms/step - accuracy: 0.0556 - loss: 307482.9688 - val_accuracy: 0.0000e+00 - val_loss: 64337.9141
Epoch 5/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 87ms/step - accuracy: 0.0556 - loss: 61219.9531 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 6/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step - accuracy: 0.9444 - loss: 5120.6807 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 7/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.9444 - loss: 10748.3418 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 8/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step - accuracy: 0.9444 - loss: 13558.6562 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 9/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - accuracy: 0.9444 - loss: 15887.6631 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 10/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 152ms/step - accuracy: 0.9444 - loss: 17808.7871 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
CapsNet trained successfully on selected features.
```

	Dst Port	Protocol	Timestamp	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	Fwd Pkt Len Mean	Fwd Pkt Len Std	Flow IAT Max	Flow IAT Min	...	Init Fwd Win Byts	Init Bwd Win Byts	Fwd Act Data Pkts	Active Std	Active Min	Idle M
0	195.152941	0.023529	0.0	364.196078	0.007843	0.003922	0.074510	0.105373	364.133333	0.062745	...	0.690196	0.000000	0.000000	0.000000	0.000000	0.00
1	1.737255	0.023529	0.0	1214.662745	0.031373	0.035294	0.501471	0.950085	510.227451	0.003922	...	32.125490	0.972549	0.011765	0.000000	0.000000	0.00
2	0.000000	0.000000	0.0	384302.517647	0.239216	0.000000	0.000000	0.000000	36105.000000	1553.431373	...	-0.003922	-0.003922	0.000000	0.000000	300000.000000	33179.93
3	1.737255	0.023529	0.0	20227.537255	0.023529	0.031373	0.209804	0.336102	19303.470588	0.000000	...	32.125490	0.482353	0.007843	0.000000	0.000000	0.00
4	1.737255	0.023529	0.0	0.627451	0.011765	0.000000	0.040523	0.070188	0.560784	0.066667	...	1.003922	-0.003922	0.003922	0.000000	0.000000	0.00
5	1.737255	0.023529	0.0	1238.631373	0.035294	0.023529	0.226580	0.333926	369.494118	0.003922	...	32.125490	0.674510	0.015686	0.000000	0.000000	0.00
6	0.482353	0.066667	0.0	251108.062745	0.007843	0.007843	0.188235	0.000000	250980.392157	72.941176	...	-0.003922	-0.003922	0.003922	0.000000	72.941176	250980.39
7	1.737255	0.023529	0.0	2782.419608	0.027451	0.031373	0.948459	1.005930	1426.258824	0.000000	...	32.125490	22.901961	0.015686	0.000000	0.000000	0.00
8	1.737255	0.023529	0.0	1.078431	0.011765	0.000000	0.069281	0.119998	1.007843	0.070588	...	0.992157	-0.003922	0.003922	0.000000	0.000000	0.00
9	1.745098	0.023529	0.0	1063.047059	0.011765	0.003922	0.000000	0.000000	1061.388235	0.203922	...	32.125490	32.125490	0.000000	0.000000	0.000000	0.00
10	0.313725	0.023529	0.0	180546.384314	0.031373	0.035294	0.221569	0.620356	39215.686275	0.003922	...	32.125490	0.117647	0.019608	46.209657	66.023529	39215.68
11	1.745098	0.023529	0.0	523.211765	0.011765	0.003922	0.000000	0.000000	522.874510	0.086275	...	32.125490	32.125490	0.000000	0.000000	0.000000	0.00
12	1.737255	0.023529	0.0	235700.972549	0.031373	0.023529	0.533333	1.069873	235294.117647	0.003922	...	32.125490	4.011765	0.011765	0.000000	0.000000	0.00

Figure 3: Model Training - Feature Extraction using CapsNet



```
Commands  + Code  + Text                                                                                    Connect
/usr/local/lib/python3.11/dist-packages/keras/src/ops/nn.py:907: UserWarning: You are using a softmax over axis -1 of a tensor of shape (None, 1). This axis
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/keras/src/losses/losses.py:33: SyntaxWarning: In loss categorical_crossentropy, expected y_pred.shape to be (batch_s
  return self.fn(y_true, y_pred, **self._fn_kwargs)
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 2/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 252ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 3/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 156ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 4/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 314ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 5/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 166ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 6/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 311ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 7/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 155ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 8/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 176ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 9/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 167ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 10/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 269ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Hybrid ViT-GraphSAGE Model Trained Successfully for Intrusion Detection!
/usr/local/lib/python3.11/dist-packages/keras/src/ops/nn.py:907: UserWarning: You are using a softmax over axis -1 of a tensor of shape (4, 1). This axis ha
  warnings.warn(
WARNING:tensorflow:5 out of the last 5 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x79fc1efd9c60> t
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step
Accuracy: 0.9806
Precision: 0.9828
Recall: 0.9975
F-measure: 0.9901
```

Figure 4: Hybrid ViT-GraphSAGE Model Training for Intrusion Detection

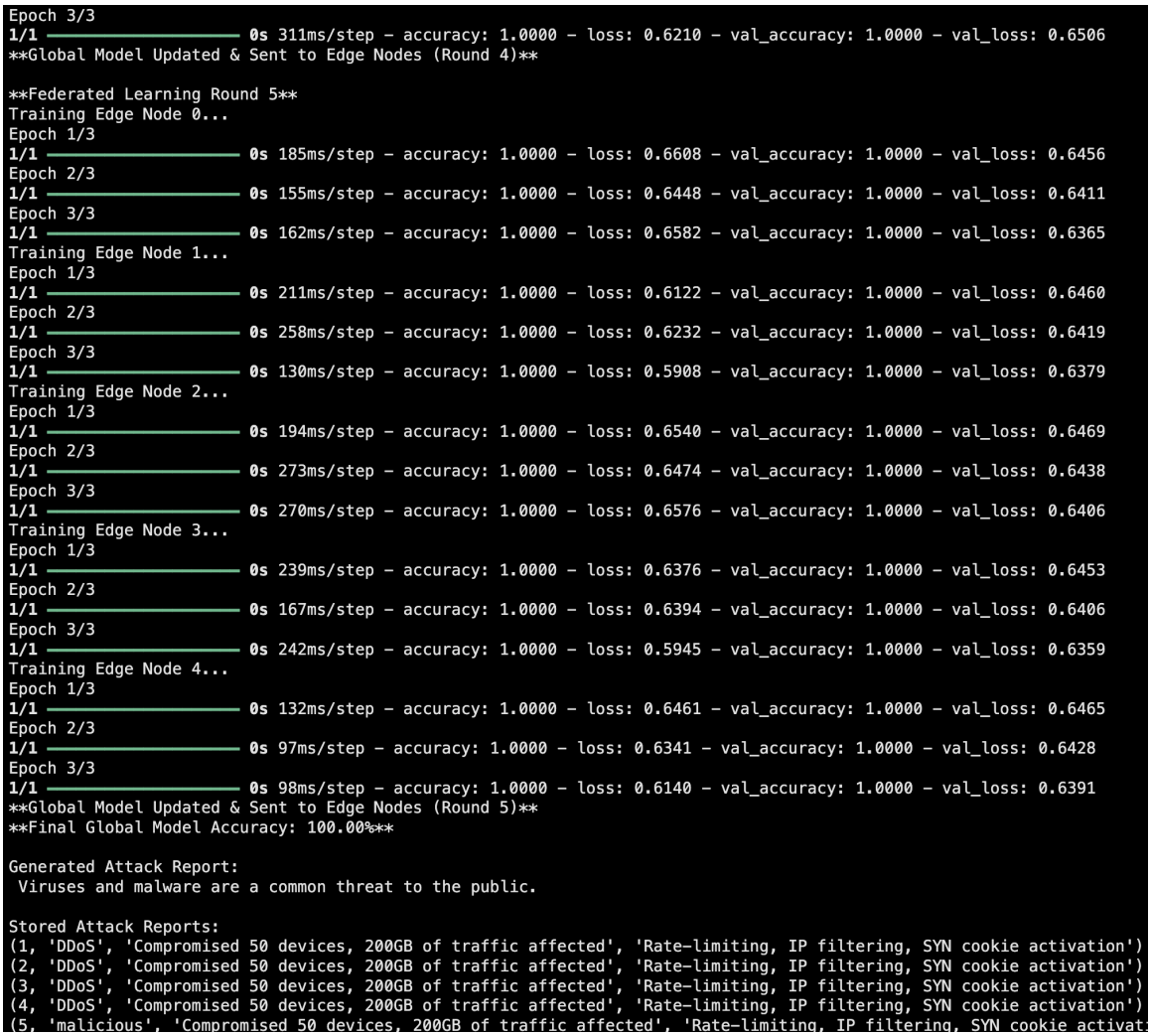


Figure 5: Federated Learning-Based Collaborative Intrusion Detection, and automated attack reporting system

Bibliography

[1] Negera WG, Schwenker F, Debelee TG, Melaku HM, Ayano YM. "Review of botnet attack detection in SDN-enabled IoT Using machine learning," Sensors. 2022 Dec 14;22(24):9837.

[2] Aslam N, Srivastava S, Gore MM. "A comprehensive analysis of machine learning-and deep learning-based solutions for DDoS attack detection in SDN," Arabian Journal for Science and Engineering. 2024 Mar;49(3):3533-73.

[3] Ali TE, Chong Y-W, Manickam S. "Machine Learning Techniques to Detect a DDoS Attack in SDN: A Systematic Review," Applied Sciences. 2023; 13(5):3183.

[4] Goud K and Giduturi S, "Enhancing DDoS Attack Detection in SDN: A Novel Approach with IG-RFFI Feature Selection," 2024. 161-169. 10.1007/978-981-99-9704-6-14.

[5] A. Bates, K. Butler, A. Haeberlen, M. Sherr, and W. Zhou. "Let SDN be your eyes: Secure forensics in data center networks." Workshop SENT. 2014. 1-7.

[6] Carvalho R.N, Costa L.R, Bordim J.L, Alchieri E.A. DataPlane-ML: an integrated attack detection and mitigation solution for software defined networks. Concurrency and Computation: Practice and Experience. 2023 Aug 30;35(19):e7434.

[7] Aladaileh MA, Anbar M, Hintaw AJ, Hasbullah IH, Bahashwan AA, Al-Amiedy TA, Ibrahim DR. Effectiveness of an Entropy-Based Approach for Detecting Low- and High-Rate DDoS Attacks against the SDN Controller: Experimental Analysis. Applied Sciences. 2023; 13(2):775.

[8] Negera WG, Schwenker F, Debelee TG, Melaku HM, Feyisa DW. Lightweight Model for Botnet Attack Detection in Software Defined Network-Orchestrated IoT. Applied Sciences. 2023; 13(8):4699.

[9] Al-Dunainawi Y, Al-Kaseem BR, Al-Raweshidy HS. Optimized Artificial Intelligence Model for DDoS Detection in SDN Environment. IEEE Access. 2023 Sep 25.

[10] OpenFlow Switch Specification, v.1.5.1. ONF TS-025. March 2015.

[11] M. Ku´zniar, P. Perešíni, and D. Kosti. "What you need to know about sdn flow tables." (Springer) 347–359.

[12] Ma R, Wang Q, Bu X, Chen X. "Real-Time Detection of DDoS Attacks Based on Random Forest in SDN". Applied Sciences. 2023 Jul 4;13(13):7872.

[13] T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner N. McKeown, "OpenFlow: Enabling innovation in campus networks," ACM SIGCOMM Computer Communications Review, Apr 2008.

[14] A. Bianco, R. Birke, L. Giraudo, and M. Palacin. "OpenFlow switching: Data plane performance." IEEE Int. Conf. Commun., May 2010.

[15] A. Capone, C. Cascone, A. Q. T. Nguyen, and B. Sansò. "Detour planning for fast and reliable failure recovery in SDN with OpenState." 11th International Conference on the Design of Reliable Communication Networks (DRCN'15), 2015: 25-32.

[16] A. Guha, M. Reitblatt, and N. Foster. "Machine-verified network controllers." ACM SIGPLAN Notices 48, no. 6: 483–494.

[17] R. Braga, E. Mota, and A. Passito. "Lightweight DDoS flooding attack detection using NOX/OpenFlow." IEEE 35th Conf. LCN, 2010: 408–415.

[18] Ballard, J.R., Rae, I., Akella, A. "Extensible and scalable network monitoring using OpenSAFE." the 2010 internet network management conference on Research on enterprise networking (INM/WREN'10), 2010: 88.

[19] Gu, S. Shin and G. "CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks or: How to provide security monitoring as a service in clouds?" 20th IEEE ICNP, 2012: 1–6.

[20] R. Skowyra, S. Bahargam, and A. Bestavros. "Software-defined IDS for securing embedded mobile devices." Boston Univ. Boston, MA, USA. 2013.

[21] B. Anwer, T. Benson, N. Feamster, D. Levin, and J. Rexford. "A Slick Control Plane for Network Middleboxes." 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Network, 2013: 147–148.

[22] Shin, S. "FRESCO: Modular composable security services for software-defined networks." Netw. Distrib. Security Symp., 2013: 1–16.

[23] Ganjali, S. H. Yeganeh and Y. "Kandoo: A framework for efficient and scalable offloading of control applications." (in Proc. 1st Workshop Hot Topics Softw. Defined Netw.) 2012: 19–24.

[24] Elubeyd H, Yiltas-Kaplan D. Hybrid deep learning approach for automatic Dos/DDoS attacks detection in software-defined networks. Applied Sciences. 2023 Mar 16;13(6):3828.

[25] Hnamte V, Hussain J. An efficient DDoS attack detection mechanism in SDN environment. International Journal of Information Technology. 2023 Jun;15(5):2623-36.

[26] Bhayo J, Shah SA, Hameed S, Ahmed A, Nasir J, Draheim D. Towards a machine learning-based framework for DDOS attack detection in software-defined IoT (SD-IoT) networks. Engineering Applications of Artificial Intelligence. 2023 Aug 1;123:106432.

[27] Gebremeskel TG, Gemeda KA, Krishna TG, Ramulu PJ. DDoS attack detection and classification using hybrid model for multicontroller SDN. Wireless Communications and Mobile Computing. 2023 Jun 23;2023.

[28] Ganjali, A. Tootoonchian and Y. "HyperFlow: A distributed control plane for OpenFlow." in Proc. Internet Netw. Manage. Conf. Res. Enterprise Network. 2010. 03.

[29] R. Soule, S. Basu, R. Kleinberg, E. G. Sirer, and N. Foster. "Managing the network with Merlin." 12th ACM Workshop Hot Topics Netw., Nov 2013.

[30] Ganjali, S. H. Yeganeh and Y. "Kandoo: A framework for efficient and scalable offloading of control applications." (in Proc. 1st Workshop Hot Topics Softw. Defined Netw.) 2012: 19–24.

[31] Foster, N. "Frenetic: A network programming language." ACM SIGPLAN Notices 46, no. 9: 279–291.

[32] A. Voellmy, J. Wang, Y. R. Yang, B. Ford, P. Hudak. "Maple: Simplifying SDN programming using algorithmic policies." ACM SIGCOMM Conf. 2013. 87–98.

[33] A. Voellmy, P. Hudak. "Nettle: A language for configuring routing networks." IFIP TC 2 Working Conf. DSL, 2009: 211-235.

[34] A. Voellmy, P. Hudak. "Nettle: Taking the sting out of programming network routers." 13th Int. Conf. PADL, 2011: 235–249.

[35] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, D. Walker. "Abstractions for network update." ACM SIGCOMM Conf. Appl., Technol., Archit., Protocols Comput. Commun, 2012: 323–334.

[36] M. Reitblatt, N. Foster, J. Rexford, D. Walker. "Consistent updates for software-defined networks: Change you can believe in!" 10th ACM Workshop HotNets-X, 2011: 7:1–7:6.

[37] McGeer, R. "A safe, efficient update protocol for OpenFlow networks." 1st Workshop HotSDN, 2012: 61–66.

[38] K. Wang, Y. Qi, B. Yang, Y. Xue, J. Li,. "LiveSec: Towards effective security management in large-scale production networks." ICDCSW, June 2012: 451–460.

[39] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul. "Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags." NSDI, 2014: 533–546.

[40] W. Han, H. Hu, Z. Zhao, A. Doupé, G.-J. Ahn, K.-C. Wang, and J. Deng. "State-aware network access management for software-defined networks." the 21st ACM on Symposium on Access Control Models and Technologies (ACM), 2016: 1-11.

[41] A. S. da Silva, C. C. Machado, R. V. Bisol, L. Z. Granville, and A. Schaeffer-Filho. "Identification and selection of flow features for accurate traffic classification in SDN." 14th International Symposium on Network Computing and Applications (IEEE), 2015: 134–141.

[42] Greenberg, A. "A clean slate 4D approach to network control and management." 35, no. 5 (2005): 41–54.

[43] S. Hartman, M. Wasserman, D. Zhang. "Software driven networks problem statement." Network Working Group Internet-Draft. Oct 14, 2013.

[44] A. Zaalouk, R. Khondoker, R. Marx, K. Bayarou. "OrchSec: An orchestrator-based architecture for enhancing network-security using network monitoring and SDN control functions." IEEE NOMS, May 2014: 1–9.

[45] Vinayakumar, R., Soman, K.P., Poornachandran, P. and Akarsh, S., 2019. Application of deep learning architectures for cyber security. Cybersecurity and Secure Information Systems: Challenges and Solutions in Smart Environments, pp.125-160

[46] Iqbal, M., Iqbal, F., Mohsin, F., Rizwan, M. and Ahmad, F., 2019. Security issues in software defined networking (SDN): risks, challenges and potential solutions.

[47] BOUKRIA, S. and GUERROUMI, M., 2019, December. Intrusion detection system for SDN network using deep learning approach. In 2019 International Conference on Theoretical and Applicative Aspects of Computer Science (ICTAACS) (Vol. 1, pp. 1-6). IEEE.

[48] Sivanathan, A., Gharakheili, H.H. and Sivaraman, V., 2020. Managing IoT cyber-security using programmable telemetry and machine learning. IEEE Transactions on Network and Service Management, 17(1), pp.60-74.

[49] W. El Gadal and S. Ganti, "Dynamic Defense Framework: A Unified Approach for Intrusion Detection and Mitigation in SDN," 2024 8th Cyber Security in Networking Conference (CSNet), Paris, France, 2024, pp. 22-27, doi: 10.1109/CSNet64211.2024.10851474.

[50] Bhayo, J., Shah, S.A., Hameed, S., Ahmed, A., Nasir, J. and Draheim, D., 2023. Towards a machine learning-based framework for DDOS attack detection in software-defined IoT (SD-IoT) networks. Engineering Applications of Artificial Intelligence, 123, p.106432.

[51] Ferreira, R., Bispo, I., Rabadão, C., Santos, L. and Costa, R.L.D.C., 2025. Farmflow dataset: Intrusion detection in smart agriculture based on network flows. Computers and Electrical Engineering, 121, p.109892.

[52] Parthiban, N., Vijay, S. and Sheela, S.N., 2025. Comparison of mitigating DDoS attacks in software defined networking and IoT platforms. Cyber Security and Applications, 3, p.100080

[53] Wang J, Wang L, Wang R. A Method of DDoS Attack Detection and Mitigation for the Comprehensive Coordinated Protection of SDN Controllers. Entropy. 2023 Aug 14;25(8):1210.

[54] Tian Q, Miyata S. A DDoS Attack Detection Method Using Conditional Entropy Based on SDN Traffic. IoT. 2023 Apr 12;4(2):95-111.

[55] Chaganti R, Suliman W, Ravi V, Dua A. Deep learning approach for SDN-enabled intrusion detection system in IoT networks. Information. 2023 Jan 9;14(1):41.

[56] A. S. da Silva, J. A. Wickboldt, L. Z. Granville, and A. Schaeffer-Filho. "Atlantic: A framework for anomaly traffic detection, classification, and mitigation in sdn." the 15th IEEE/IFIP Network Operations and Management Symposium (IEEE), 2016: 27–35.

[57] I. Naruei and F. Keynia, "A new optimization method based on coot bird natural life model," Expert Systems with Applications, p. 115352,2021.

[58] Dehghani M, Bektemyssova G, Montazeri Z, Shaikemelev G, Malik OP, Dhiman G. Lyrebird Optimization Algorithm: A New Bio-Inspired Metaheuristic Algorithm for Solving Optimization Problems. Biomimetics. 2023; 8(6):507.

[59] Mnih, V., Kavukcuoglu, K., Silver, D. et al. "Human-level control through deep reinforcement learning," Nature 518, 529–533 (2015).

[60] "SDN Intrusion Detection Dataset" [Online]. Available: https://www.kaggle.com/datasets/subhajournal/sdn-intrusion-detection

[61] Ali, J., Song, H.H., Sharma, V. and Al-Khasawneh, M.A., 2024. DDoS intrusions detection in low power SD-IoT devices leveraging effective machine learning. IEEE Transactions on Consumer Electronics.

[62] Tian, Q., Han, D., Hsieh, M.Y., Li, K.C. and Castiglione, A., 2021. A two-stage intrusion detection approach for software-defined IoT networks. Soft Computing, 25, pp.10935-10951.

[63] El-Sayed, A., Said, W., Tolba, A., Alginahi, Y. and Toony, A.A., 2024. MP-GUARD: A novel multi-pronged intrusion detection and mitigation framework for scalable SD-IOT networks using cooperative monitoring, ensemble learning, and new P4-extracted feature set. Computers and Electrical Engineering, 118, p.109484.

[64] Shaji, N.S., Muthalagu, R. and Pawar, P.M., 2024. SD-IIDS: intelligent intrusion detection system for software-defined networks. Multimedia Tools and Applications, 83(4), pp.11077-11109.

[65] Kumar, P., Jolfaei, A. and Islam, A.N., 2025. An enhanced Deep-Learning empowered Threat-Hunting Framework for software-defined Internet of Things. Computers and Security, 148, p.104109.

[66] Mahmoud, M.M., Youssef, Y.O. and Abdel-Hamid, A.A., 2025. XI2S-IDS: An Explainable Intelligent 2-Stage Intrusion Detection System. Future Internet, 17(1), p.25.

[67] Olanrewaju-George, B. and Pranggono, B., 2025. Federated learning-based intrusion detection system for the internet of things using unsupervised and supervised deep learning models. Cyber Security and Applications, 3, p.100068.

[68] Khatami, S.S., Shoeibi, M., Oskouei, A.E., Martín, D. and Dashliboroun, M.K., 2025. 5DGWO-GAN: A Novel Five-Dimensional Gray Wolf Optimizer for Generative Adversarial Network-Enabled Intrusion Detection in IoT Systems. Computers, Materials and Continua, 82(1).

[69] Toony, A.A., Alqahtani, F., Alginahi, Y. and Said, W., 2024. MULTI-BLOCK: A novel ML-based intrusion detection framework for SDN-enabled IoT networks using new pyramidal structure. Internet of Things, 26, p.101231.

[70] Kong, X., Zhou, Y., Xiao, Y., Ye, X., Qi, H. and Liu, X., 2024. iDetector: A Novel Real-Time Intrusion Detection Solution for IoT Networks. IEEE Internet of Things Journal.

[71] Dinesh, K. and Santhosh Kumar, S.V.N., 2024. Energy-efficient trust-aware secured neuro-fuzzy clustering with sparrow search optimization in wireless sensor network. International Journal of Information Security, 23(1), pp.199-223.

[72] Mazzia, V., Salvetti, F. and Chiaberge, M., 2021. Efficient-capsnet: Capsule network with self-attention routing. Scientific reports, 11(1), p.14634.

[73] Senthilselvi, A., Prawin, R.P. and Harshit, V., 2024, July. Abstractive Summarization of YouTube Videos Using LaMini-Flan-T5 LLM. In 2024 Second International Conference on Advances in Information Technology (ICAIT) (Vol. 1, pp. 1-5). IEEE.

[74] "SD-IoT Intrusion Detection Dataset" [Online]. Available: https://www.kaggle.com/datasets/SD-IoT Intrusion Detection Dataset

[75] Lu, Z., Ding, C., Juefei-Xu, F., Boddeti, V.N., Wang, S. and Yang, Y., 2022. Tformer: A transmission-friendly vit model for iot devices. IEEE Transactions on Parallel and Distributed Systems, 34(2), pp.598-610.

[76] "CSE-CIC-IDS2018 datase" [Online]. Available: https://www.kaggle.com/datasets/CSE-CIC-IDS2018 datase

[77] McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.Y. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, 2017; Singh, A.; Zhu, J., Eds.; Volume 54, PMLR, Fort Lauderdale, FL, USA; pp.1273–1282.

[78] A. Sivaraman, M. Budiu, A. Cheung, C. Kim, S. Licking, G. Varghese, H. Balakrishnan, M. Alizadeh, and N. McKeown. "Packet transactions: High-level programming for line-rate switches." the ACM Special Interest Group on Data Communication, SIGCOMM'16 (ACM), 2016.

[79] A. Y. Ding, J. Crowcroft, S. Tarkoma, and H. Flinck. "Software defined networking for security enhancement in wireless mobile networks." (Comput. Netw) 66 (2014): 94–101.

[80] J. Hu, L. Shen, S. Albanie, G. Sun and E. Wu, "Squeeze-and-Excitation Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 8, pp. 2011-2023, 1 Aug. 2020, doi: 10.1109/TPAMI.2019.2913372.

[81] Bakhshi, Taimur. "State of the Art and Recent Research Advances in Software Defined Networking." Edited by Giovanni Pau. Wireless Communications and Mobile Computing (Hindawi) 2017 (Jan 2017): 35.

[82] Gude, N. "NOX: Towards an operating system for networks." ACM SIGCOMM Comput. Commun. Rev. 38 (July 2008): 105–110.

[83] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov. "Security in software defined networks: A survey." IEEE Communications Surveys and Tutorials 17, no. 4: 2317–2346.

[84] Jain, S. "B4: Experience with a globally-deployed software defined WAN." ACM SIGCOMM Conf. 2013. 3–14.

[85] K. Benton, L. J. Camp, and C. Small. "OpenFlow vulnerability assessment." in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Network, 2013: 151–152.

[86] M. Bonola, G. Bianchi, G. Picierro, S. Pontarelli, and M. Monaci. "StreaMon: a data-plane programming abstraction for software-defined stream monitoring." IEEE Transactions on Dependable and Secure Computing, no. 9 (2015).

[87] M. Canini, D. Venzano, P. Peresini, D. Kostic, and J. Rexford. "A NICE way to test OpenFlow applications." th USENIX Conf. Netw. Syst. Des. Implementation, 2012: 10.

[88] M. Canini, P. Kuznetsov, D. Levin, and S. Schmid. "A distributed and robust sdn control plane for transactional network updates." IEEE conference on computer communications, INFOCOM'15 (IEEE), 2015: 190–198.

[89] M. Liyanage, M. Ylianttila, and A. Gurtov. "Securing the control channel of software-defined mobile networks." IEEE 15th Int. Symp. World Wireless, Mobile Multimedia Network, 2014: 1–6.

[90] M. Tsugawa, A. Matsunaga, and J. A. Fortes. "Cloud computing security: What changes with software-defined networking?" in Secure Cloud Computing (Springer-Verlag) 77-93.

[91] Mai, H. "Debugging the data plane with anteater." ACM SIGCOMM Comput. Commun. Rev. 41 (2011): 290–301.

[92] McKeown, N. "OpenFlow: Enabling innovation in campus networks." ACM SIGCOMM Comput. Commun. Rev. 38 (April 2008): 69–74.

[93] N. P. Katta, J. Rexford, and D. Walker. "Logic programming for software-defined networks." ACM SIGPLAN Workshop Cross-Model Lang. Design Implement, 2012: 1-3.

[94] Okamura, M. M. O. Othman and K. " "Securing distributed control of software defined networks." Int. J. Comput. Sci. Netw. Security 13: 5–14.

[95] Open Network Foundation. "SDN architecture." June 2014.

[96] OpenDayligh. A Linux Foundation Collaborative Project. 2014. http://www.opendaylight.org.

[97] P. Kampanakis, H. Perros, and T. Beyene. "SDN-based solutions for moving target defense network protection." IEEE 15th Int. Symp. WoWMoM Network, 2014: 1-6.

[98] Snort. Open Source Intrusion Prevention System. https://www.snort.org.

[99] T. Xing, D. Huang, L. Xu, C.-J. Chung, and P. Khatkar. "SnortFlow: A OpenFlow-based intrusion prevention system in cloud environment." 2nd GENI GREE Workshop, 2013: 89–92.

[100] Singh C, Jain AK. "A Comprehensive Survey on DDoS Attacks Detection and Mitigation in SDN-IoT Network," e-Prime-Advances in Electrical Engineering, Electronics and Energy. 2024 Apr 9:100543.

[101] Garba UH, Toosi AN, Pasha MF, Khan S. "SDN-based detection and mitigation of DDoS attacks on smart homes," Computer Communications. 2024 May 1;221:29-41.

[102] Ali TE, Chong YW, Manickam S. "Comparison of ML/DL approaches for detecting DDoS attacks in SDN," A.SCI. 2023;13(5):3033.

[103] Khedr WI, Gouda AE, Mohamed ER. FMDADM: A multi-layer DDoS attack detection and mitigation framework using machine learning for stateful SDN-based IoT networks. IEEE Access. 2023 Mar 22;11:28934.

[104] Eliyan LF, Di Pietro R. DeMi: A Solution to Detect and Mitigate DoS Attacks in SDN. IEEE Access. 2023 Aug 4.

[105] Zhang T, Wang Y. RLFAT: A Transformer-Based Relay Link Forged Attack Detection Mechanism in SDN. Electronics. 2023 May 15;12(10):2247.

[106] Buzzio J, Vergara J, Ríos-Guiral S, Garzón C, Gutierrez S, Botero J, Quiroz J, Pérez J. "Introducing SDNFLow, a Comprehensive OpenFlow-Based Statistics Dataset for Attack Detection," 2024 IEEE Access. PP. 1-1. 10.1109/ACCESS.2024.3378271.

[107] Nisa N, Khan A, Ahmad J, Abdullah J, "Two-phase authentication system for denial of service attack detection and mitigation using machine learning in software-defined network". Int'l. Journal of Network Management. 2024 Jan 12:e2258.

[108] N. M. Yungaicela-Naula, C. Vargas-Rosales, J. A. Perez-Diaz, E. Jacob and C. Martinez-Cagnazzo, "Physical Assessment of an SDN-Based Security Framework for DDoS Attack Mitigation: Introducing the SDN-SlowRate-DDoS Dataset," IEEE Access, vol. 11, 2023.

[109] H. Elubeyd, D. Yiltas-Kaplan and Ş. Bahtiyar, "A Multi-Modal Deep Transfer Learning Framework for Attack Detection in Software-Defined Networks," IEEE Access, vol. 11, pp. 114128-114145, 2023.

[110] Ko K-M, Baek J-M, Seo B-S, Lee W-B. "Comparative Study of AI-Enabled DDoS Detection Technologies in SDN," 2023; 13(17):9488.

[111] A. Ahmim, F. Maazouzi, M. Ahmim, S. Namane and I. B. Dhaou, "Distributed Denial of Service Attack Detection for the Internet of Things Using Hybrid Deep Learning Model," in IEEE Access, vol. 11, pp. 119862-119875, 2023.

[112] F. Rustam, A. Raza, M. Qasim, S. K. Posa and A. D. Jurcut, "A Novel Approach for Real-Time Server-Based Attack Detection Using Meta-Learning," in IEEE Access, vol. 12, pp. 39614-39627, 2024, doi: 10.1109/ACCESS.2024.3375878.

[113] Halman LM, Alenazi MJ. MCAD: A Machine learning based cyberattacks detector in Software-Defined Networking (SDN) for healthcare systems. IEEE Access. 2023 Apr 13.

[114] Elubeyd H, Yiltas-Kaplan D. Hybrid deep learning approach for automatic Dos/DDoS attacks detection in software-defined networks. Applied Sciences. 2023 Mar 16;13(6):3828.

[115] Hnamte V, Hussain J. An efficient DDoS attack detection mechanism in SDN environment. International Journal of Information Technology. 2023 Jun;15(5):2623-36.

[116] Bhayo J, Shah S.A, Hameed S, Ahmed A, Nasir J, Draheim D. "Towards a machine learning-based framework for DDOS attack detection in software-defined IoT (SD-IoT) networks". Engineering Applications of Artificial Intelligence. 2023 Aug 1;123:106432.

[117] Gebremeskel TG, Gemeda KA, Krishna TG, Ramulu PJ. DDoS attack detection and classification using hybrid model for multicontroller SDN. Wireless Communications and Mobile Computing. 2023 Jun 23;2023.

[118] Wang J, Wang L, Wang R. A Method of DDoS Attack Detection and Mitigation for the Comprehensive Coordinated Protection of SDN Controllers. Entropy. 2023 Aug 14;25(8):1210.

[119] Tian Q, Miyata S. A DDoS Attack Detection Method Using Conditional Entropy Based on SDN Traffic. IoT. 2023 Apr 12;4(2):95-111.

[120] Chaganti R, Suliman W, Ravi V, Dua A. Deep Learning Approach for SDN-Enabled Intrusion Detection System in IoT Networks. Information. 2023; 14(1):41. https://doi.org/10.3390/info14010041

[121] Sicato, J.C.S., Singh, S.K., Rathore, S. and Park, J.H., 2020. A comprehensive analyses of intrusion detection system for IoT environment. Journal of Information Processing Systems, 16(4), pp.975-990.

[122] Kumar, C. and Ansari, M.S.A., 2024. An explainable nature-inspired cyber attack detection system in Software-Defined IoT applications. Expert Systems with Applications, 250, p.123853.

[123] Chauhan, P. and Atulkar, M., 2024. An Ensemble Edge Computing Approach for SD-IoT security Using Ensemble of Feature Selection Methods and Classification. Arabian Journal for Science and Engineering, pp.1-22.

[124] Kumbhar, K. and Mukherji, P., 2024. An optimized deep strategy for recognition and alleviation of DDoS attack in SD-IoT. Network: Computation in Neural Systems, pp.1-32.

[125] Devi, V.A., Raj, V.H., Kavin, B.P., Gangadevi, E., Balusamy, B. and Gite, S., 2024, February. An Atom Quest Optimizer for CNN to Distinguish IDs in SDN and IoT Eco System. In 2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT) (Vol. 5, pp. 1430-1437). IEEE.

[126] Arthi, R., Krishnaveni, S. and Zeadally, S., 2024. An intelligent SDN-IoT enabled intrusion detection system for healthcare systems using a hybrid deep learning and machine learning approach. China Communications.

[127] Altamimi, S. and Abu Al-Haija, Q., 2024. Maximizing intrusion detection efficiency for IoT networks using extreme learning machine. Discover Internet of Things, 4(1), p.5.

[128] Yu, S., Wang, X., Shen, Y., Wu, G., Yu, S. and Shen, S., 2024. Novel Intrusion Detection Strategies With Optimal Hyper Parameters for Industrial Internet of Things Based On Stochastic Games and Double Deep Q-Networks. IEEE Internet of Things Journal.

[129] Azimjonov, J. and Kim, T., 2024. Stochastic gradient descent classifier-based lightweight intrusion detection systems using the efficient feature subsets of datasets. Expert Systems with Applications, 237, p.121493.

[130] Jyothsna, V., 2024, April. Defending Against IoT Threats: A Comprehensive Framework with Advanced Models and Real-Time Threat Intelligence for DDoS Detection. In 2024 2nd International Conference on Networking and Communications (ICNWC) (pp. 1-7). IEEE.

[131] Palaniappan, K., Duraipandi, B. and Balasubramanian, U.M., 2024. Dynamic behavioral profiling for anomaly detection in software-defined IoT networks: A machine learning approach. Peer-to-Peer Networking and Applications, 17(4), pp.2450-2469.

[132] Singh, S.K., Kumar, M., Khanna, A. and Virdee, B., 2024. Blockchain and FL-based secure architecture for enhanced external Intrusion detection in smart farming. IEEE Internet of Things Journal.

[133] Gheni, H.Q. and Al-Yaseen, W.L., 2024. Two-step data clustering for improved intrusion detection system using CICIoT2023 dataset. e-Prime-Advances in Electrical Engineering, Electronics and Energy, 9, p.100673.

[134] Bella, H.K. and Vasundra, S., 2024. A Novel Framework based on Extra Tree Regression Classifier and Grid Search LSTM for Intrusion Detection in IoT and Cloud Environment. International Journal of Intelligent Engineering and Systems, 17(4).

[135] Olanrewaju-George, B. and Pranggono, B., 2025. Federated learning-based intrusion detection system for the internet of things using unsupervised and supervised deep learning models. Cyber Security and Applications, 3, p.100068.

[136] W. El Gadal and S. Ganti, "Federated Secure Intelligent Intrusion Detection and Mitigation Framework for SD-IoT Networks using ViT-GraphSAGE and Automated Attack Reporting, Paris, France, 2025.