Arab American University

Faculty of Graduate Studies

Flood Attack Detection System with IAT As Main Feature Using Machine Learning

By

Nima Mohammad Lutfi Khalil

Supervisor

Dr. Huthaifa Ashqar

Co- Supervisor

Dr. Omar Darwish

This thesis was submitted in partial fulfillment of the requirements for the Master`s degree in Cyber Security

March /2024

Thesis Approval

Flood Attack Detection System with IAT As Main Feature Using Machine Learning

By

Nima Mohammad Lutfi Khalil

This thesis was defended successfully on March 9, 2024 and approved by:

Committee members	Signature
1. Dr. Huthaifa Ashqar: Main Supervisor	
2. Dr. Omar Darwish: Co- Supervisor	
3. Dr. Mohammad Hamarsheh: Internal Examiner	
4. Dr Mohammed Elhenawy: External Examiner	Mohammed Elhenawy

Declaration

I declare that, except where explicit reference is made to the contribution of others, this dissertation is substantially my own work and has not been submitted for any other degree at the Arab American University or any other institution.

Student Name: Nima Mohammad Lutfi Khalil

Student ID: 202112787

Signature: Nima Khalil

Date: September 4,2024

Dedication

With all my love and appreciation, I dedicate this thesis to the soul of my dear father Mohammad Lutfi Khalil Terawi “Abo Lutfi” Who though no longer with us, continues to inspire and guide me His memory and influence remain a guiding light in my journey.

Acknowledgement

I would like to thank my supervisors, Dr. Huthaifa Ashqar and Dr. Omar Darwish, for their patient guidance. And for helping me stay on track with my success with their guidance and support.

My family has provided me with constant support and encourage during my years of education, for which I am incredibly grateful. Without them, this achievement would not have been feasible.

I extend my special thanks to Dr. Mohammad Hamarsheh for their invaluable support in the Network part of my project.

I wish to thank everyone who was involved in the making of this thesis

Abstract

According to the explosion of Cloud Computing domain, Network Intrusion Detection systems or IDSs, also Arbor adaptive DDoS devices are essential for cyber security. When it comes to the feature phase, the network traffic could contain a range of components, including host information, malicious scripts, attack subcategories, and reference attack types (Seo, 2020), When compared to regular traffic, In terms of network phase, there may be an unequal distribution of destructive attacks in network traffic.

Regarding cybersecurity, the impact of DoS attacks on the CIA triad is crucial to consider. Attacks that cause a denial of service, by their nature, poses a significant threat to the Availability aspect of the triad. These attacks aim to overwhelm and exhaust the resources of a network or system, calling it unavailable to legitimate users, these attacks may have serious effects. leading to service disruptions, downtime, and potential financial losses. Therefore, as we explore and evaluate various intrusion detection methods, understanding and mitigating the impact of DoS attacks on the availability of network resources becomes paramount in addition to trying to act as fast as possible to network threats, an automatic detection system is needed. Network administrators have less time to modify their detection and repairs systems and update their signatures when an attack is discovered later.

It is challenging to identify different types of DoS attacks passing through encrypted network channels due to complex features and the similarity between many types of DoS.

This thesis aims to address the behavior of DOS attacks based on the interval time between packets in the network environment, by creating a Simulink environment and evaluate the behaviors of various types of DoS attacks to meet the requirements of a good IDS.

To make the following experiments, First, we set up a Simulink cloud environment using the Mininet platform and create different types of Dos attack, second capture the network traffic and calculate statistical features, Interval between arrivals of typical network traffic and Dos traffic, third we apply Hypered RNN+ LSTM, DNN+ LSTM, Random Forest and some traditional algorithms as SVM (Support vector machine), LG (Logistic Regression) over the created dataset and evaluated to address high score metrics, forth the synthetic minority oversampling technique (SMOTE) is implemented in the dataset to fix the imbalance dataset issue, Finally, compare the evaluation results

Keywords: Mininet, Dos Flooding, Inter arrival time, Random Forest algorithm.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

IAT	Inter arrival Time
PK	Network Packet
SMOTE	Synthetic Minority Over-sampling Technique
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
SVM	Support Vector Machine
ROC – curve	Receiver Operating Characteristic curve
TP	True Positive
FN	False Negative

Chapter one Introduction

Chapter Outline

1. Chapter One

1.1 Introduction

Background

Since their inception, computers and networks have been vulnerable to various threats such as viruses, worms, and hacker attacks. A significant milestone was reached in 2008 when the quantity of internet-connected devices surpassed the global population, a trend that has continued to escalate. By 2020, it was projected that around 50 billion devices would be having an internet connection. As this number increases, safeguarding these devices and the data they exchange has become increasingly challenging, particularly as the frequency of security breaches continues to rise annually (K, 2012).

In recent times, numerous organizations have adopted diverse methods for data storage, with their primary objective being the protection and security of their confidential and official information against unauthorized access and external threats. There is also the risk that authorized users might leak sensitive data for various reasons. Identifying the source of an attack in real-time poses a significant challenge due to the possibility of attackers using duplicated IP addresses and crafting deceptive attack packets. Standard security methods like firewalls and Intrusion Prevention Detection Systems (IPDS) have often fallen short in pinpointing the attackers. Broadly, a computer network is a synergy of hardware and software, each with its own set of risks, vulnerabilities, and security concerns. Attacks targeting software can compromise data integrity.

Individuals with a background in programming and systems are usually able to monitor and understand system activities through log files. However, the issue becomes more pronounced for those lacking programming knowledge. When their systems are breached by intruders, they find themselves ill-equipped to identify or address the problem (Hidayat et al., 2022).

Additionally, one of the most challenging aspects of network security is identifying insider attacks. In the field of network security, every user seeks to safeguard their systems from all forms of malicious activities, whether they originate from within or outside the organization. An Intrusion Detection System (IDS) plays a crucial role in this context. It is capable of detecting attacks from external intruders and is also equipped to identify internal threats. Essentially, an IDS is a system designed to monitor network traffic, flagging any suspicious activities and issuing alerts upon their detection(Seo, 2020), Denial of Service (DoS) attacks have become a significant threat to computer networks. At the close of the twentieth century, these attacks were known for disrupting major websites, including prominent ones like Amazon, CNN, and Yahoo, leading to their temporary shutdown (Al-Morjan, 2010).

The attacker executed the DoS attack by inundating the target network with an overwhelming number of packets. This tactic exploits the network's inherent properties, such as bandwidth and computing power, rendering it incapable of functioning normally. As a result, legitimate clients are unable to access services from the affected network, leading to a significant deterioration in its performance (Rashid et al., 2020).

2.1 Motivation

The growing popularity of cloud and technologies in recent years has changed digital services, offering unparalleled scalability and flexibility. However, this surge in cloud usage has also led to a concerning rise in sophisticated cyber threats, particularly Denial of Service (DoS) attacks. These attacks address a serious risk to the availability and reliability of cloud-based services, lead the development of innovative security measures.

Traditional methods of DoS detection often struggle to adapt to the dynamic nature of cloud environments, prompting the investigation of new strategies. One method is the analysis of Inter Arrival Time (IAT), which refers to the time intervals between successive network packets or requests. This temporal dimension holds valuable information about network traffic patterns, with anomalies in Inter Arrival Time (IAT) potentially indicating malicious activities.

The motivation for this thesis arises from recognizing that IAT has untapped potential as a relevant feature for enhancing DoS detection systems. inspecting the timing between network events can reveal subtle deviations in behavior that may signal the onset of a DoS attack. Combining machine learning techniques with IAT analysis offers an intelligent and adaptive security approach, capable of learning normal patterns and swiftly identifying anomalies.

Addressing the issue that DoS attacks confuse to network systems, particularly in encrypted communication, is crucial. The increased difficulty in detecting and stopping these attacks within encrypted traffic highlights the need for methods that employ inter-

arrival time and statistical features of network traffic. This promising approach can discern abnormal patterns linked with DoS attacks.

3.1 Research Gap

The research gap highlighted the need for effective and adaptive security measures, particularly in the context of Denial of Service (DoS) attacks. Traditional methods of DoS detection are acknowledged to struggle in adapting to the dynamic nature of cloud platforms. This recognition leads to the exploration of novel approaches, with a specific emphasis on the analysis of Inter Arrival Time (IAT) as a potentially valuable feature.

The research identifies a gap in understanding and addressing the security challenges confused by the surge in cloud adoption. While cloud computing offers unparalleled scalability and flexibility, it also introduces new vulnerabilities that traditional security measures may not effectively address. The exploration of IAT as a relevant feature aims to close this distance by providing a more nuanced and adaptive approach to DoS detection.

Moreover, the research acknowledges the increasing difficulty in detecting and stopping DoS attacks adding another layer to the identified gap. The need for methods that utilize inter-arrival time and statistical features of network traffic, this highlights a specific gap in existing security measures, particularly in the context of evolving cyber threats and the protective measures required for preserving service availability.

In essence, the identified research gap lies in the inadequacy of traditional security methods to effectively address Cloud areas' dynamic nature and the specific challenges

posed by DoS attacks. The exploration of IAT and the incorporation of methods for machine learning aim to contribute a more adaptive and sophisticated tool for enhancing the resilience of cloud environments against the escalating threat of DoS attacks.

4.1 Problem Statement

The growing number of Denial-of-Service (DoS) attacks, calls for the development of stronger detection techniques. Creative methods are needed to solve this urgent problem, such as examining the statistical characteristics of data transmission and closely examining the time intervals between network activities. By the use of these techniques, patterns matching a denial-of-service attack can be identified, supporting the effectiveness of machine learning models in quickly spotting and removing such threats and protecting the security and integrity of network infrastructures.

5.1 Research Questions

1- How do inter-arrival times vary in normal network traffic, and what statistical features characterize their distribution?

2- What are the distinctive inter-arrival time patterns indicative of a Denial-of-Service (DoS) attack, and how can statistical analysis enhance the identification of these patterns?

3. Can machine learning models effectively leverage inter-arrival time and statistical features to differentiate between normal network behavior and potential DoS attacks?

4. What effects does the size of the dataset have on the statistical characteristics and inter-arrival time-based DoS detection accuracy and reliability?

5. What is the optimal sampling rate for capturing inter-arrival time patterns and statistical features to enhance DoS detection accuracy without excessive resource consumption?

6. What is the impact of network scale and traffic volume on the scalability and efficiency of DoS detection systems utilizing inter-arrival time and statistical features?

7. Can the combination of inter-arrival time and statistical features provide insights into the evolution and progression of DoS attacks, aiding in the development of proactive defense strategies?

6.1 Objectives

This thesis aims to bridge a critical gap in understanding cloud security by exploring the significance of Inter Arrival Time (IAT) as a relevant feature. Through this exploration, the research struggle to contribute a valuable and adaptive tool to the arsenal of cybersecurity measures, fortifying the resilience of cloud environments against the escalating threat of DoS attacks.

In order to address the research inquiries, the subsequent goals have been established:

Objective 1 characterize Inter-Arrival Times in Normal Network Traffic

Doing this by identify variations in inter-arrival times within normal network traffic also define the statistical features that explain the inter-arrival time distribution.

Objective 2 Identify Distinctive Patterns for DoS Attacks using Inter-Arrival Time

Employ statistical analysis techniques to enhance the identification of these patterns during a potential DoS attack

Objective 3 Develop and evaluate Deep Learning Model

Design machine learning models capable of leveraging inter-arrival time and statistical features, and measure how well the machine learning models work in differentiating between normal network behavior and potential DoS attacks

7. 1 Contribution:

The contributions of the thesis are the following:

1- Create Real data include DoS attacks in Simulink environment

2- Implement classic and deep learning models for comparison.

3- Study the impact of Inter-arrival time and statistical features to identify the DoS flooding in the network traffic.

4- Run The models on different size of datasets and compare the results.

5- Benchmark best models against existing IDSs

8. 1 Summary

This chapter gives a summary of why it's important to protect network systems and the sensitive information of organizations. It goes on to define the problem Statement that has been addressed. Also talks about why using deep learning algorithms for building Intrusion Detection Systems (IDS) is a good idea. Then, it sets out the goals of the research, presents a research question that the study aims to answer, and lists the ways in which this research adds the current understanding in the domain.

Chapter Two: IDS System

Chapter Outline

2.1 Background

- 2.1.1 Software-Defined Networking
- 2.1.2 Mininet tool
- 2.1.3 Intrusion Detection System
- 2.1.4 Capturing Network Traffic Tools

2.1.5 Denial of Service Types

- 2.1.6 Signature-based Detection
- 2.1.7 Machine Learning
- 2.1.8 Deep Learning
- 2.1.9 Intrusion Detection Dataset

2.2 Literature reviews

2.3 Summary

---

2. Chapter Two Software-Defined Networking

2.1 Background

2.1.1 Software-Defined Networking

The emergence of Cloud Computing has opened up innovative and promising possibilities for individuals, small to medium-sized businesses, and large corporations alike. It gives them the ability to use network, processing, and storage resources as needed, tailored to their varying needs over time. (Cziva et al., 2016), The development of multi-tenancy and virtualization has completely changed how computing resources are distributed, allowing for their shared use among multiple users. This approach eliminates the need for fixed resource allocation and addresses the issue of underutilized servers. Virtual Machines (VMs) play a crucial role in this environment, serving as key components that encapsulate operating systems and abstract them from the physical hardware on which they run. This setup not only enhances application performance for users but also ensures efficient utilization of the provider's physical resources, helping to resolve potential bottlenecks. A particularly significant technique in this context is live VM migration, which is primarily used to optimize server-side resource consumption (like CPU, RAM, and I/O) and to minimize power usage during operations(Qureshi & Braun, 2019).

While server-side metrics can be used to minimize the number of servers that must be powered on at any given time and ensure that server resources are fully utilized, they do not account for the network congestion that results from this. Recent research

indicates that machine virtualization can negatively affect cloud environments, leading to dramatic variations in performance and cost that are primarily related to networking rather than software bottlenecks.

To Clarify specific network behaviors necessitates the individual configuration of devices, and introducing new functionalities or policies requires these devices to be re-configured. Traditional IP networks lack flexibility, and their static nature hinders their ability to adapt and evolve over time (Qureshi & Braun, 2019).

The change in computer network architecture is represented by software-defined networking (SDN), moving away from traditional network configurations. It divides the network's control plane from the data-plane, centralizing control in a singular controller. This controller is equipped to dynamically program and manage the network.

Most SDN controllers, including Open Daylight, Ryu, POX, and Floodlight, provide APIs that enable the configuration of network components, management of firewalls, and monitoring of traffic counters. These controllers have been extensively utilized in various network-related projects, such as full network migration, development of new management interfaces, Quality of Service (QoS) management, and participatory networking (Mousavi & St-Hilaire, 2015).

Software-defined networking (SDN) provides a powerful transition platform from traditional networks to a flexible, programmable, and open paradigm. The continuous growth of the internet, the emergence of cloud computing, and the widespread adoption

of social media necessitate a reliable, adaptable, and scalable network infrastructure. The increasing complexity of both public and private network traffic poses challenges during implementation, highlighting the need for programmability and a centrally managed network architecture, as its name implies, SDN is a computer network that is implemented and managed centrally through software programming. By interconnecting devices, SDN facilitates global communication, enabling seamless interaction across the globe. However, this extensive interconnectivity also presents challenges that can potentially disrupt the entire network. SDN's decoupled architecture and programmability capabilities position it as the frontrunner in the networking world, as it allows for centralized management from a single point of control (Naveed et al., 2022).

Work of Software-defined networking (SDN)

Through a secured channel, the OpenFlow protocol enables safe communication between the controller and OpenFlow switches. (Mousavi & St-Hilaire, 2015), When a new packet reaches the switch, it checks its tables for a match. If there's a match, the packet is forwarded based on the table instructions. If there's no match, the packet goes to the controller, which decides whether to create a new rule for similar packets to be forwarded or add a rule to discard them.

Experimenting with new SDN (Software-Defined Networking) and OpenFlow concepts using actual hardware is both difficult and expensive. While using a network of virtual machines (VMs) might be considered, this approach often proves to be too cumbersome. Due to these challenges, researchers are increasingly turning to Mininet

an emulator known for its efficiency. Mininet provides a rapid prototyping environment that features lightweight virtualization, along with extensive support for both command-line interface (CLI) and application programming interface (API) (Dao et al., 2017). Developed with key features such as flexibility, scalability, interactivity, and shareability, Mininet stands out in comparison to other prototyping environments that often lack these attributes.

It offers a straightforward and user-friendly approach to real-world systems, ensuring that a prototype created in Mininet can be deployed on actual hardware without necessitating any modifications to the code or configuration(Abdullah, 2014), Mininet includes a GUI editor known as Miniedit, an experimental tool designed for rapid creation and execution of network topologies. This utility facilitates the testing of various configurations by allowing users to easily modify the specifications of different components within the network (Dao et al., 2017).

Mininet is versatile enough to support a wide range of tasks essential for research, development, and education. Built on the Linux kernel version 2.2.26, it employs process-based virtualization to simulate individual processes of virtual hosts, switches, controllers, and links. This setup allows real code to be executed without any alterations. Furthermore, Mininet simplifies the process of setting up a complete network testbed, enabling easy installation within virtual machine environments like VMware or VirtualBox, across various operating systems including Mac, Windows, and Linux (Dao et al., 2017).

2.1.3 **Intrusion Detection Systems**

The purpose of an intrusion detection system (IDS) is to describe attack manifestations in order to positively identify all actual attacks without incorrectly identifying none. There are a variety of reasons to utilize ID technology in order to keep information systems safe (Mchugh et al., 2000).

This system helps catch intruders who are trying to get into the network without permission (Amin et al., 2021), If someone messes with how the system is set up, it can start acting strangely. So, it's important to regularly update and monitor the system to see if anything unusual is happening.

Intrusion Detection Systems (IDS) come in two main types: one that directly checks specific devices and another that looks at the whole network, The initial kind is known as Host-Based IDS. (HIDS), which keeps an eye on how an individual computer system is doing. The second type is Network Intrusion Detection System (NIDS), which monitor the live traffic on a network to catch any unwanted intrusions using special detection methods. Additionally, IDS can be grouped based on what they're looking for, like misuse, system problems, or a mix of both, known as hybrid IDS.

2.1.2 Network Intrusion Detection System (NIDS):

NIDS is designed to monitor and analyze the traffic on a computer network, its primary goal is to identify any unusual or suspicious activities that may indicate an intrusion or security threat (Dhanya et al., 2023).

To keep files, secure in a network, different hashing algorithms like MD5 are used. When a network-based Intrusion Detection System (NIDS) detects a potential attack, it notifies administrators. The IDS examines network traffic for patterns that suggest an attack, such as a large number of related items indicating a possible denial-of-service attack, or a sequence of related packets suggesting a port scan. NIDSs are placed at specific points in the network, like routers, to monitor incoming and outgoing traffic in a particular segment or to watch over specific host computers. They can also be set up to keep an eye on every system-to-system communication within a network (Sayyed et al., 2023).

2.1.3 Host Based Intrusion Detection System (HIDS):

A Host-Based Intrusion Detection System (HIDS) is like a security guard specifically assigned to one computer or server, called the host. It keeps an eye on what's happening only on that system. There are two types of HIDS: one that looks for known patterns of attacks (signature-based) and another that watches for unusual behavior (anomaly-based). HIDS pays attention to important system files and can tell if someone tries to add, change, or delete these files. It sends an alert if it notices changes like file attributes being modified, new files being made, or existing files being removed. The big difference from Network IDS is that HIDS focuses on what's happening inside a single computer and can't see encrypted information as it travels through the network (Dhanya et al., 2023).

A Host-Based Intrusion Detection System (HIDS) can spot things happening on a computer and catch attacks that might slip past network-based security.

Signature-based Detection Technique

Several cybersecurity systems employ signature-based detection as a technique for network intrusion detection. particularly in intrusion detection systems (IDS) and antivirus software, to identify and block known threats, It's akin to using a "most wanted" list to catch known criminals (Akyildirim et al., 2022), When inspecting files, network traffic, or system behaviors, the detection system compares the incoming data against the signature database. If a piece of data or behavior matches a known signature, it's flagged as a potential threat.

Signature-based detection is really good at finding and stopping known computer viruses and hacking attempts because it just compares data to known threat patterns, making it quick and not too demanding on a computer's resources. However, it's not perfect. It struggles to identify brand new or very cleverly disguised threats that it hasn't seen before. It also needs regular updates to know about the latest threats. Sometimes, it can mistakenly think harmless software or internet activity is harmful if it looks similar to something in its database of known threats (Bul’ajoul et al, 2019).

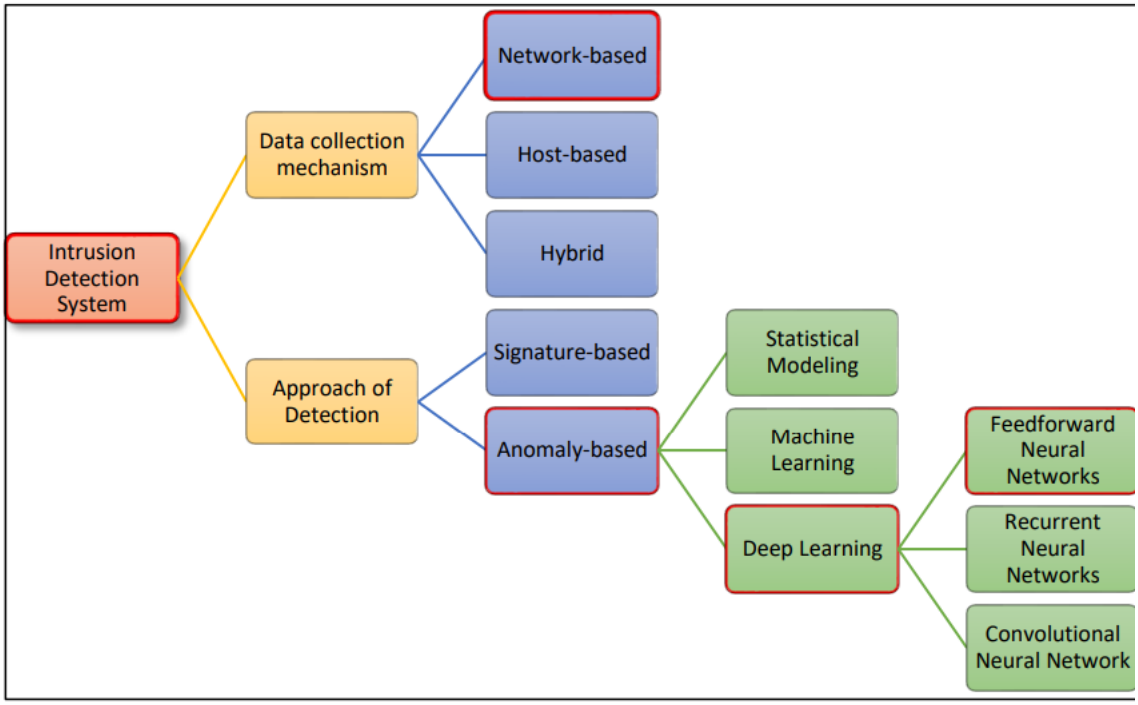


Figure 1 Categorizing of intrusion detection systems

2.1.4 Capturing Network Traffic

Organizations must gather and maintain records of network traffic in order to monitor, evaluate, and assess their network configurations (Ribeiro et al., 2016), There are two primary methods used in this process: first, full packet capture, and second, flow summarization. Although packet capture provides all network traffic, it isn't always feasible since high-capacity storage is required to record even a small quantity of traffic; the bulk of this data presents challenges for analysis and raises privacy and security issues. To record and analyze network traffic, network administrators and security analysts use a variety of technologies and programs.

To record and examine the entire payload of packets, tools such as Wireshark and Tcpdump, Capturing and storing the full data of network packets to check for intrusions is tough, especially in big networks, because there's a lot of traffic. Also, there's a concern about privacy and keeping information safe, even these packets often have encrypted data.

A network flow is defined as a series of packets moving between two points that share certain characteristics. This flow of packets can be either one-way or two-way. The shared attributes typically include source and destination IP addresses, ports at the transport layer (IPFIX), although there are many different ways to represent network flows, the NetFlow is the accepted industry standard.

Cisco Systems created the NetFlow network protocol as a tool for tracking and gathering IP traffic data. When it's enabled on network devices like routers and switches, NetFlow records data about the traffic passing through these devices. Sensitive Details contained in these packets include protocol type, port numbers, source and destination IP addresses, and, The amount of data transferred, the start and end time of the traffic flow, this information is valuable for network administrators because it helps them understand traffic patterns, network usage, and bandwidth consumption (Awad et al., 2022). It's used for various purposes such as network performance monitoring, traffic analysis, and identifying unexpected connections of traffic that could indicate security threats like DDoS attacks or network intrusions.

Grasping the nature and variety of attacks that can impact systems and networks is essential. By examining captured packets, one can discern attack patterns, aiding in the identification and remediation of harm inflicted by attackers. In this study, we employed Wireshark, an open-source packet analyzer, to detect potential network attacks. This was achieved by analyzing a set of trace files generated in real-world networked environments, providing valuable insights into security breaches. (Ndatinya et al., 2015), You don't have to be an expert to analyze network traffic, thanks to powerful tools that simplify this task. Among the various network traffic analyzers available, Wireshark stands out as one of the most effective and user-friendly software options for network analysis.

Wireshark is like a multi-tool for network analysis within a single application. It helps in examining your network traffic to spot possible setup mistakes and security breaches. Wireshark can recognize various forms of data packaging and can separate and show every part of a network packet. Despite its wide range of functions, learning how to use Wireshark might seem challenging (Ndatinya et al., 2015).

The traffic gathered using Wireshark include the following aspect:

1- Number of Packet:

2- Time: when the packet has captured

3- Src IP

4- Dest IP

5- Protocol

6- Len of Packet

7- Info

2.1.5 Denial of Service (DoS) Attacks:

Network attacks aim to disrupt and overwhelm legitimate traffic, thereby preventing users from accessing sensitive data. Attacks fall into one of two primary categories: "Passive" refers to when a network intruder captures data moving through the network, while "Active" involves an intruder sending commands to interfere with the network's regular functioning (Pawar & Anuradha, 2015), In this section we will present some of the types of DoS attacks.

Distributed denial of service (DDoS) assaults are among the most dangerous current threats (Alashhab et al., 2022), According to reference (Hussain & Nashat, 2022), Denial of Service (DoS) attacks have become one of the most significant threats to computer networks. These attacks had a significant effect, resulting in the closure of well-known websites at the end of the 20th century, including Yahoo, CNN, and Amazon. (Hussain & Nashat, 2022) In a Denial of Service (DoS) attack, the attacker overwhelms the target with an enormous quantity of packets, effectively flooding the network. This exploit leverages the network's own resources, such as bandwidth and computing power, As a result, legitimate clients are unable to receive services from the affected network, leading to a significant deterioration in network performance (Catak & Mustacoglu, 2019).

Multiple forms of Denial of Service (DoS) attacks are prevalent, with the primary objective being to saturate a client's network capacity or connectivity. Bandwidth focused attacks aim to degrade network efficiency by consuming all available

bandwidth, leading to significant delays or preventing the processing of legitimate user requests. On the other hand, connectivity-focused attacks inundate the victim's server with an excessive number of application-layer requests, depleting all its resources. Consequently, the server becomes incapable of responding to legitimate requests from users.

DoS attacks come in different forms: (i) Flood attacks, where the attacker sends too many packets to the victim, using up their resources and overloading their internet connection. (ii) Vulnerability attacks, where the attacker takes advantage of a flaw in the victim's system, sending specific messages that cause it to crash. Also, Distributed Denial of Service (DDoS) attacks involve many sources bombarding the target with a huge amount of traffic. Figure 8 shows how these flood-based DoS attacks work.

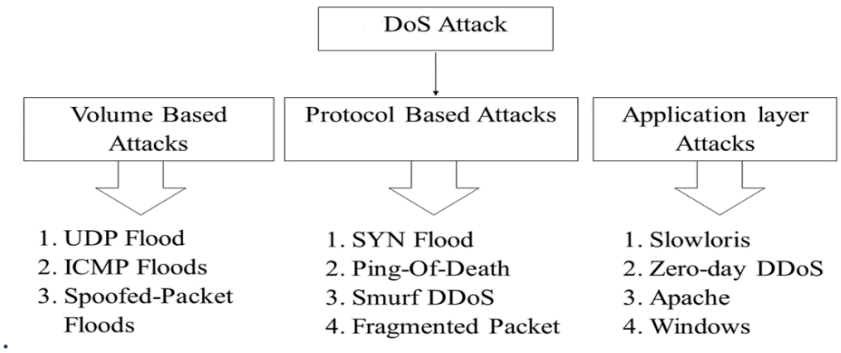


Figure 2 Flood DoS Types (Pawar & Anuradha, 2015)

DOS attack classified into three categories:

1- Volume based DDOS attack

2- Protocol based attack

3- Application Layer based attack

Volume based DDOS attack

Volumetric DDoS attacks involve flooding the target server device with an exceptionally high volume of traffic or request packets, causing a flood effect that can slow down or halt their services. Various types of volume-based attacks will be discussed further

UDP Flood

A UDP flood attack is a type of Distributed Denial of Service (DDoS) attack occurs when an attacker attacks a target port with a huge number of User Datagram Protocol (UDP) packets, overwhelming the server and making it impossible to reply to valid requests. As a result, the target system stops responding (Y. Wang et al., 2023)**.**

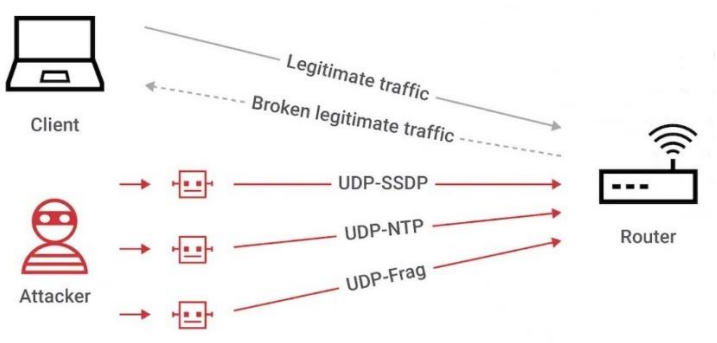


Figure 3 UDP Flood

2.1.6 ICMP Flooding Attack

The ICMP flooding DDoS (Distributed Denial of Service) attack presents a significant security challenge in networks that operate on IPv6. During such an attack, the perpetrator bombards the victim's network with an overwhelming number of packets in a brief timeframe. This deluge of traffic exhausts the network's bandwidth and resources. As the victim's system attempts to rapidly process these

incoming packets, it becomes overloaded and eventually incapacitated, leading to impaired functionality (Elejla et al., 2022).

Spoofing ICMP messages is a usual strategy used by attackers to avoid detection by intrusion detection systems or network administrators. Additionally, they might send a deluge of ICMPv6 error messages to disrupt the functionality of hosts and routers. The same outcome can be reached by using ICMPv6 informational messages, such as neighbor solicitation (NS), echo request and reply, and router advertisement (RA) messages. Figure 11 demonstrates the mechanism of an ICMP flooding attack and its operational dynamics.

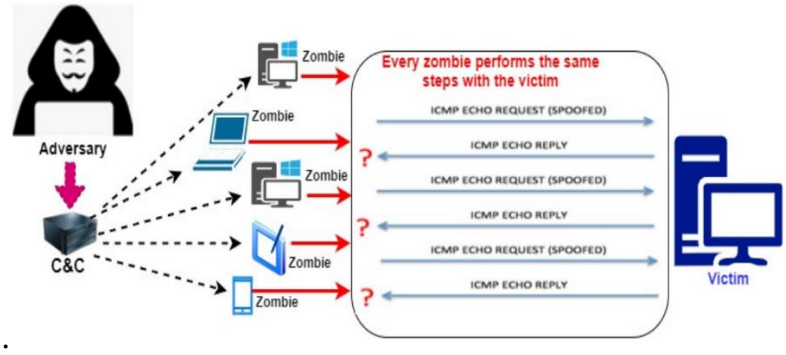


Figure 4 ICMP Flood Attack

Spoofed Packet Flood

A Spoofed Packet Flood, also known as a spoofed flood attack or simply a spoofed flood, is a type of DoS attacks where the attacker sends a large volume of network packets to a target system, each containing a falsified or spoofed source IP address. This flooding of packets overwhelms the target system's resources, causing it to become slow, unresponsive, or even crash.

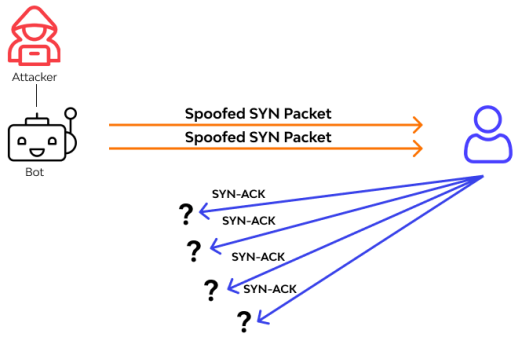


Figure 5 Spoofed Flood

Protocol Attacks

Protocol attacks target and consume the resources of actual servers and network communication equipment, such as load balancers (LB) and protective devices. The intensity of a protocol attack is typically quantified in terms of packets per second (Catak & Mustacoglu, 2019), SMURF, SYN flood, and ping of death are a few instances of protocol attacks.

SYN flooding attack

Initialization of a Transmission Control Protocol connection occurs using a three-way handshake procedure. A user sends a connection request (a SYN message) to a server. The server responds by sending back a confirmation and request for acknowledgment (a SYN-ACK message) to the user. Lastly, the user sends a msg (ACK message) back to the server to start the connection, However, in a SYN flooding attack, an attacker sends a lot of SYN messages to the server but doesn't complete the handshake by sending the final ACK message. Because the server is waiting to finish these

incomplete handshakes, it gets overwhelmed and can't handle requests from legitimate users (Darwish et al., 2013).

As shown in figure 10, In a SYN flooding attack, packets are sent with a fake IP address. This is similar to a sniffing attack, where the attacker guesses the sequence number of an ongoing TCP connection and sends a packet with this number, but with a fake IP. As a result, the server can't respond properly to these requests, which strains the resources and performance of the cloud system.

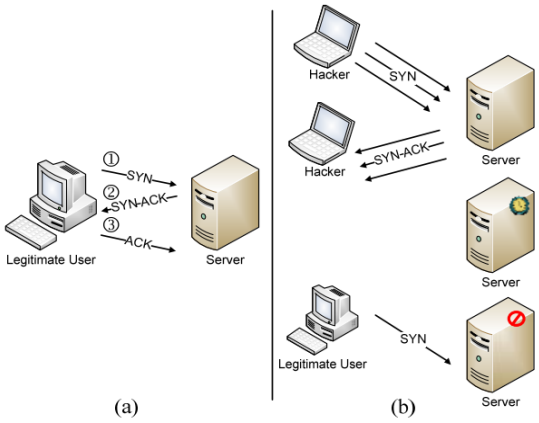


Figure 6 A: Normal B: Attack

Ping of Death

The attacker sends special packets, called malformed or oversized packets, using a simple ping command. This kind of attack aims to crash or freeze the computer or server that's being targeted. One specific type of attack, called the "Ping of Death," involves sending IP packets larger than 65,536 bytes to the target server. This attack, also known as a teardrop attack, can affect both ICMP and TCP protocols and is one of the most damaging ICMP attacks. It can even target other protocols like UDP.

Normally, a ping command can send data packets up to 65,500 bytes in size. If you try to send data larger than 64,500 bytes, the ping command will display an error message. The problem lies in the fact that the IPv4 header, a part of the packet, has a 16-bit field. The highest possible value for a 16-bit binary number is 65,535. If an ICMP echo request packet larger than 65,535 bytes is sent in an IPv4 datagram using the ping command, it can lead to memory overflow on the receiving end, potentially causing the target machine to freeze or crash. This mechanism of the Ping of Death attack is explained in detail in Figure 13 (Raja et al., 2022).

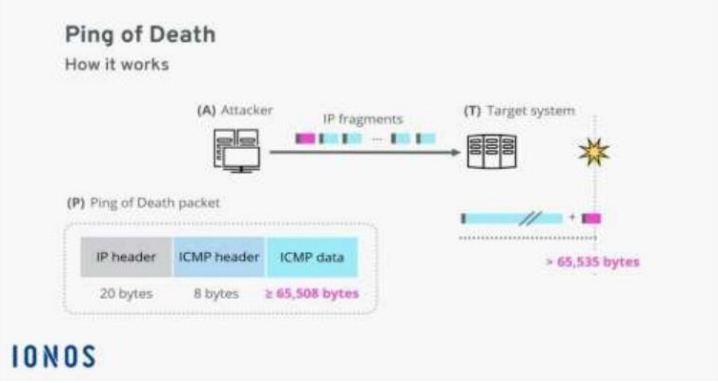


Figure 7 Ping of Death

SMURF Attack

The concept of an amplification attack was first implemented using ICMP in what's known as the Smurf Attack. This technique focuses on increasing the quantity of sent packets, rather than increasing the size of each packet (Dewidar et al., n.d.), the Smurf attack uses the ICMP echo broadcast where an echo request to a broadcast address with the spoofed IP of a victim would generate a large number of packets from the amplifier network to the victim, Consequently, the target is overwhelmed with response messages. The situation worsens if too many hosts respond to the

ICMP echo requests. Stopping this kind of attack can be challenging (Darwish et al., 2013).

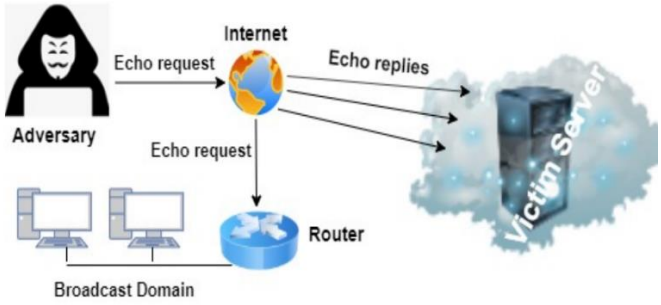


Figure 8 The Topology of Smurf Attack

Fragmented Packet

A Fragmented Packet Flood is a type of cyber-attack that involves sending a massive volume of IP packets that are fragmented to the target system or network. This method is particularly insidious because fragmented packets are smaller pieces of a larger packet that need to be reassembled by the target system's networking stack to be processed correctly. This type of attack exploits the way that Internet Protocol (IP) allows for large packets to be broken down into smaller fragments for efficient routing and then reassembled at the destination (Ashrif et al., 2024).

Application layer attack

Application layer attacks called layer 7 attack, were particularly impactful in Q4 2021. The manufacturing industry experienced the highest impact, with a staggering 64% increase in attacks compared to the previous quarter. Following closely, the business services and gambling sectors were also significantly affected. Notably, China led the charts for the fourth time in the year with the highest percentage of attacks originating from Application Layer attacks. Additionally, a new botnet called Meris was identified in mid-2021, capable of launching attacks with an alarming rate of 17.5 million requests per second against servers.

Slowloris

Slowloris is a type of DoS cyber-attack that works by opening multiple connections to a target web server and keeping them open for as long as possible. It sends partial HTTP requests, causing the server to open more and more connections. Slowloris continues to send subsequent HTTP headers for each request but never completes the request. This fills up the server's maximum concurrent connection limit, making it unable to accept legitimate connection attempts. The difference between a normal HTTP request-response connection and a Slowloris attack is illustrated in Figure 3, showing how the attack prolongs the connection time without completing the request (Ashrif et al., 2024).

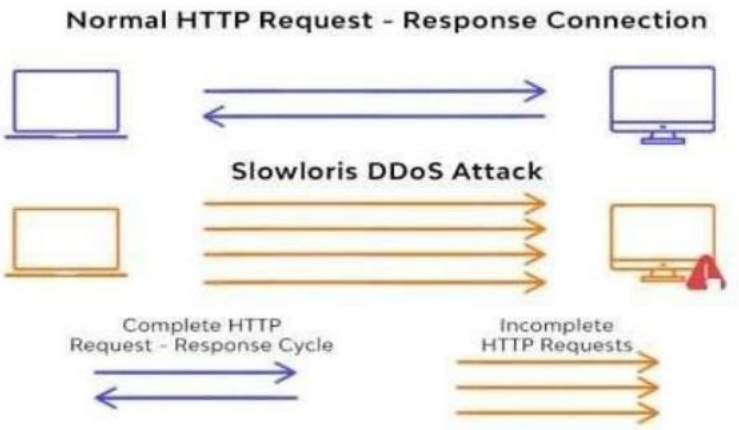


Figure 9 Slowloris Attack

Zero Day DDOS

When a user visits a website, they can exploit security vulnerabilities in their web browser, leading to system infection. Cybercriminals often utilize social engineering tactics to achieve this. For instance, they may send phishing emails containing attachments. Clicking on these attachments can execute malicious code, leading to malware being downloaded and the system becoming infected. Figure 14 illustrates the concept of a zero-day attack (Upreti, 2019).

Figure 10 Zero-Day Attack

Apache

An Apache flood attack, also known as an Apache HTTP server flood attack, is a type of distributed denial-of-service (DDoS) attack that targets Apache web servers, The attacker assembles a botnet, which consists of a large number of compromised computers or devices under the attacker's control. These compromised devices are often infected with malware and can be remotely controlled to carry out the attack (Wibowo et al., 2023)**.**

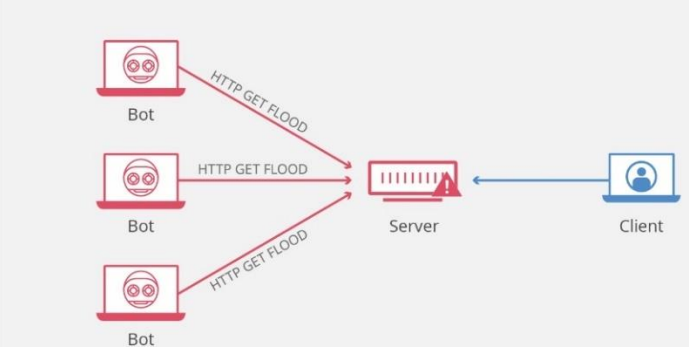


Figure 11 Apache Flood

2.1.6 Machine Learning

Machine learning is a part of artificial intelligence where computers learn to make decisions by themselves without being given specific instructions for every single situation. It uses special algorithms and statistical models that let computers analyze data, understand it, and then use what they've learned to predict outcomes or make choices (M. Wang et al., 2023).

Due to the difficulties in identifying unusual network activities and security breaches with conventional methods, especially in the face of large data volumes and emerging attack types, experts are now employing machine learning techniques (Elejla et al., 2022).These sophisticated algorithms help develop models that can differentiate between regular network behavior and abnormal or potentially malicious traffic, and can also classify various kinds of cyber-attacks based on their specific features.

Machine Learning Algorithms divided into 3 mains categorical as figure 12 shows below.

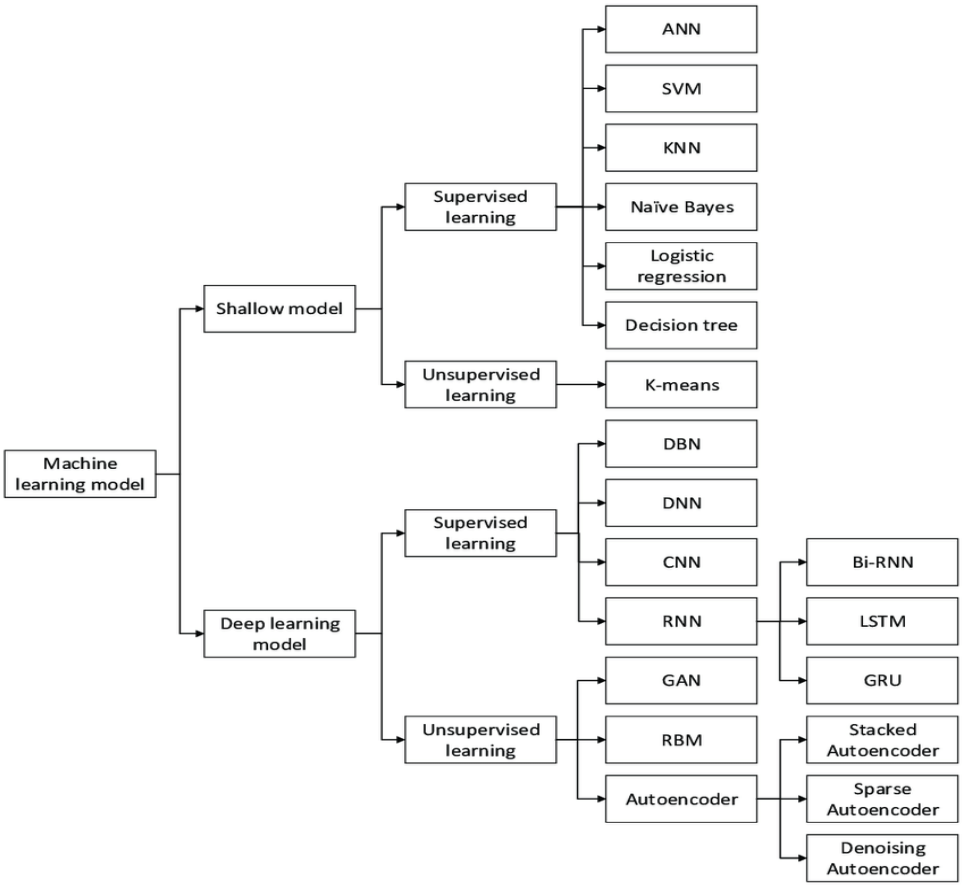


Figure 12 Taxonomy of machine learning algorithms

Machine learning techniques are grouped as follows in Figure 12 based on whether or not labeled examples are included in the dataset:

Supervised Learning

Supervised learning is a method in machine learning where the algorithm is taught using a dataset that already has the correct answers (labels) for each example. The idea is for the algorithm to understand how to match inputs with the correct outputs. Once it has learned from these examples, it can then make predictions or decisions when it comes across new data it hasn't seen before.

The dataset consists of samples from a particular field, each containing input elements known as features, and an output element referred to as the target. In supervised machine learning, an algorithm learns the model's parameters from this dataset. It does this by reducing the difference or error between the actual labels (the known outcomes) and the predictions made by the model for the target variable.

Unsupervised Learning

Unsupervised machine learning works with data input that lacks labels or set categories. The goal is to explore the structure of the data to extract meaningful information without guidance. This is in contrast to supervised learning, where models are trained on labeled data.

On the other hand, we can divide Machine learning based on the task as figure 13 illustrate:

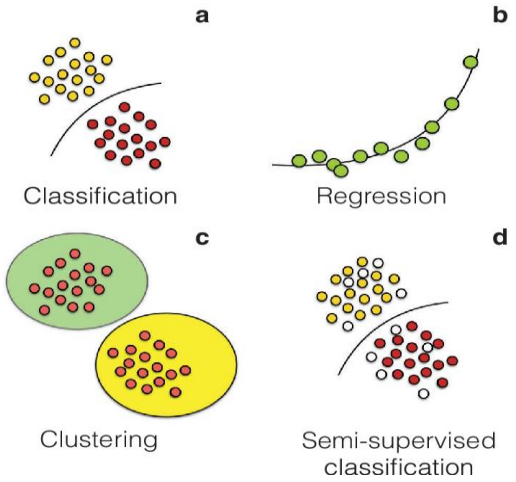


Figure 13 Machine Learning based on Task

Regression: Regression in machine learning and statistics is a method used for predicting a continuous outcome. It involves identifying the relationship between one or more independent variables (predictors) and a dependent variable (target).The goal is to understand how changes in the predictors influence the target variable (M. Wang et al., 2023).

Classification: The aim of classification, a form of supervised learning, is to predict categorical (discrete, finite) results. The target variable in classification is made up of categories, such as "Normal" or "abnormal" in IDS. In a classification problem, the algorithm learns to assign input data into one of the predetermined categories (Sarker, 2021).

Clustering: is a kind of unsupervised learning that clusters data elements according to how similar they are. Unlike classification, in clustering, there are no predefined categories. The algorithm groups the data into clusters based on how similar the data points are to one another compared to those in other clusters (Sarker, 2021).

2.1.7 Semi-supervised classification

is a kind of artificial intelligence that mixes elements of unsupervised and supervised learning. In this approach, a model is trained using a dataset that includes both labeled and unlabeled data. Labeled data has known outcomes or categories, while unlabeled data does not (Guarango, 2022).

Algorithms for Machine Learning Classification:

The Logistic Regression algorithm (LR)

Logistic Regression is an algorithm often used to figure out the chance that something belongs to a specific group. For example, it can tell us the probability of a file being a virus. If this probability is more than 50%, the model predicts that the file is indeed part of that group (in this case, it's labeled as "1" or the positive class)

Conversely, when the calculated probability is below 50%, the model predicts that the sample does not belong to the target group. This prediction is referred to as the negative class, denoted by the value 0. Such a mechanism of classification, based on the dichotomy of probabilities, establishes Logistic Regression as a binary classifier.

The Logistic Regression Algorithm works by adding up the values of various input characteristics (along with a bias), but it doesn't provide the direct result. Instead, it uses a special function called the logistic function, which is often denoted as σ(t). This function gives an output ranging from 0 to 1. The formula for the logistic function is:

$$\sigma(t) = 1 / (1 + e^{(-t)})$$

In this formula, 't' represents the input, and 'σ(t)' is the output. This function helps us determine the probability of belonging to a specific class based on the input values.

Support Vector Machine (SVM)

(SVM) is a highly popular algorithm in cybersecurity. What makes it special is that it uses a line (or plane) to separate different groups of data. This line is positioned in a way that maximizes the space between it and the nearest data points.

You can use SVM in both two-dimensional (2D) and three-dimensional (3D) situations. Its main goal is to accurately categorize data. Some advantages of SVM include its ease of use, high accuracy, and the ability to create this separating line quite efficiently.

However, one drawback is that choosing the right line can be tricky. Despite this challenge, SVM is widely used in various fields like medicine, security applications, and pattern recognition.

2.1.8 Deep Learning

The machine learning algorithms that are based on artificial neural networks comprise a subset known as deep learning, as illustrated in Figure 14, Deep Neural Network (DNN) models involve a neural network architecture that includes two or more hidden layers. These layers perform complex, nonlinear transformations on the input data. As the data moves through each hidden layer, the network extracts increasingly higher-level features. In the context of supervised learning, a DNN model takes an input feature vector 'x' and, via these intermediate transformations, maps it to an output label 'y'.

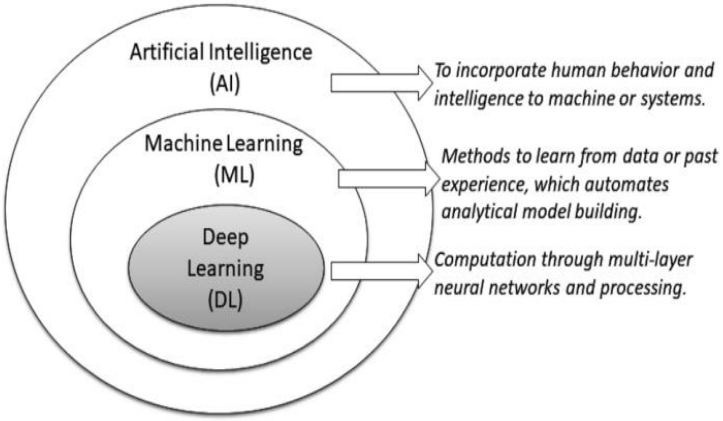


Figure 14 Deep Learning

The most common types of DNN are

Feedforward Deep Neural Network (DNN): operate on the principles of simulating the way human brains process information, using a layered structure of nodes or "neurons.", as figure 15 illustrate the general DNN (Naveed et al., 2022).

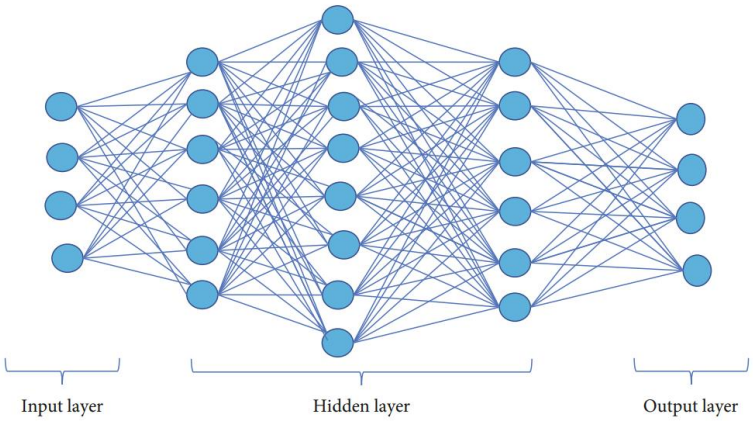


Figure 15 DNN Algorithm

Additionally, it consists of an input layer, multiple hidden layers, and an output layer. Each of these layers is made up of units or neurons that carry out various computations. The input layer is responsible for receiving the initial data. This data is then processed

sequentially through the hidden layers, and finally, the output layer generates the end results. Neurons in each layer are interconnected with those in the preceding and succeeding layers via weights. These weights act as adjustable parameters, enabling the network to learn and identify patterns within the data.

Recurrent Neural Network (RNN)

Artificial neural networks that learn from fluctuations in a temporal sequence of inputs and predict future data are called recurrent neural networks, or RNNs (Robertson et al., 2023), Recurrent Neural Networks (RNNs) are composed of input units, output units, and crucially, hidden units. The core functioning of the RNN model is characterized by a directional flow of information, starting from the input units, progressing via the hidden units, and finally to the output units. What sets RNNs apart is the unique way in which they process information over time. This is demonstrated in the model where information from hidden units at a previous time step is carried forward and combined with the current input.

This process, which can be visualized in Figure 16, allows the RNN to maintain a form of memory, enabling it to handle sequential or time-dependent data effectively.

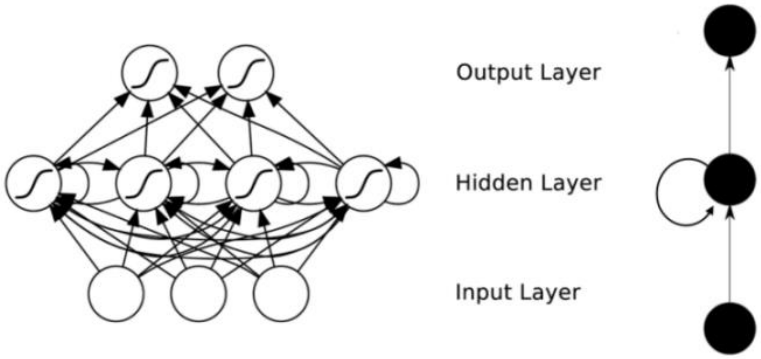


Figure 16 Recurrent neural networks (RNN) (Note & Ali, 2022)

Hidden units in a Recurrent Neural Network (RNN) act like the network's memory, holding onto end-to-end information. When we 'unfold' an RNN, it reveals its deep learning structure, showing how it processes information through layers over time. This characteristic of RNNs makes them well-suited for supervised classification learning, where they can learn from labeled data by recognizing patterns and sequences in the input.

Random Forest

The Random Forest Classifier is a widely-used machine learning method that is great for sorting data into different categories. It's like having a team of decision-makers (trees), each providing their opinion, and then combining all those opinions to get a final decision that is often more accurate than any single one. It's a type of ensemble learning method.

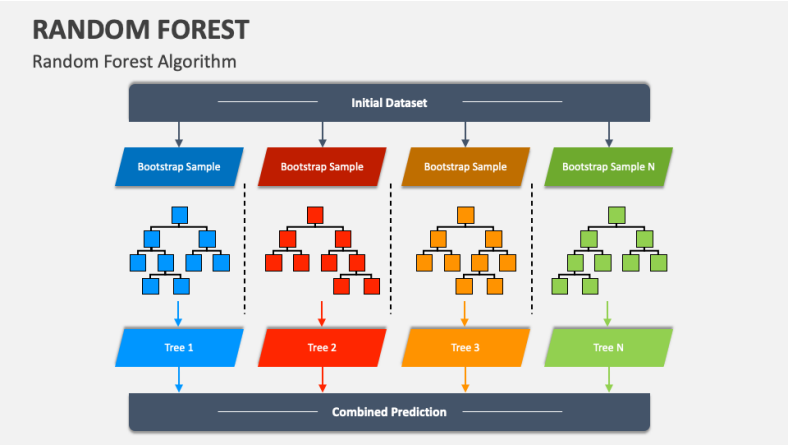


Figure 17 Random Forest Architecture (Dhanya et al., 2023)

Random Forest works by building numerous decision trees during the training phase and producing an output that represents either the most common class among the trees (for classification tasks) or the average prediction across the trees (for regression tasks). It creates a collective or ensemble of decision trees, each trained on distinct segments of the same training dataset. The uniqueness of each tree in the forest arises from being trained on varying subsets of data, leading to slight differences among them.

- **Bootstrap**: Each tree is trained on a random sample of the data, drawn with replacement (this means some instances may be repeated in each sample). This process is known as bootstrap sampling.
- **Feature Randomness**: During the construction of each node in the decision tree, the optimal split is randomly chosen from a subset of features. This introduces randomness into the model, making it more robust and less likely to overfit the training data compared to a single decision tree.

- **Training:** During the training phase, each tree is grown to the largest extent possible and there is no pruning (reducing the size of the tree by removing some nodes).
- **Prediction:** In classification tasks, for a new instance, each tree for the forest predicts a class Label, and the final output prediction is based on the majority vote of all the trees.

2.1.9 Intrusion Detection Dataset

When utilizing deep learning and machine learning methods to develop models for Automated Network Intrusion Detection Systems (ANIDS), it's crucial to have access to extensive network data for training and testing these models' intrusion detection capabilities. Additionally, to effectively compare various models and methods, a benchmark dataset that's publicly accessible is vital. However, generating a comprehensive and labeled network intrusion dataset involves significant time and resources, and privacy concerns further complicate this process. Consequently, there are only a limited number of such datasets publicly available for researchers to evaluate and refine their ANIDS models, some of the famous benchmark IDS database:

- The University of New South Wales Network Behavior 15 dataset (UNSW-NB15): The UNSW-NB15 dataset, created in 2015, is a detailed and diverse resource used for studying network security. It contains over 2.5 million records and is more up-to-date than the older KDD’99 dataset, featuring nine types of modern cyberattacks compared to KDD’99's fourteen types. This dataset also includes a variety of normal internet activities. It stands out with its 49 different characteristics, grouped into six types: Flow, Basic, Content, Time, Additional Generated, and Labelled Features. The Additional Generated Features

are further divided into two types: General Purpose Features (features 36-40) and Connection Features (features 41-47). The dataset sorts cyberattacks into nine categories, including Reconnaissance, Shellcode, Exploit, Fuzzers, Worm, DoS, Backdoor, Analysis, and Generic (Moustafa & Slay, 2015).

UNSW-NB15 is split into two parts: a Training dataset containing 82,332 records and a Testing dataset comprising 175,341 records. Both datasets include all types of attacks and normal traffic records, featuring 45 distinct features (referenced in Table I). It's noteworthy that the 'id' feature, although not listed in the full dataset, is present, while features such as 'scrip', 'sport', 'dstip', 'stime', and 'ltime' are absent in both the Training and Testing datasets (Janarthanan, 2017).

CIC-IDS2017

This dataset developed by the Canadian Institute for Cybersecurity, is a detailed collection of network data that encompasses both normal (benign) activities and a range of modern malware attacks. This dataset is particularly notable for including diverse attack types like Brute Force FTP, Brute Force SSH, DoS (Denial of Service), Heartbleed, various Web Attacks, Infiltration, Botnet activities, and DDoS (Distributed Denial of Service) (Baldini et al., 2021)**.**

Each entry in the dataset is meticulously labeled, with details such as timestamps, src and dest IP addresses, source and destination port numbers, protocols used, and the specific type of attack. The data was gathered through a carefully configured network topology,

which contained parts including modems, firewalls, switches, routers, and nodes running several Linux distributions, Microsoft Windows, and Apple's macOS and iOS (Ullah et al., 2023)**.**

The CIC-IDS2017 dataset is comprehensive in its scope, capturing 80 different network flow features from the network traffic data. This rich feature set makes it an invaluable resource for cybersecurity research, in particular while building and evaluating intrusion detection systems.

2.2 literatures Review

In the previous 20 years many researchers worked over DoS attacks they have tried to enhanced the detection metrices of these attacks, researchers used several methodologies included Deep Learning and Machine learning.

1- Deep Learning Methods

In the work by Farhan Ullah (Ullah et al., 2023) used Transfer Learning approach, it’s a multi-head attention using the BERT large model, that is designed to mine network data and extract features to improve the performance of IDS, also used (CNN-LSTM), (CNN_RNN) and (CNN- GRU) they have applied their model over three benchmark datasets NSL-KDD, UNSW-NB15 and CIC-IDS2017, the model addressed 94%.

Authers In this work (Naveed et al., 2022) proposed Deep Neural Network (DNN) model over the NSL-KDD dataset for intrusion detection. The model achieved an accuracy of 99.73%. One drawback of this method could be the complexity of implementing and

training deep learning models, which may require significant computational resources and expertise.

Researchers also employed additional statistical techniques. For example in this work (Kurniawan et al., 2021) focused on statistical analysis methods rather than deep learning or machine learning algorithms, they utilizes statistical distribution models such as Weibull, Pareto, Gamma, Exponential, and Lognormal distributions to analyze and model the packet inter-arrival times, The dataset for the study was gathered using the NetFlow protocol, which provides information on network traffic patterns and packet inter-arrival times , one drawback of the model used in the study is that it assumes uniform time intervals between packets within network flows, which may not fully capture the complexity of real-world network traffic dynamics. Additionally, the study highlights the importance of understanding distribution trends in network traffic modeling but does not delve into the application of deep learning or machine learning algorithms in this context.

2- Machine Learning Methods

In another work, (Garsva et al., 2014) also used UNSW-NB15 applied Random Forest, the model addressed 76% accuracy but the method has certain drawbacks, such as the difficulty of comparing datasets with disparate characteristics, the requirement to reduce feature counts in order to improve detection rates, and certain complications with managing high-dimensional data in intrusion detection systems.

In another investigation (Baldini et al., 2021) Random Forest algorithm was employed to analyze the UNSW-NB15 dataset this subset included service type, source to destination

bytes, source to destination time to live, mean of packet size transmitted by the scrip, and the number of rows of the same dst ip and the sport in 100 rows, the UNSW-NB15 dataset consists of nine different modern attack types, with a total of 2,540,044 records, categorized into six groups of features. The study aimed to reduce the number of features to enhance intrusion detection rates while minimizing resource consumption. While the Random Forest algorithm performed better than other algorithms tested, it is essential to note that the KDDCUP'99 dataset contains more instances compared to the UNSW-NB15 dataset. This difference in dataset sizes can impact the performance metrics and comparisons between the two datasets.

According to (Abdalla et al., 2018) machine learning algorithms such as Support Vector Machine, Decision Tree, and Naive Bayes applied for intrusion detection. The Decision Tree algorithm showed optimal detection accuracy. they focused on detecting DDoS and Port Scan attacks on the CIC-IDS2017 dataset. The proposed approach achieved competitive results compared to existing methods, with lower Error Rate and False Negative Rate. This work highlighted the importance of feature selection and hyperparameter optimization for effective intrusion detection. There were some limitations, including the need for further evaluation on diverse datasets to generalize the algorithm's performance. Additionally, the interpretability of the algorithm and scalability to large-scale networks could be areas for future research and improvement.

In another study, (Garcia & Blandon, 2022) uses a combination of machine learning algorithms, such as The algorithms used in this work include neural networks, decision trees, one-class support vector machine, subspace clustering, replicator neural networks,

random forest, and more, for anomaly detection in telecommunication traffic. The accuracy of the models varies based on the level of attack and the characteristics of the traffic data. The dataset SNMP (Simple Network Management Protocol) data for detecting network anomalies used contain both legitimate traffic and DDoS attacks, with different parameters set for detection thresholds, The drawbacks of the model include potential false positives, lost data, and varying performance based on the complexity of the traffic patterns and attack levels.

Table 1 Summary of Literature review

Work	**Algorithm**	**Dataset**	**Accuracy**
(Ullah et al., 2023)	-CNN-LSTM -CNN-RNN -CNN-GRU	NSL-KDD	94%
(Garsva et al., 2014)	Random Forest	UNSW-NB15	76%
(Abdalla et al., 2018)	Support Vector Machine, Decision Tree, and Naive Bayes	CIC-IDS2017	90%
(Kurniawan et al., 2021)	statistical analysis methods	Collected data	80%
(Garcia & Blandon, 2022)	decision trees, one-class support vector machine, subspace clustering, replicator neural networks, random forest	SNMP (Simple Network Management Protocol)	82%
(Naveed et al., 2022)	Deep Neural Network (DNN)	NSL-KDD	99.73%

(Baldini et al., 2021)	Random Forest	UNSW-NB15	93%

Based on the above literature review, a DoS collected data will used in this work, so this data contains various types of DoS attacks.

Chapter Three: Methodology

Chapter Outline

3. Chapter Three Methodology

3.1 System Setup

3.1.1 Environment Architecture

Network emulation is a commonly employed method for assessing performance, testing and troubleshooting protocols, and addressing various research topics related to networks(Fontes et al., 2015), Emulation offers a cost-effective alternative to expensive real testbeds and, unlike simulations, it enables the execution of actual code under realistic network and computing scenarios. This approach is particularly supportive in research related to in the domain of Software-Defined Networking (SDN) architectures, the Mininet emulator facilitates extensive experimentation and rapid prototyping cycles.

Mininet is equipped with Linux network software and supports OpenFlow, enabling custom routing and the implementation of Software-Defined Networking (SDN) (Flauzac et al., 2019). Since Mininet requires installation on a Linux server, we chose VM Workstation Pro 15 for our emulations. The emulation was conducted on a PC equipped with 64-bit Windows 10, powered by a Core-i7 processor, and supported by 16 GB of RAM.

The methodology divides into five stages, first stage build the environment and install the tools, second stage create several types of DoS attacks, third stage monitoring the network traffic fourth stage collect the data with pcap file and preparing dataset to apply machine learning models fifth stage evaluate the models.

We developed a basic Mininet SDN (Software Defined Networking) virtual network environment, as illustrated in Figure 18. This setup, built on a preconfigured Linux-based Mininet Virtual Machine, was installed on a VMware workstation. The environment features four hosts linked with three switches, all overseen by three internal mininet Open VSwitch (OVS) controllers.

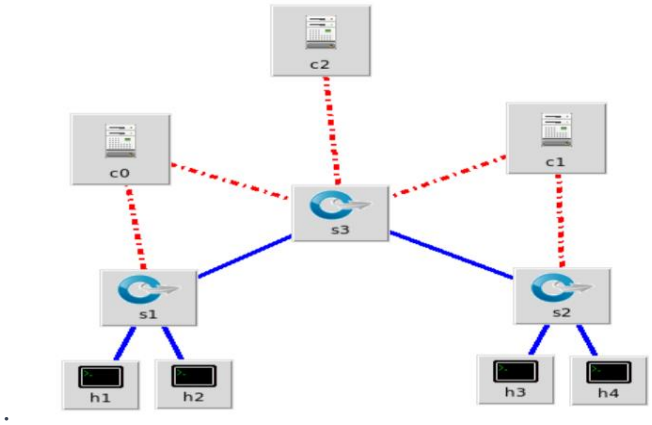


Figure 18 Mininet Topology

Then we install needed tools such as Hping3, and Wireshark on a switch, Hping3, originally developed as a security tool, has evolved into a versatile instrument for testing networks and hosts. This TCP penetration tool serves various purposes, including firewall testing, conducting traceroutes across different protocols, scanning ports, and generating packets. It is widely recognized for its ability to actively probe and analyze network environments (Upreti, 2019).

Also, Hping3 enables users to craft and send customized packets with specific headers, payloads, and flags. This functionality is useful for testing network responses and behavior under different packet scenarios, in addition it can perform various types of

network scans, including TCP, UDP, and ICMP scans, to discover open ports, services running on target hosts, and network topology information.

While hping3 can be misused for malicious purposes, it can also be used for legitimate Denial of Service (DoS) and Distributed Denial of Service (DDoS) testing to evaluate network resilience and response to high-volume traffic (Suthar & Patel, 2023).

Table 2 Represent the command used to perform a DoS attack where -S: SYN flag, -c: packet count, -p: destination port

Attack Type	Command
ICMP	hping3 — icmp -c 4 -d 9000 Target host IP
TCP	hping3 -S -p 80 Target IP — 9000
SMURF	hping3 — icmp — 9000 IP host 1 -a IP host 2

The creation of the proposed models utilized these resources:

- Anaconda, an open-source environment, with Python version 3.8.11.
- Keras library
- TensorFlow version 2.3.0
- Scikit Learn for machine learning tasks.
- Pytorch Library

The proceedings were conducted on an HP Inspiron 15 laptop, equipped with an Intel(R) Core (TM) i7-7600U CPU, clocked at 2.80GHz to 2.90GHz, and 16 GB of RAM. Additionally, the Jupyter Notebook environment was employed for conducting some of the experiments

3.1.2 Data Collection

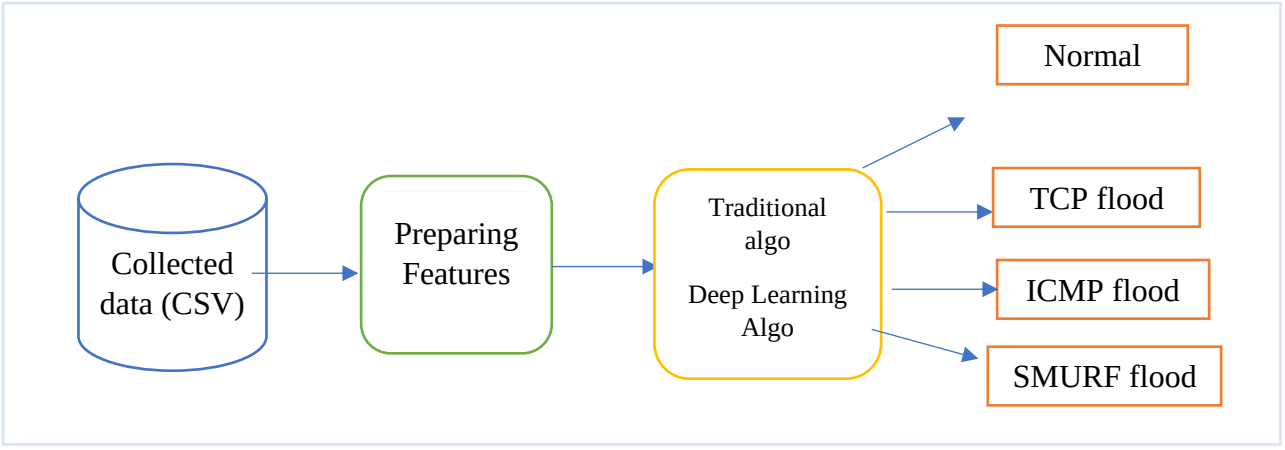
Wireshark:

The widely used and well-known Wireshark used network analyzer, serves many purposes globally for both commercial and non-commercial organizations. This tool enables these entities to conduct thorough examinations of network traffic and protocols, The software is adept at capturing, analyzing, and monitoring live network traffic and protocols, with the added capability to save and export the results to various platforms(Upreti, 2019), after hitting the hosts by DDOS attacks by hping3 we collect the monitoring traffic by Wireshark tool, For Each type of DoS there is a PCAP file with main features as illustrated in table 3 bellow

Table 3 Features Extracted by Wireshark

Features extracted by Wireshark (PCAP)		
#	Feature	Description
	IP Src	The address of the device sending the data
	IP Dest	The address of the device receiving the data
	Time	refers to the timestamp when the packet was captured
	Protocol	Type of the protocol
	Length	Packet's size
	Info	provides a brief summary or description of each packet

The architecture of the methodology after collecting the data illustrated bellow



3.1.3 Data Preparation:

Deep learning models require numerical input, which is mean all categorical features must be converted into numeric form for processing, while deep learning algorithms excel in recognizing anomalies in raw data and representing features, the creation of new, relevant features can further enhance the model's performance. Additionally, selecting only essential features can both reduce the complexity and optimize the efficiency of the model. To adequately prepare data for this, the following processes can be employed.

Feature Engineering

Calculating a new Features from provided one in our case we use the statistical features which calculate from the CSV file. Using statistical features in Intrusion Detection Systems (IDS) to detect DoS (Denial of Service) attacks is highly effective due to several reasons**,** each statistical feature captures a different characteristic of the network traffic. For example, mean and median can reflect the average traffic flow, standard deviation indicates the variability, The minimum value in a stream containing eight packets and the maximum value among those eight values and Max value of the stream which indicate to the maximum value from the eight values in stream (Papadogiannaki et al., 2022).

Determining the similarity between two-time intervals is a key aspect of time series analysis, particularly in scenarios involving encrypted traffic. Numerous tasks, including segmentation, prediction, clustering, classification, and anomaly detection, depend on this technique. The selection of the minimum range for the inter-arrival time feature, which can vary, should be based on the desired accuracy and the time unit used (seconds or milliseconds).

Data Preprocessing

Data preparation encompasses processes such as cleaning the data, applying encoding techniques, discretizing, and scaling the data.

a) **Data cleaning**:

In the preprocessing phase, this step is essential as it addresses missing values in the CSV file. This can be accomplished through various techniques, such as filling the gaps with 'NAN' or eliminating the rows with empty cells. Additionally, this step involves managing any duplicate rows to ensure data integrity.

b) **Data Encoding**:

Algorithms only can work with numerical data. datasets often contain nominal categorical features. To make these features compatible, they must be converted into numerical form. This can be achieved through either ordinal or Boolean encoding. For nominal categories comprising n different values, one common approach is to use One-Hot encoding, which creates (n_categories - 1) binary dummy variables. If there are many values, these categories can be converted into ranked numbers, a process called Ordinal Encoding. This is part of Data Discretization: we apply encoding over the Label column which is the target column.

c) Data Scaling or Normalization:

Data Scaling or Normalization is a crucial preprocessing step in data analysis and machine learning for several reasons:

- Uniformity in Feature Range: In a dataset, different features often exist on different scales Feature scaling is important step to ensure that attributes with larger scales do not dominate or poorly influence those with smaller scales. By scaling, we ensure that each feature contributes equally to the analysis or model, providing a more balanced and effective input for machine learning algorithms."

MinMaxScaler: make all of variables between 0 and 1 as in equation bellow

$$\mathbf{Xscaled} = \frac{\mathbf{X} - \mathbf{minX}}{\mathbf{maxX} - \mathbf{minX}}$$

Standard Scaler:

If the input variable is normally distributed, it can be normalized to achieve a mean of zero and a standard deviation of one.

Feature Selection:

Feature selection entails reducing the number of input variables. By designing a machine learning model with only the most pertinent features, the model's generalization capability is enhanced, which in turn improves its accuracy. This procedure also cuts down on computational costs and decreases the complexity of the model.

In our case, we calculate the timestamp from the Time column then we calculate the Inter arrival time (IAT) then we grouped each eight packets with stream and for each stream we calculate the min value, max value, mean, median, Std_diviation and entropy. At last step we labelled the dataset.

In the Network in general the Byte has eight bits, having eight packets in a stream can help in error detection and correction mechanisms. It allows for better error recovery and retransmission strategies in case of packet loss or corruption, in addition Some systems or protocols may have specifications or limitations that make eight packets a suitable choice for stream size in order to ensure compatibility and interoperability with other components.

Inter arrival time (IAT) refers to the time interval between the arrivals of consecutive packets, it’s the interval between each sequential tow packets, this metric is crucial in understanding traffic patterns and network performance. In more technical terms, if you consider a sequence of data packets arriving over a network, the inter-arrival time is the difference in arrival times between two successive packets. For instance, if one packet arrives at time T1 and the next arrives at time T2, the inter-arrival time is T2 - T1. Analyzing inter-arrival times can help detect anomalies in network traffic, which could indicate cyber-attacks, like Denial-of-Service (DoS) attacks, The features in our dataset presented as in Table 4 below.

Table 4 The Statistical features for streams in our Dataset

#	Feature	Explain
1	stream	We aggregated the packets with different size in stream (8 packet)
2	Interarrival time (IAT)	Refers to the interval between each sequential tow packets
3	Mean	offer insight into the central tendency of the data
4	Median	indicating the typical behavior or size of the network traffic.
5	Standard deviation	measures the dispersion or variability of the data points around the mean, helping to gauge the level of fluctuation or stability in the traffic patterns.
6	Entropy	high entropy may indicate a lack of clear structure, potentially reflecting attack attempts that aim to obscure their patterns.
7	Min	The min value in each stream
8	Max	The max value in each stream

3.14 Deep Neural Network Classifier

Deep learning has become increasingly important in cybersecurity, particularly in detecting cyber-attacks like Distributed Denial of Service (DDoS) attacks. These attacks aim to flood a system, network, or service, causing it to become inoperable.

Deep learning models, specifically those using convolutional neural networks (CNNs) or recurrent neural networks (RNNs), are effective in automatically extracting and learning features from raw data, which is essential in identifying sophisticated DDoS attacks.

Recurrent Neural Networks (RNN)

RNNs have recently implemented feed-forward neural networks.

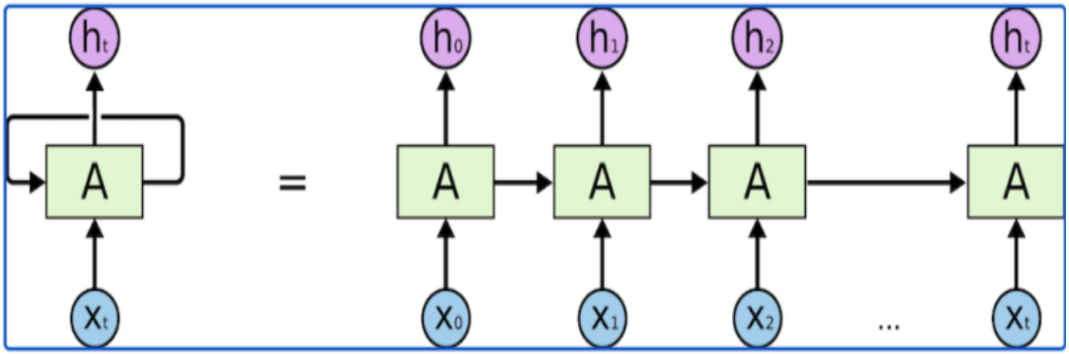


Figure 19 Simple RNN (Robertson et al., 2023)

RNNs Designed to handle input sequences of indefinite length, the term "series" implies that each input in the sequence is related to or influences its neighboring inputs, as illustrated in figure 19 above. While basic feed-forward networks retain knowledge from their training phase, their memory is confined to what was learned during that period. however, Recurrent Neural Networks (RNNs) not only learn during training but also retain information from previous inputs when generating outputs. This ongoing memory capability enables RNNs to more efficiently process sequential data. However, there are two significant drawbacks to this approach.

- Long sequences often result in a vanishing gradient issue or challenges with long-term dependencies. Essentially, this means that the network's memory struggles to retain and utilize information from distant connections in the sequence.
- it’s slow to train.

Long Short- Term Memory networks are a specialized form of Recurrent Neural Networks (RNNs), specifically developed to overcome the issue of vanishing gradients. These networks are adept at learning and maintaining long-term dependencies. Indeed, their inherent strength lies in their ability to remember information over prolonged durations as a default feature, rather than a capability they need to acquire.

When dealing with network traffic as input, LSTM neurons have a unique feature compared to standard neurons: they include a pathway that permits certain information to bypass the extensive processing of the current cell. This mechanism enables the network to preserve memory over longer periods. While this design helps mitigate the vanishing gradient issue, it does so with limited effectiveness.

Simple Recurrent Neural Networks (RNNs) are notably slow in training, which can be even more pronounced compared to other neural architectures. This sluggishness arises because these networks process inputs sequentially, one after the other, which doesn't effectively leverage the capabilities of GPUs that are optimized for parallel computations. Two significant challenges with RNNs are the vanishing gradient problem, which hampers the effective training of deep networks, and the inherently slow training process. In subsequent discussions, strategies for parallelizing sequential data will be explored to address these issues.

In each step of the encoder or decoder, the RNN processes its inputs and produces output specific to that time step. During these steps, the RNN updates its hidden state, taking into account both the current inputs and the outputs it has generated previously.

As illustrated in the animation, this hidden state serves as the context vector, which is then passed on to the decoder.

Random Forest Classifier

The Random Forest Classifier is a popular and versatile machine learning algorithm used for classification tasks. It's a type of ensemble learning method.

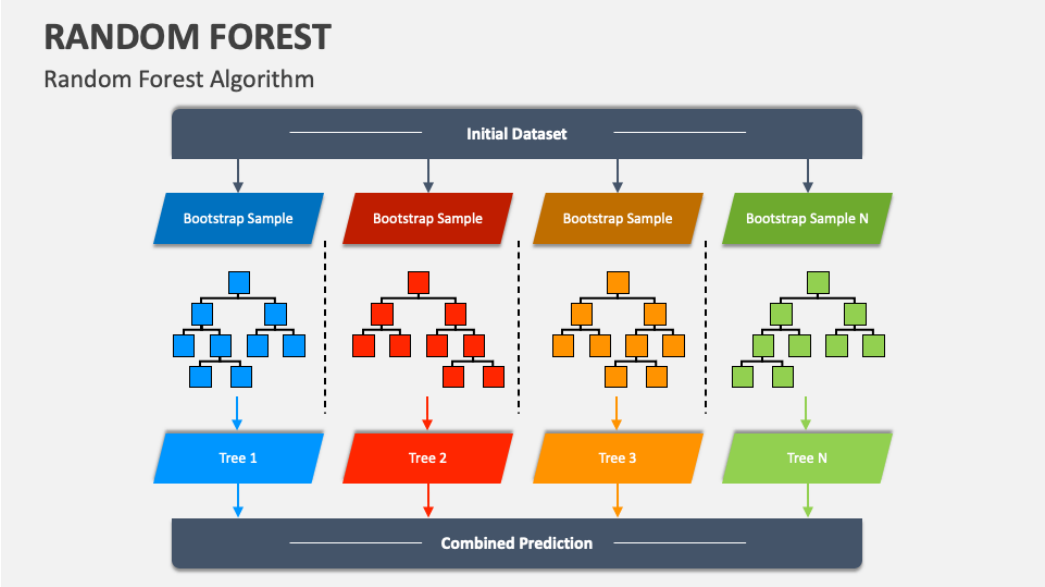


Figure 20 Random Forest Architecture (Fernando et al., 2023)

Random Forest

works by building a large number of decision trees during training, and the result represents either the majority class (in classification tasks) or the mean prediction (in regression tasks) across these trees. It creates an ensemble of decision trees, with each tree trained on different subsets of the training data. This approach ensures that each tree is distinct, influenced by different segments of the dataset, leading to a diverse and robust ensemble.

- **Bootstrap**

Process where every tree is trained using a different subset of the data, drawn with replacement (this means some instances may be repeated in each sample).

- **Feature Randomness**

At each node in the decision tree construction process, the optimal split is selected at random from a subset of features. Compare it to a single decision tree, the model is more resilient and less prone to overfit the training set thanks to this randomness.

- **Training**

During the training phase, every tree is grown to its full potential without any pruning, which would reduce the tree's size by removing some of its nodes.

- **Prediction**

In classification tasks, for a new instance, each tree for the forest predicts a class Label, and the last output prediction is based on the majority vote of all the trees.

3.2 Data Collection

In this phase, we will talk about the dataset, data cleaning methods, and the design, training, and evaluation of the suggested model. Figure 20 provides a visual representation of these processes.

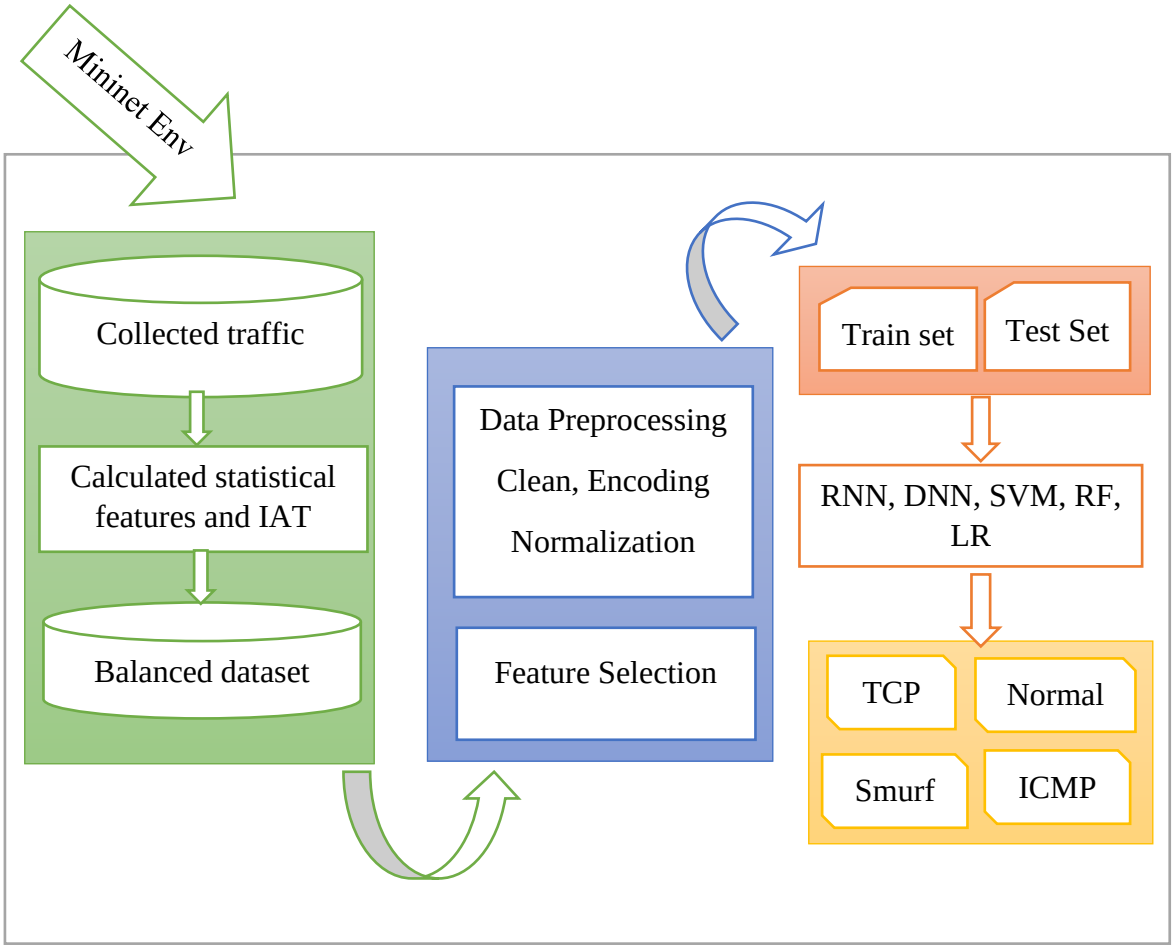


Figure 21. Proposal Model

Algorithm 1 Proposed Recurrent Neural Network Model

Input: Collected DoS and Normal data

Output: Prediction of four Classes (1 ICMP, 2 Normal, 3 Smurf, 4 Tcp)

Step 1. collect DoS, Normal data as PCAP.

Step 2. Merge all PCAP file to be Labelled CSV file.

Step 3. Split the dataset into the train set with size 75%, and the test set with size 25%.

Step 4. Design RNN model architecture

Step 5. Learn the RNN model on train set

Step 6. Examine the execution of the model using the test set.

Step 7. Adjust the model's performance by varying and optimizing the criteria

Step 8. Store the trained model and implement it for the detection of various types of DoS (Denial of Service) attacks.

3.2.1 Created Data

Creating the dataset stands as a crucial stage in implementing a machine-learning approach. Data is typically sourced from diverse origins and may exhibit noise, redundancy, incompleteness, or even contradictions. This foundational step is crucial in shaping the quality and efficacy of the ensuing machine-learning process.

The dataset was generated through controlled simulations using the Mininet network emulator, the mininet topology contains four hosts, two switches, and one built-in controller type POX, these devices are connected in the Mininet environment. The DoS attacks were created by the hping3 packet crafting tool, each type of DoS was created by running a command with specific conditions to ensure data integrity and consistency.

At first, Normal traffic was created, with around 6600 individual connections. These connections were split into streams with 8 packets each one, and they had some numbers that tell us about them, like the Min and Max values, Mean, Median, Standard deviation, and the entropy for each stream.

In a TCP flood attack, we send a huge amount of SYN (synchronized) packets to the target with the intention of overwhelming its resources and causing disruption. We created 779681 connections of SYN requests and ACK replays. Due to the nature of the three-hand shake of the normal TCP connection this flood will take a bit more time than others.

The ICMP flood was captured using Wireshark, this tool is a powerful network protocol analyzer used for network troubleshooting, analysis, software and communications protocol development, In this particular attack, an extensive multitude of ICMP packets were dispatched to the designated target, with the strategic objective of generating an overwhelming effect. approximately 2496 requests were generated. This attack manifests as a form of volumetric assault, characterized by its focus on inundating the target's resources.

SMURF flood is a type of network-layer distributed denial-of-service (DDoS) attack. The attack involves sending a large volume of Internet Control Message Protocol (ICMP) echo request (ping) traffic to a network's broadcast address. The attackers typically use IP spoofing to send these requests, making them appear to originate from the victim's address, we generated around 850 observations.

Wireshark generated a pcap traffic file for each type with a few features like (IP source, IP destination, Time when this packet has captured, Protocol, Length, and Notes about the packet) the most feature was important to us was Time, based on this feature we calculate the timestamp and inter-arrival time for each file. As a result, our dataset contained three type of DoS attacks in addition to Normal traffic each packet with its label.

As illustrated in figure 22 the packets for each type were different after grouped to the streams with size 8 pkt/ stream, so this brings an imbalanced dataset

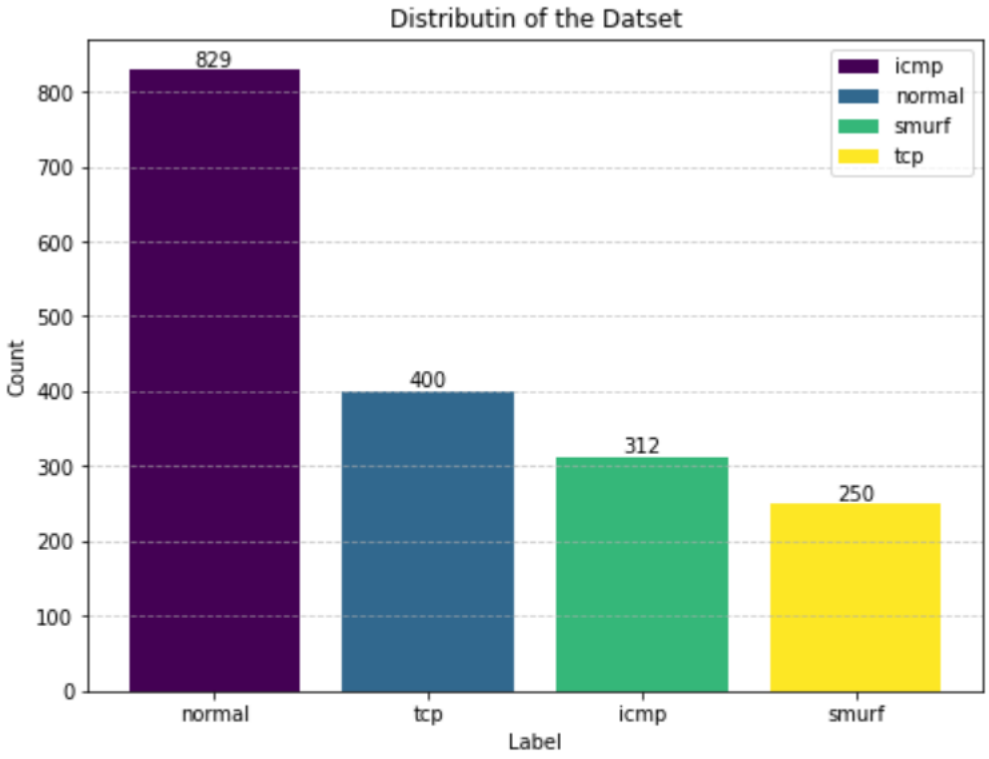


Figure 21 Distribution of the dataset

This problem can be addressed using the SMOTE technique, which utilizes two primary methods: oversampling and under sampling. The oversampling aspect of SMOTE concentrates on the minority class, replicating its observations to match the quantity of the majority class. On the contrary of under sampling focus on the majority class, we balanced the dataset using under sampling. Each class has 800 connections.

3.3 Data Preprocessing:

3.3.1 Clean Data

Algorithm 2 Cleaning Created dataset

Step 1. Load selected dataset

Step 2. Clean Data and remove duplicate rows, fill the empty cell with NAN.

Step 3. Calculate the IAT and statistical Feature and grouped each 8 pkt/stream (Min, Max, Mean, Median, std-deviation, Entropy).

Step 4. Remove unvalued columns.

Step 5. Under sample the majority classes.

Step 6. Do One-Hot encoding for categorical Features

Step 7. Using MinMaxScaler

Step 8. Repeat Cleaning features in dataset

Step 9. Divided the dataset into 75% for train set and 25% for test set

Statistical-Based Approach

The basic idea behind these methods is that similar attacks share common statistical features. By spotting these similarities, we can identify these attacks without needing to understand the context. These methods keep track of how different events relate to each other and use past data analysis to figure out how often certain things happen when the system is being trained, The statistical characteristics of a packet flow with temporal fluctuations, such as mean and variance, are impacted by a DoS attack (Nichelini et al., 2023).

Our method for spotting DoS attacks in cloud computing is mainly based on using a better feature than what was used before. Instead of just looking at individual packets, we focus on something called "packet IAT" which stands for the time between when one packet arrives and when the next one does. By studying how these time intervals work, we can figure out how likely it is that a DoS attack is happening in a stream of network traffic.

Consider a packet, which we'll call "P" We can describe this packet by looking at the time between when it arrives and when the next one arrives in a sequence of traffic. Let's call this time "Pt," which equals At+1 (the time the next packet arrives) minus At (the time the current packet arrives). We do this for all the packets in the sequence.

$$Pt = \text{At}+1-At$$

Pt: represents the time interval between the arrival of two consecutive packets.

At: represents the time the current packet arrives.

At + 1: represents the time the next packet arrives

Now, what makes a DoS attack stand out is that it often follows a pattern where the time between packet arrivals seems somewhat predictable, like following an exponential distribution. We can use this pattern to spot unusual behavior or anomalies in the traffic.

3.3.2 Setup of Proposed Model

In this phase we will talk in general about each Model Setup and the inside contents of each one.

RNN

a. Multi-Class RNN_LSTM_GS Model Setup

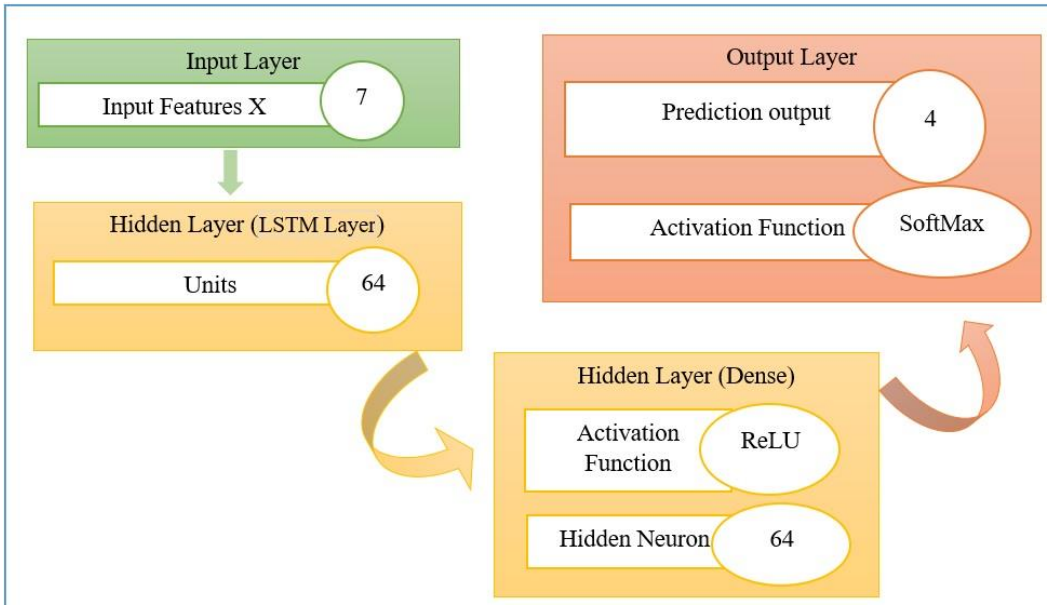


Figure 22 Design and setup of RNN_LSTM_GS model

For the task of multiclass classification, the designed model aims to categorize network flows into seven distinct groups, these include one category for normal connections, labeled as 'Normal', and three categories for different types DoS attacks: Icmp, Tcp and Smurf. The architecture of the model is a RNN_ LSTM_Grid Search that specifically facilitates multiclass classification. It is structured as follows:

Input Layer: The first layer of the model is designed to handle seven features of dataset

Hidden Layers:

LSTM Layers:

The first hidden layer is an LSTM (Long Short-Term Memory) layer. The quantity of instances in this LSTM layer is specified as 64. This means there are 64 LSTM cells, each capable of maintaining a state and producing an output which is passed on to the next layer.

Dense Layer

The Dense layer which also acts as a hidden layer in this network. this layer has 64 units, as indicated by the first parameter in the model, it employs the Rectified Linear Unit (ReLU) as its activation function, which is standard for deep learning models due to its effectiveness in non-linear transformations, and a dropout of 20% is incorporated in this layer. Dropout is a regularization technique where a fraction of the neurons is switched off at random during training, requiring the network to pick up more resilient features

Output Layer

The final part of the model is the output layer, which is composed of four neurons, corresponding to the four classes of network flows the model is designed to identify. The activation function used here is SoftMax, which is particularly suited for multiclass classification tasks. SoftMax converts the output of the network into a probability distribution over the predicted output classes, ensuring that the sum of the probabilities for all classes equals one.

b. Selecting Model Parameters

Hyperparameters consult the particular parameters of the model that needs to be decided upon before the training phase begins. The selection of certain hyperparameters for our model was guided by heuristic approaches and recommendations found in existing literature. Table 5 presents a detailed list of these chosen hyperparameters for the suggested models.

Table 5 hyperparameters of the RNN_LSTM_GS model

Permeter	Value
Optimizer	Adam (learning rate = 0.001)
Loss	Cross Entropy
Epochs	100 with early stopping and patience of 20
Batch size	32
Weight initialization	uniform
Regularization	Dropout (20%)

the hidden layers utilize the ReLU activation function, while the Adam optimizer is employed, set at a standard learning rate of 0.001. In the case of multiclass classification, the SoftMax activation function is used to avoid overfitting and enhance model performance, the early stopping regularization method halts training as soon as improvements on the validation set cease due to further parameter updates. Consequently, this technique determines the ideal number of training epochs. With RNN we used two methods Grid Search and SMOTE (Synthetic Minority Over-sampling Technique) alongside RNNs (Recurrent Neural Networks) is primarily driven by the need to address class imbalance in datasets, especially in scenarios where time-series or sequential data is involved. On the other hand, Grid Search is a widely used technique for hyperparameter optimization in machine learning models. It entails methodically going through a preset list of hyperparameters in order to identify the configuration that provides the best results.

Grid Search and SMOTE serve distinct purposes in machine learning. Grid Search is a method for hyperparameter optimization, focusing on systematically exploring various combinations of parameters to enhance a model's performance. It operates during the model training phase, influencing how a model learns from data without altering the data itself. On the other hand, A data processing technique called SMOTE (Synthetic Minority Over-sampling Technique) is used to solve class imbalance problems in datasets. Before the training process starts, it balances the class distribution by generating artificial samples from the minority class. While Grid Search optimizes the learning process, SMOTE modifies the training dataset to improve the model's ability to learn from underrepresented classes.

DNN: The model is set up as a multi-class DNN, suitable for classifying instances into one of several classes.

- **Input Layer:** The model starts with an input layer designed to handle 7 features.
- **Hidden Layers:** The first hidden layer has 64 neurons (Dense(64)) and uses the ReLU activation function (activation='relu'). ReLU (Rectified Linear Unit) is a popular option for hidden layers because it helps with non-linearity and gradient flow, The second hidden layer has 32 neurons and also uses the ReLU activation function.
- **Output Layer:** this layer has 4 neurons (Icmp, Normal, Smurf, Tcp) according to the quantity of classes in the dataset. It uses the softmax activation function (activation='softmax'), It is commonly used for multi-class classification tasks, producing a probability distribution among the classes.

a. Multi-Class DNN Model Setup

we will present in general DNN Model Setup and the inside contents. As the figure15 presented below

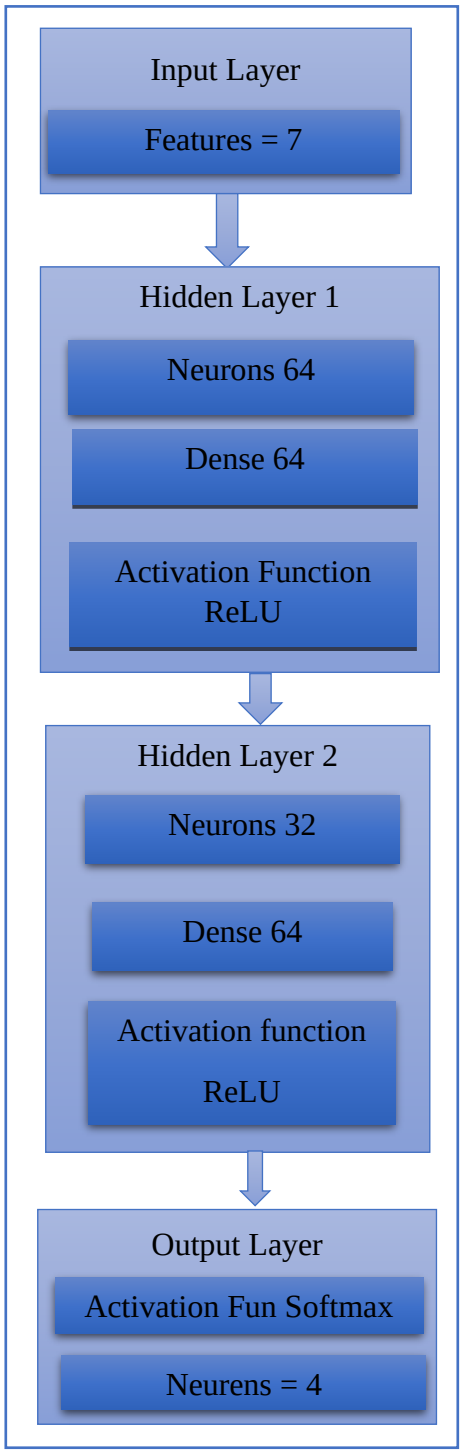


Figure 23 Design DNN model

b. Selecting Model Parameters

Parameters refer to the configurable settings of the model that are established prior to initiating the training process. For our study, specific Parameters were determined based on heuristic methods and guidance from relevant literature. Table 6 provides a comprehensive enumeration of these chosen hyperparameters for the model we propose

Table 6 hyperparameters of the DNN model

Hypermeter	Value
Optimizer	Adam
Loss	Cross Entropy
Epochs	10 with early stopping and patience of 20
Batch size	32
Weight initialization	uniform
Regularization	Dropout (20%)

Also, with DNN we used SMOTE and Grid Search due to their efficiency with imbalanced dataset.

Random Forest

a. Selecting Model Hyperparameters

This is used for classification tasks, where the goal is to categorize data into predefined classes or labels. For example, a Random Forest Classifier can be used to identify whether an email is spam or not, classify types of plants based on features, or determine if a transaction is fraudulent. In classification, the output is a discrete label.

Table 7 hyperparameters of the Random Forest model

Permeter	Description	Value
Estimator	Increasing the number of trees can enhance the accuracy of the model, but it also escalates the computational cost and complexity	100, 200, 300
Max _Depth	Deepest level allowed for the trees. While deeper trees have the potential to capture more detailed information from the data, they also run the risk of over fitting.	None, 10, 20
Min _Sample Split	Min number of samples necessary to split an internal node. Higher values act as a restraint, discouraging the creation of nodes that represent an insufficient number of instances.	2, 5, 10
Min Sample LEaf	This parameter specifies min number for a leaf node. Setting a smaller minimum for the leaf can make the model more sensitive to noise in the training dataset	1, 2, 4

a. Multi-Class Random Forest Model Setup:

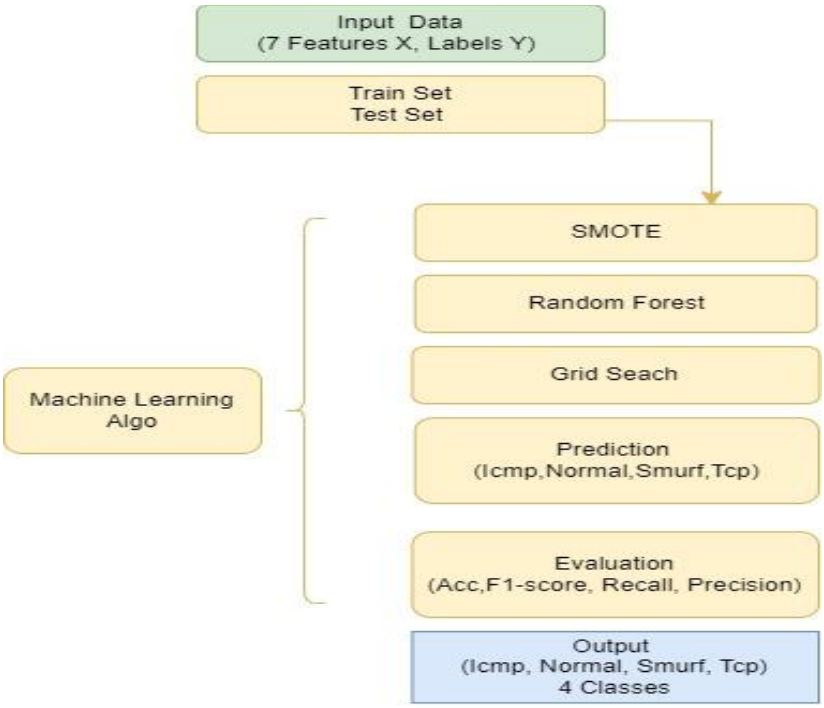


Figure 24 Random Forest Diagram

3.4 Model Training

In machine learning domain, we have a big aggregation of data that we split up into two sections: the training set and the test set, The training set used to teach the model how to recognize patterns and make predictions, Once the model has 'learned' from the training set, we need to check how well it has learned. This is where the test set comes in, the test set is made up of data that the model hasn't seen before. By using the test set, we can evaluate how good the model is at making predictions on new, unseen data, The derived dataset was split into 75% for training and 25% for testing.

The way we split the data into training and test sets is similar to how we prepare for detecting new, unknown cyber threats, like zero-day attacks, the term "zero-day" comes from the fact that the attack occurs before the vulnerability's existence is known to the software developers or the public. This means that there are zero days between the time the vulnerability is discovered by attackers and the first attack exploiting it, the model learns from the training set, which is like having prior knowledge of existing threats. However, the real challenge is dealing with threats it hasn't seen before, much like the zero-day attacks. The test set represents these unknown threats.

It contains data that the model has never encountered. By evaluating the model on this test set, we can get a sense of how well it might perform in identifying new and unusual network activities, mimicking the real-world scenario where it needs to detect threats it has never seen before.

Loss Function

When setting up a Recurrent Neural Network (RNN) and Deep Neural Network (DNN), training the model with a labeled dataset is crucial. This training aims to fine-tune the model's parameters, ensuring its predictions are as close as possible to the actual labels. In this process, the model processes data in batches during a phase known as the forward pass, where it generates its initial predictions. The next step involves assessing the accuracy of these predictions using a loss function, which calculates the difference between the predicted outputs and the actual labels, providing a measurable assessment of the model's accuracy.

For models dealing with multiple classes, like in multiclass classification tasks, the Categorical Cross-Entropy loss function is typically used. This function effectively measures the performance of the model in such scenarios and is elaborated in Equation as shown in the figure 25.

$$\mathbf{Loss}_{CCE} = -\frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{k} y_{ji} \cdot \log\left(p(y_{ji})\right)$$

Figure 25 Loss Function for Multiclass Classifier (Liu et al., 2018)

In this situation, (Yij)represents the actual class label for class i and observation j, The symbol N stands for the whole quantity. of observations in the batch, while K indicates the total number of distinct categories or class labels. The expression p(Yij) refers to the model's predicted probability that observation j belongs to class i as calculated by the SoftMax function

Optimizer

An optimizer's role in machine learning is to fine-tune the model's parameters (like weights and biases) to reduce the error, as indicated by the loss function. Deep neural networks often work with complex, non-linear loss functions, making it challenging to ensure convergence to the global minimum. To manage the learning rate effectively and navigate the nonconvex nature of these functions, sophisticated optimization techniques are employed. While the stochastic gradient descent is a fundamental optimization algorithm, an advanced and efficient variant often used is the Adam optimizer.

This optimizer enhances the learning process, guiding the model more reliably towards optimal performance.

Regularization

To guarantee that a deep learning model performs effectively on new, unseen data while avoiding overfitting to the training data, implementing a regularization strategy is essential. Regularization techniques modify the learning process to promote model simplicity and improve generalization. Common approaches like L1 and L2 regularization involve adding a penalty term to the cost function, leading to smaller weight values. In deep learning, a popular method is dropout regularization. This technique temporarily deactivates a certain percentage of neurons in a network layer during a single gradient update step. The proportion of neurons to drop, known as the dropout rate, is a critical hyperparameter, typically set between 10% and 50%. In our study, we've chosen a 20% dropout rate, applied to the second and third dense layers of the model, to enhance its ability to generalize well to new data.

3.5 Hyperparameters Tuning

Hyperparameters are crucial settings that need to be configured before training a deep learning model, as they significantly influence its complexity, accuracy, and training speed. Hyperparameters are set up, in contrast to model criteria, which the model learns from data during training.

Several techniques used to adjust hyperparameters of our models:

- Grid Search: This method involves exploring a range of hyperparameters to find the optimal combination. It systematically works through multiple combinations of parameter values, forming a grid, to determine the best set
- SMOTE (Synthetic Minority Over-sampling Technique): This is a statistical method used to balance datasets by increasing the number of instances in a minority class. It's primarily employed in machine learning to fix the issue of class imbalance, where certain classes in a dataset are underrepresented compared to others.

Chapter Four: Experimental Findings

Chapter Outline

4. Chapter Four Analysis and Results

This section outlines the testing framework, details the performance metrics employed for assessing the proposed models, Explanation techniques used to understand the predictions result, and discusses the outcomes of the experiments conducted on multiclass.

4.1 Analysis

In order to evaluate the effectiveness of a classifier, a variety of metrics are employed, applicable to multiclass classification scenario.

Accuracy: Accuracy is a measure used to determine the proportion of correctly classified predictions and is presented as follows(Churcher et al., 2021):

$$\text{Accuracy} = \frac{\text{Number of correct prediction}}{\text{Total number of prediction}}$$

This concept can be further elaborated by leveraging the outcomes from a confusion matrix, this is described as follows and includes True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

$$\text{Accuracy } \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Precision: measures the percentage of true positives compared to all expected positives, and the results are displayed as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall: Recall is a metric that calculates the proportion of accurately predicted positive instances relative to all outcomes in a specific class, and it can be defined as follows:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F1 Score: A balanced statistic combining recall and precision into one is the F1 score. ranging from 0 to 1. It is often considered a more comprehensive indicator of model performance compared to accuracy alone

$$\text{F1 score} = \frac{2 * (\text{recall} * \text{Precision})}{\text{recall} + \text{Precision}}$$

The choice between F1 score and accuracy as a performance metric is influenced by the distribution of data classes. In scenarios with highly imbalanced classes, the F1 score is typically more effective because it considers the data distribution and compensates for any imbalance. This makes it a more reliable metric for most real-world classification problems where unequal class distribution is common. On the other hand, accuracy is more suitable when the classes are evenly distributed, as it does not account for

distribution differences, which can sometimes result in misleading conclusions (Churcher et al., 2021).

LIME Library

The LIME Library, short for Local Interpretable Model-agnostic Explanations, is a method employed to elucidate the predictions of machine learning models (Ribeiro et al., 2016).

ROC AUC

A useful visual tool for illustrating a model's performance across various decision thresholds is the Receiver Operating Characteristic (ROC) graph. Plotting the True Positive Rate (TPR) versus the False Positive Rate (FPR) offers a clear visual representation of the model's ability to differentiate classes. TPR and FPR are calculated with the aid of certain formulas.

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

$$FPR = \frac{FP}{FP + TN}$$

Confusion Matrix

visualization technique to assess the classification's accuracy as shown in Table 8.

Table 8 Confusion Matrix

		Attack	Normal
Predicted class	**Attack**	**TP**	**FP**
	Normal	**FN**	**TN**

4.2 Results

In this section, we examine the outcome of five ML models across two categories: traditional machine learning, including Support Vector Machine (SVM) and Logistic Regression (LR), and deep learning algorithms such as Recurrent Neural Network (RNN), Deep Neural Network (DNN), and Random Forest.

Multiclass SVM classifier results

SVM model was trained on the train set which contain 75% from the whole dataset, to fix the imbalanced issue and to improve we apply SMOTE once, SMOTE can help balance the minority classes, to evaluate the model's effectiveness and reliability, it underwent evaluation using previously unseen data y-test set, also we used Grid Search with SVM to find the best parameters, some metrics used to evaluate the models as presented in table 9.

Table 9 Result of SVM-GS model and SVM_SMOTE listed per class label

Model	Attacks	Precision	Recall	F1-score	Accuracy
SVM+Smote	ICMP	54.93	59.39	57.07	78.77
	NORMAL	100	100	100	100
	SMURF	0	0	0	75.39
	TCP	50.26	97.03	66.22	75.87
overall		**52.56**	**65.02**	**56.69**	**65.02**

Based on Table 9 presents the classification performance of the Support Vector Machine (SVM) model with the Synthetic Minority Over-sampling Technique (SMOTE) across various types of network attacks, The results reveal notable variations in performance across different attack types, with the model achieving high precision and recall for NORMAL attacks but struggling to accurately classify SMURF attacks. Overall, the SVM with SMOTE technique demonstrates moderate performance, achieving an accuracy of 65.02%.

Table 10 Result of SVM-GS model listed per class label

Model	Attaacks	Pre	Recall	F1-score	Accuracy
SVM+Grid Search	ICMP	54	59	57	54
	NORMAL	100	100	100	100
	SMURF	0	0	0	0
	TCP	50	97	66	50
overall		**52**	**65**	**56**	**65**

With our trying to improve our model we applied Grid Search, table 10 presents the classification performance of the Support Vector Machine (SVM) model optimized with Grid Search, the result indicating flawless classification of normal network traffic, it struggles significantly with other attack types. Specifically, the model fails to correctly classify any instances of SMURF attacks, For ICMP and TCP attacks, the model shows moderate performance, with precision, recall, and F1-score values ranging from 50% to 66%,

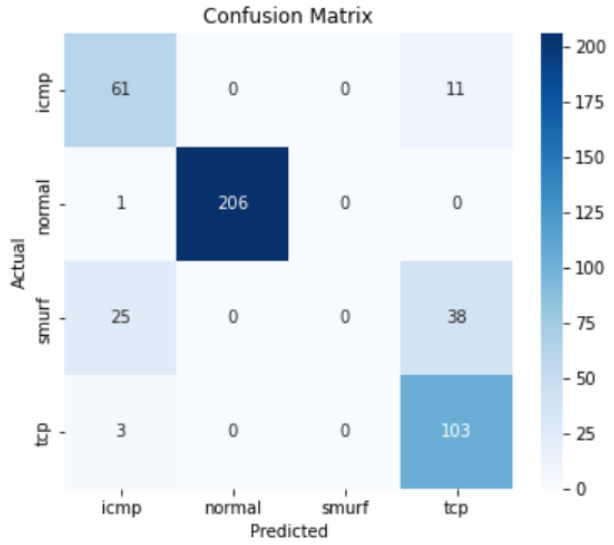


Figure 26 Confusion Matrix for SVM with Grid Search

Figure 26 above show that all "Smurf" instances classified as "Tcp" or as "Icmp" in addition the lack number of "Smurf" Class, 3 % observation from TCP classified as "ICMP", 11% of "Icmp" classified as "Tcp".

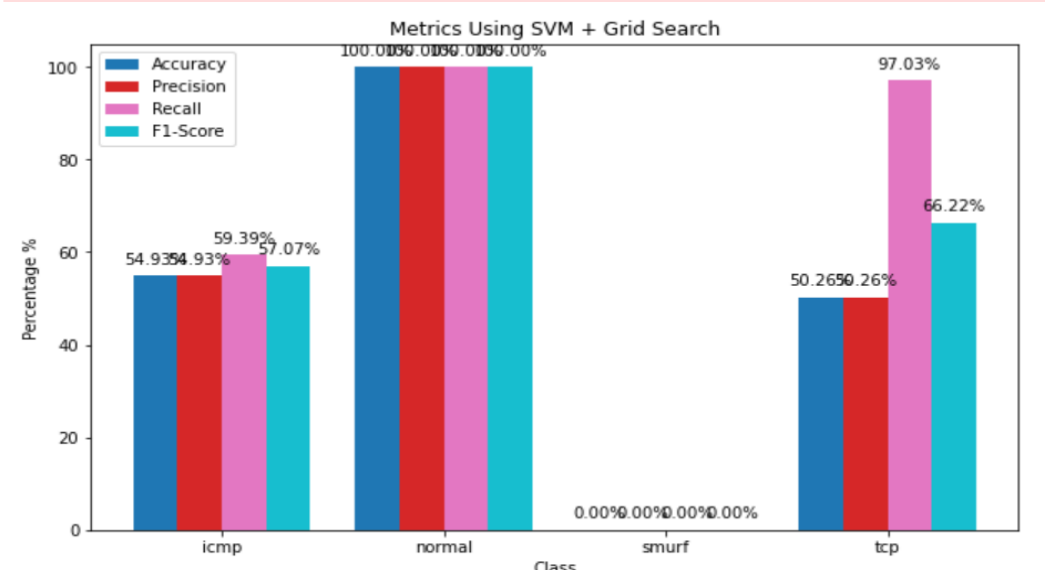


Figure 27 Evaluation of SVM + Grid Search

Figure 27 shows each metrics for each class using SVM improved with SMOTE technique, except “Normal” class which addressed 100 rate detection, as presented in the figure the “icmp” class addressed higher accuracy detection and higher precision rate with 54.93% comparison with other classes, with higher precision, the rate of false positives is reduced, though this comes at the expense of a lower recall rate. A decrease in recall suggests that the model is missing or underestimating positive cases, leading to an increase in false negatives. The "tcp" category, however, achieved a higher recall rate of 97.03%, indicating that it is more effective at correctly identifying positive instances.

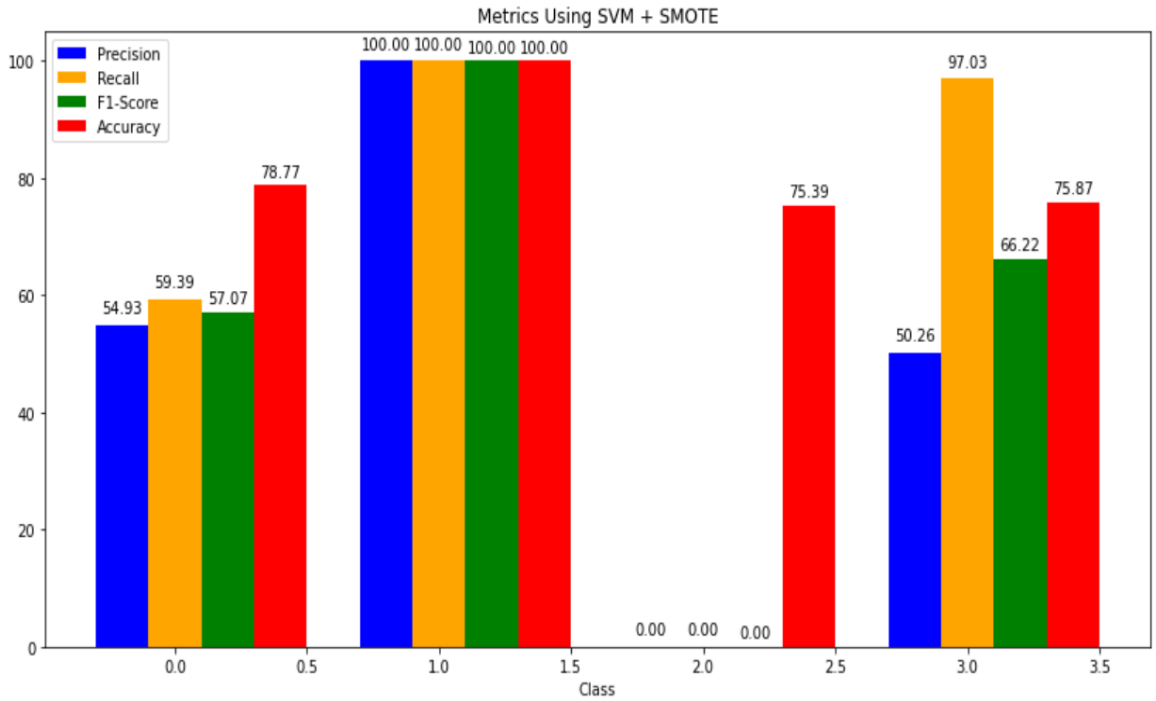


Figure 28 SVM + SMOTE

Explanation of the abnormal behavior to detect “SMURF” class:

When using SMOTE to balance classes and finding that all metrics except accuracy are 0% (with an overall accuracy of 75.39%), it suggests that SMOTE may not be creating helpful synthetic examples for the model to generalize better. Despite the decent overall accuracy, the absence of other metrics highlights a potential problem. This could be due to issues like class imbalance, difficulties with SMOTE, or problems with the SVM model itself. It's crucial to consider these factors when interpreting the model's performance.

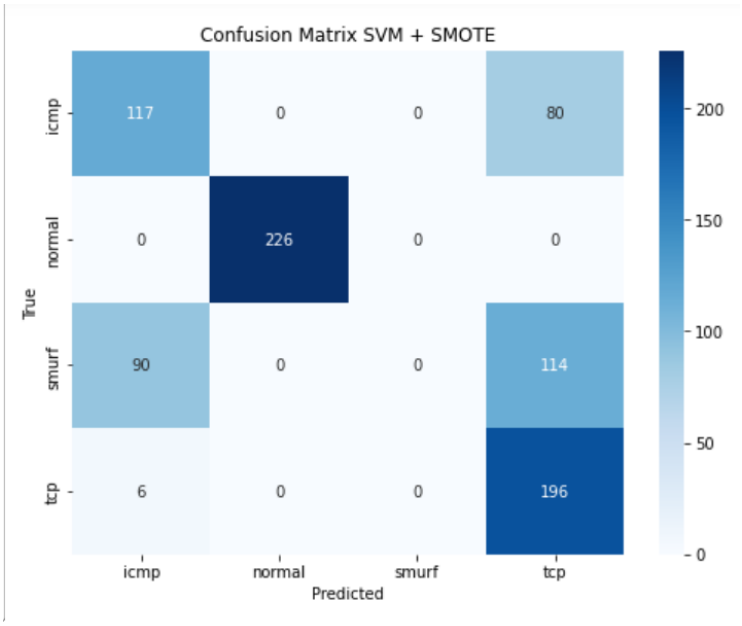


Figure 29 Confusion Matrix for SVM with SMOTE

As illustrated in the figure 29 above, most of "Smurf" observations detected as "TCP" and a little of "Smurf" classified as "ICMP", in real dataset the Smurf class is the minority one, "tcp" class addressed 97.03% higher rate, on the other hand "ICMP" class addressed moderate rates around 50% for each metrics, which means that the SVM_SMOTE model success to detect "ICMP" class and tcp class. But struggled with Smurf class.

The other traditional machine algorithm we applied over the dataset was logistic Regression (LR),

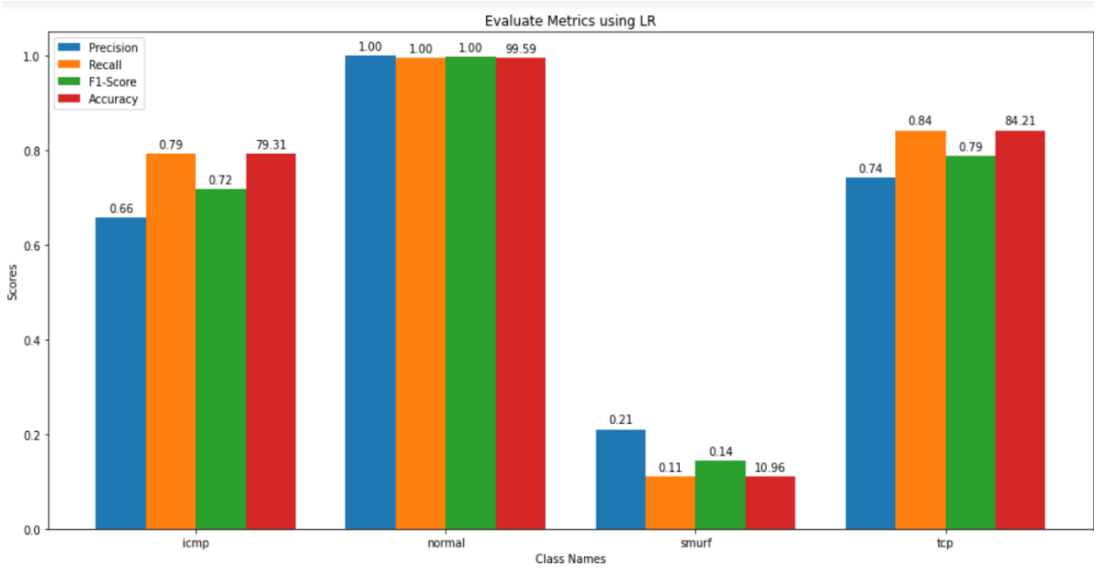


Figure 30 Evaluation Using LR

As figure 30 shows LR detect the “Smurf” class but with A low accuracy this mean the model is making a large number of incorrect predictions for the "Smurf" class compared to all classes, 11% recall means that the model is not effectively identifying a significant portion of the actual "Smurf" observations.

The model may struggle to learn patterns for the minority class “Smurf”, The logistic regression model might be too simple to capture the complexity of the "Smurf" class, especially if the decision boundary is non-linear.

Table 11 comparison between Traditional Algorithms

Model	Attacks	Prec	R	F1-score	Acc
LR	ICMP	66	79	72	79
	NORMAL	100	100	100	100
	SMURF	21	11	14	10
	TCP	74	84	79	84
overall		**65**	**69**	**66**	**80**
SVM+ Smote	ICMP	57	59	54	78
	NORMAL	100	100	100	100
	SMURF	0	0	0	75
	TCP	50	97	66	75
overall		**52**	**65**	**58**	**65**
SVM+ Grid Search	ICMP	54	59	57	54
	NORMAL	100	100	100	100
	SMURF	0	0	0	0
	TCP	50	97	66	50
overall		**52**	**65**	**56**	**65**

Table 11 presented a simple comparison between the three Traditional Algorithm used to evaluate the dataset and test the metrics, LR sign the best evaluation than SVM_SMOTE, SVM_GS.

Deep Learning Algorithms

To Get best Detection Rate Also, we applied a deep learing algorithms such as RNN,DNN and Figure 28 illustrate the evaluation using DNN enhanced with some methods.

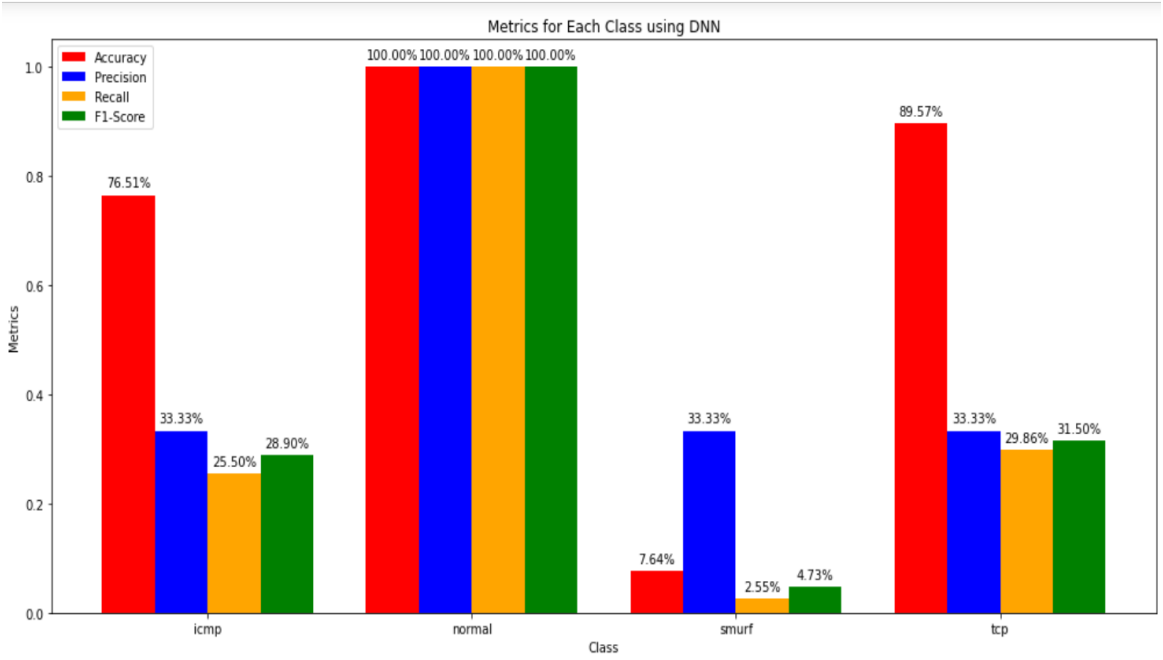


Figure 31 Evaluation of Matrices for DNN

Figure 31 above presented the evaluation of DNN , after preprocessing steps and encoding using one-hot encoding, Using non-linear activation functions like ReLU in the hidden layers help the network learn complex patterns, in our case “Smurf” class although the detection rate for this class was low at least , the DNN not facing an issue with “Smurf”, DNN success to detect all classes without applying SMOTE even it’s detection rate low scores.

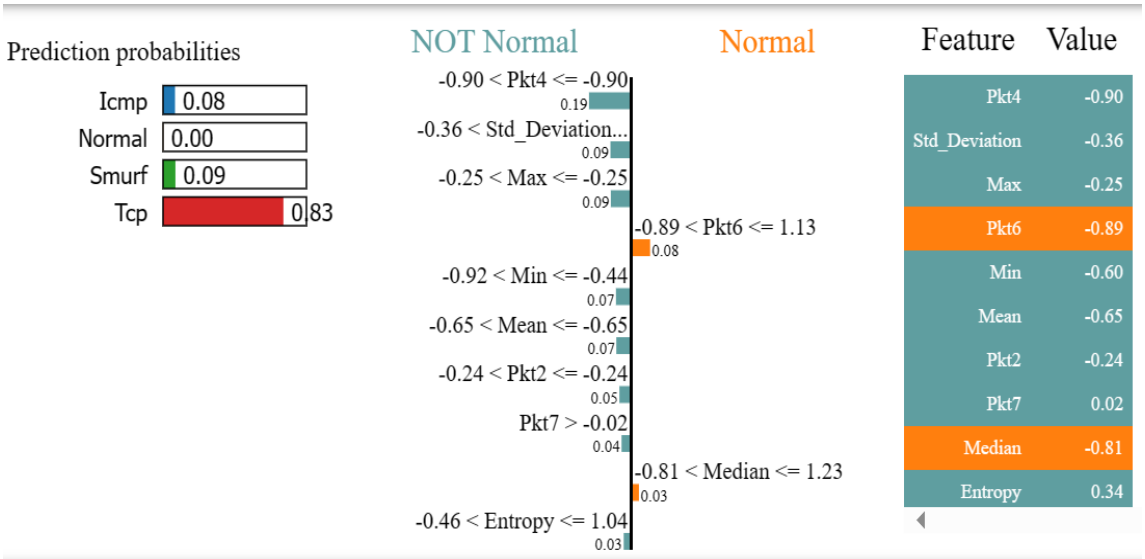


Figure 32 LIME for DNN behavior 221 instances

To understand the result of DNN model we apply LIME technique for 221 instances, for 0.83 means the model confident that the sample belongs to the “Tcp” class which is significantly higher than its confidence for the other classes. To gain a deeper understanding, LIME provides a breakdown of features from our dataset, detailing their values for this particular sample, along with the extent to which each feature influenced the model's decision-making process

Features and Values: This column shows the features in our dataset and their values for sample 221, For example, Pkt4 has a value of -0.90 for this sample.

Weights: Next to each feature, there is a weight (importance) that tells you how much each feature influenced the model's prediction. A positive weight means that the feature pushes the model towards classifying the sample as "Tcp", and a negative weight indicates the opposite. To be clear Std_Deviation with a weight of 0.09 and a value of -0.36 also contributes towards the "Tcp" prediction, but with less impact than Pkt4.

For test sample 221, the model is most confident that it belongs to the "Tcp" class. This decision is primarily influenced by features like Pkt4, Std_Deviation, Max, and others, as indicated by their weights. The values of these features and their corresponding weights collectively lead the model to lean towards the "Tcp" prediction.

We test another sample size, figure 33 presented the features impact for 344 instance,

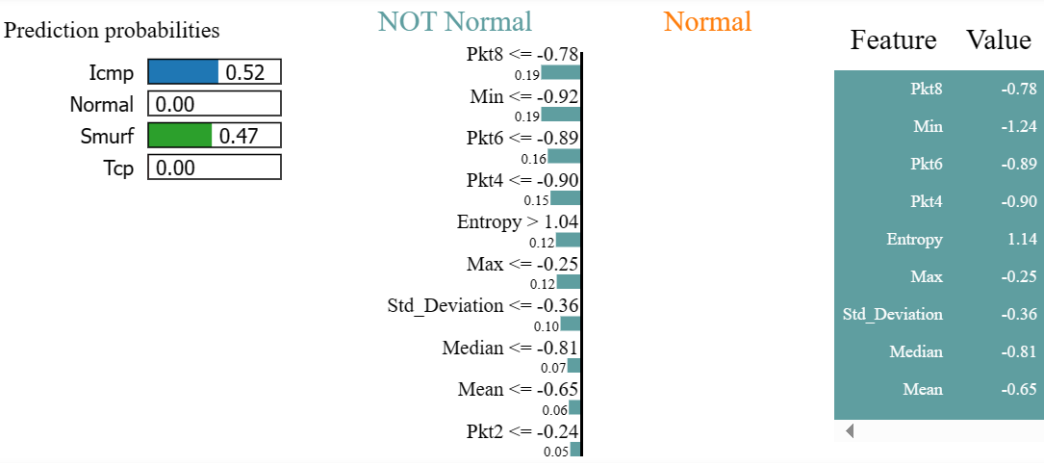


Figure 33 LIME for DNN behavior 344 instances

The model's decision for sample 344 is a close call between "Icmp" and "Smurf", with a slightly higher probability for "Icmp". The decision is primarily driven by the values of features like Pkt8, Min, Pkt6, and others. Their specific values in this sample make the model lean more towards "Icmp" than "Smurf".

To enhance the evaluation of DNN we apply DNN_SMOTE, Figure 34 illustrate the rate of DNN after applying SMOTE to overcome the imbalance “Smurf” class, it improved a little bit the performance but not too much.

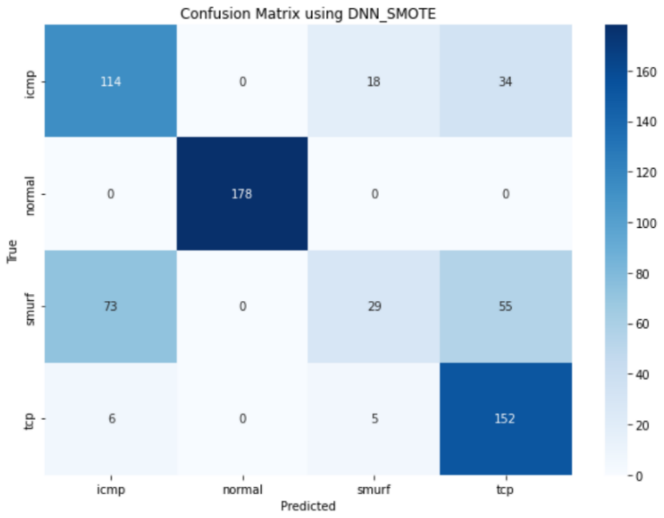


Figure 34 Confusion Matrix For DNN_SMOTE

114 represents the True Positives for “Icmp” class which is mean that 114 instances of “Icmp” were correctly identified by DNN_SMOTE as “Icmp”, 73 indicates that there were 73 instances where 'smurf' was incorrectly classified as 'icmp'. So when we want to observe the smurf, 18 instances were smurf but classified as icmp and 5 classified as tcp.

The last DNN combination we used to be DNN with Grid Search, as we mentioned before Grid Search method used to choose the best parameters to enhance the evaluation.

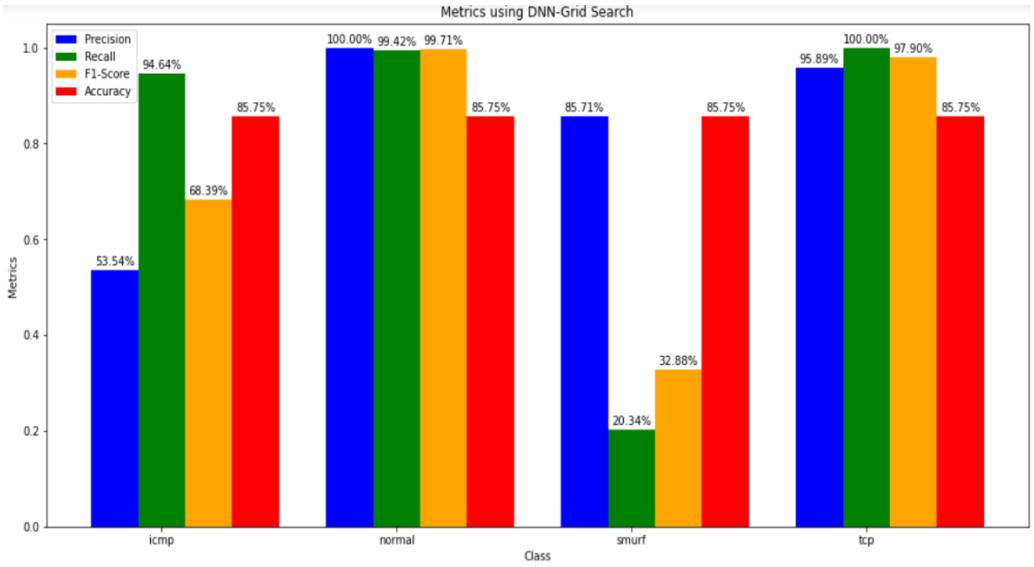


Figure 35 Evaluation of Matrices for DNN_Grid Search

DNN act like black boxs making it hard to interpret which features are most important for the classification, DNN_GS sign good detection results, with all classes as the figure above showed. Rating for “Smurf” are good too.

DNN_GS addressed low rate for recall, A 64% recall for the “Smurf” class highlight on the performance of the DNN_GS (Deep Neural Network with Grid Search) model and the nature of the data being processed. In classification tasks, recall measures the fraction of true positive observations that the model successfully identifies. Thus, a recall of 64% for the “Smurf” category indicates that the DNN_GS model has accurately identified 64% of the cases that are truly “Smurf”. This metric is vital in understanding how effectively the model is recognizing this specific class within the dataset.

The Combination DNN with Grid Seach reported best rate for all classes, and the figure below presented the ROC Curve.

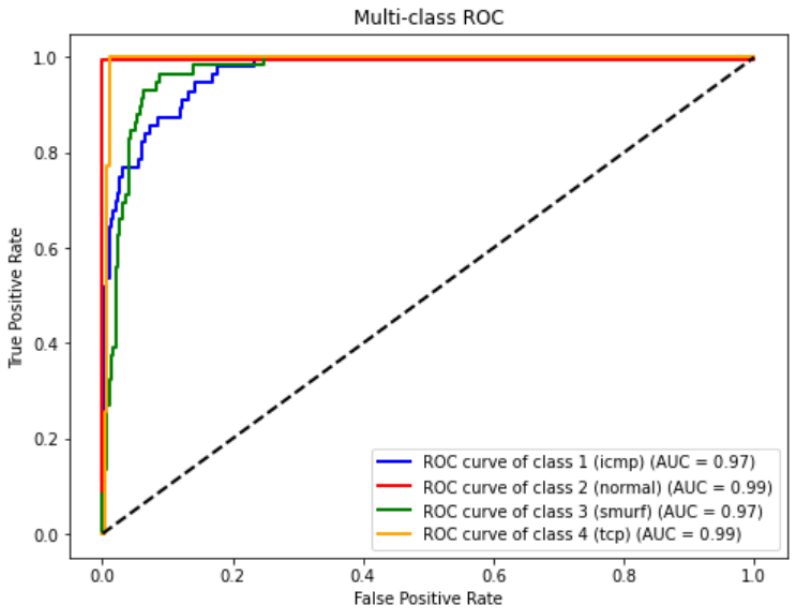


Figure 36 ROC AUC for DNN and Grid Search

High AUC Value (Normsl = 99 , Tcp = 99), The model demonstrates near-perfect accuracy in differentiating between the positive and negative classes within these categories.

The table below sammurize the comparison between DNN models performance.

Table 12 Comparison Deep learning performance

Model	Attacks	Precision	Recall	F1-score	Accuracy
DNN - GS	Icmp	70.59	85.71	77.42	91.62
	Normal	100	99	99	911.62
	Smurf	84.44	64.41	73.08	91.62
	Tcp	95.89	100	97.90	91.62
Overall		**92.62**	**87.73**	**87.39**	**87.03**
DNN-Smote	Icmp	33.33	25.50	25.50	76.52
	Normal	100	100	100	100
	Smurf	33.33	2.55	4.73	7.64
	Tcp	33.33	29.86	31.50	89.57
Overall		**50**	**39.48**	**41.28**	**69.73**
DNN	Icmp	50	47.32	48.62	94.64
	Normal	50	49.71	49.86	99.42
	Smurf	33.33	0.56	1.11	1.69
	Tcp	100	100	100	100
Overall		**58.33**	**49.40**	**49.90**	**82.68**

As table 12 shows the bad detection was from DNN compared to DNN-Smote and DNN -GS, and due to this reason, we applied DNN once with Smote and once with Grid Search, DNN_GS addressed best rating comparing with the others.

The last combination was RNN, figure 38 present the result of evaluation of RNN_LSTM_Grid search, which illustrate an issue with “Smurf” detection

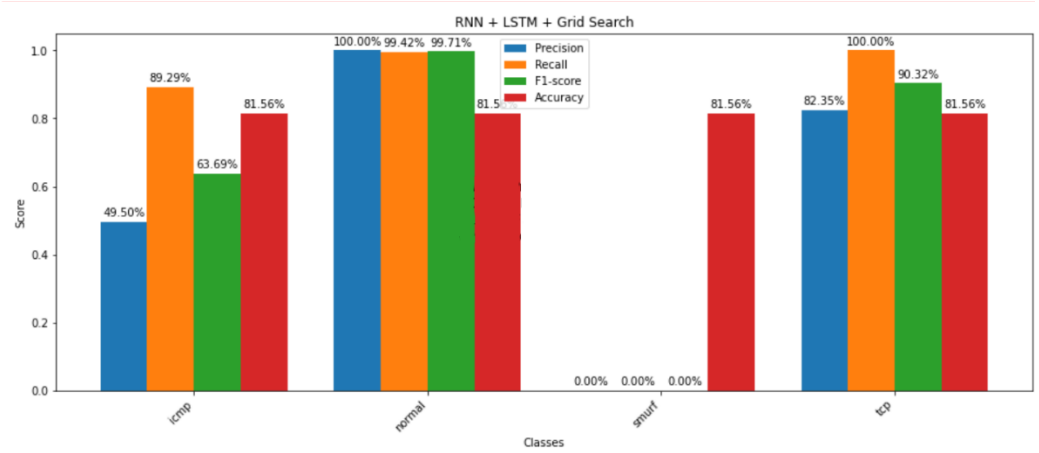


Figure 37 Evaluation of RNN_LSTM_Grid Search

As presented in figure 37 above the detection of “Smurf” class rate low rate, - for recall rate or the sensitivity A recall of 0 for the “Smurf” class indicates that the model failed to identify any actual “Smurf” instances correctly. All “Smurf” instances were missed by the model (false negatives). The model missed all “Smurf” instances.

In addition, the harmonic mean of recall and precision is known as the F1-score. A score of 0 for F1-score especially when both precision and recall are 0, indicates that the model is completely ineffective in predicting the “Smurf” class, A high overall accuracy 81% combined with poor performance in class-specific metrics like precision and recall for “Smurf” often points to a class imbalance problem, due to the few “Smurf” observations combined to other classes, the model is good at predicting the majority classes but fails with the minority “Smurf” class, after we get this result we applied RNN_Smote to overcome the imbalanced dataset.

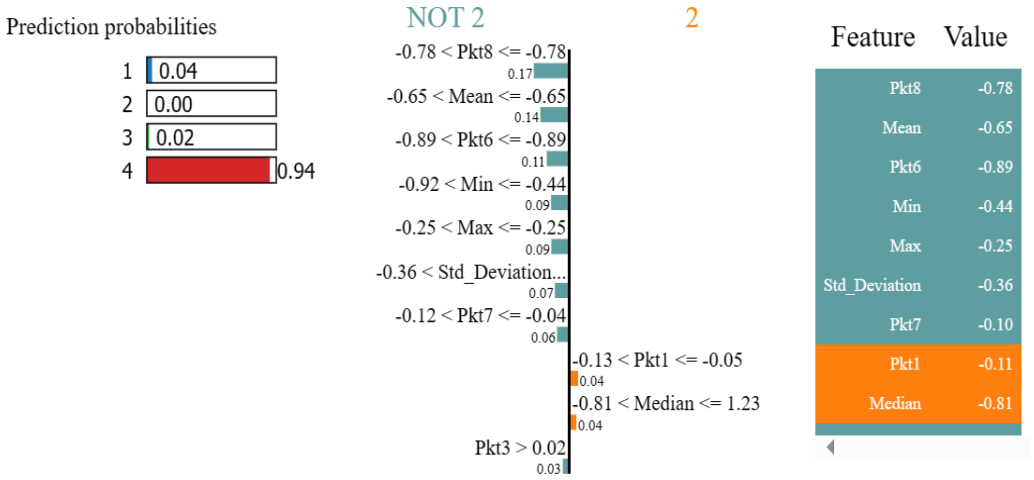


Figure 38 Explanation for RNN_LSTM_GS

94% for “Tcp” class indicates that the model is 94% confident that this instance belonge to “Tcp” class. 20% indicates that 20% instance belonge to “ Smurf” class and so on. On Feature contribution side he feature Pkt8 being in the range -0.78 <= Pkt8 <= -0.78 contributes 0.17 towards the model's final decision.

To raise the standard of performance of RNN Model, we decide to combine the RNN with SMOTE technique to Fix the imbalanced issue, and we get the following result represented in figure 31 below.

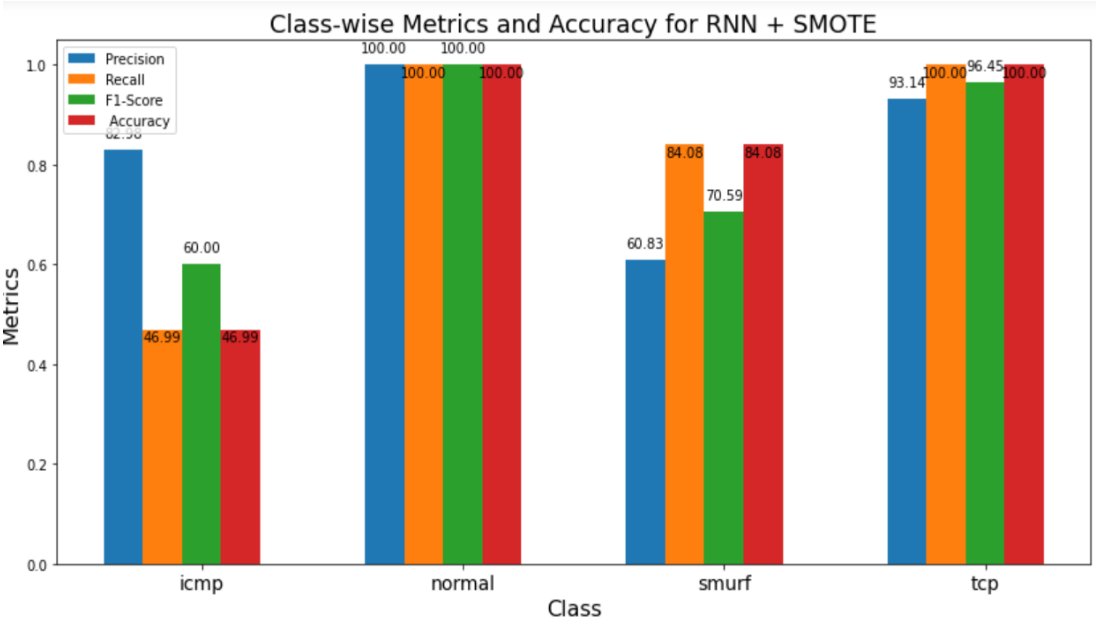


Figure 39 Evaluation of RNN combined with SMOTE

As we mentioned before, SMOTE working on minority class which is “Smurf” in our dataset, so when we fix it the detection rate get more better than RNN_LSTM_GS, and get precision 61% for “Smurf” class, 73%, 67%,73% recall,F1-score and accuracy respectively, this combination work better with the dataset

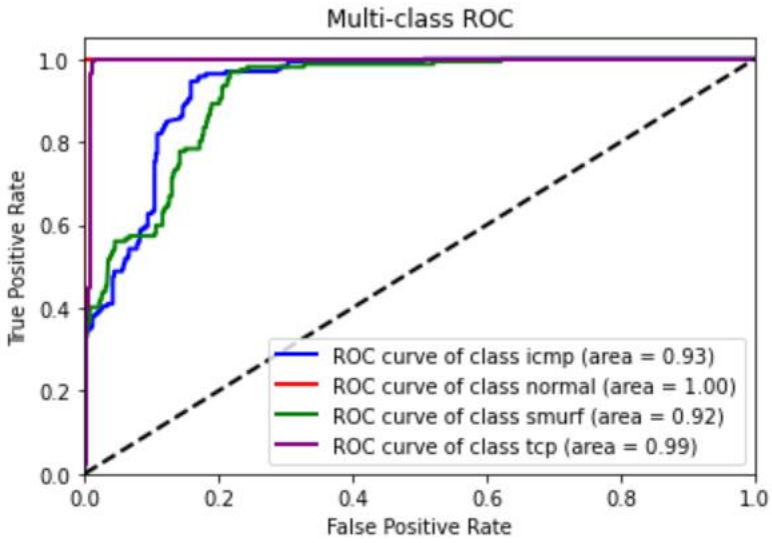


Figure 40 ROC Curve RNN_SMOTE

As figure 33 present the high score was 99% for “Tcp” class, and 100% for “Normal” class , Table 13 show the comparison between RNN Learning algorithms performance.

Table 13 comparison between RNN performance

Model	Attacks	Precision	Recall	F1-score	Acc
RNN-SMOTE	ICMP	73	54.82	62.78	54.82
	NORMAL	100	100	100	100
	SMURF	61	73.25	67.06	73.25
	TCP	92.61	100	96.17	100
OVERALL		81.96	82.02	81.49	82.02
RNN-LSTM-GS	ICMP	49.50	89.29	63.69	81.56
	NORMAL	100	99	99	81
	SMURF	0	0	0	8156
	TCP	82.35	100	90.32	81.56
OVERALL		57.96	72.18	63.43	81.56

At Last step we decide to apply Random Forest combined with Grid Search to choose the best parameters and SMOTE to fixe the imbalanced dataset, and we finally get more effective detection model as figure 34 presented below.

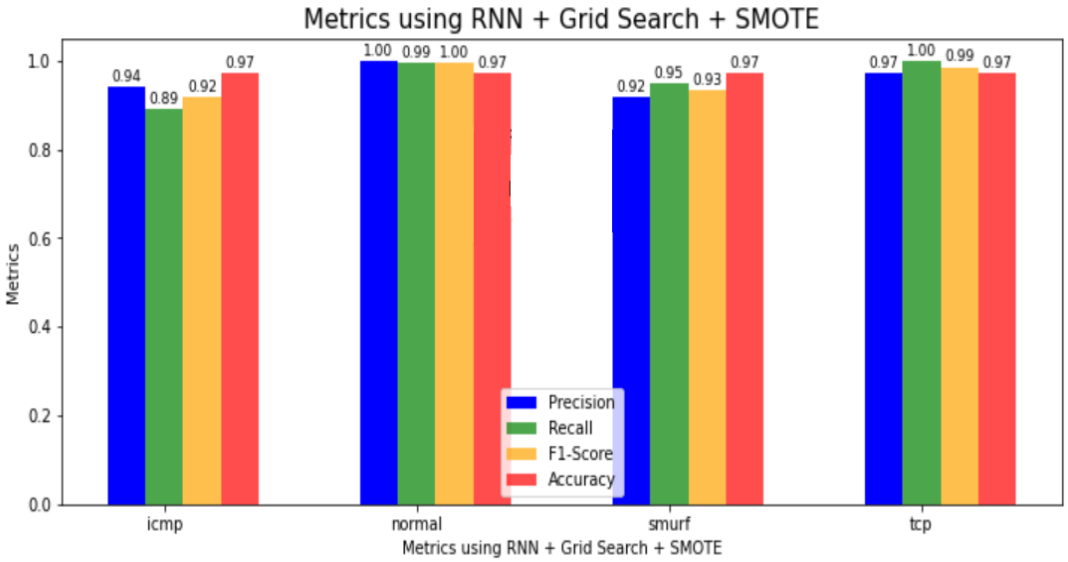


Figure 41 Random Forest _SMOTE_GS

The Random Forest reported reported accuracy around 92%, 88%,90% and 97% rate Accuracy, precision, Recall and F1-score respectively for normal Class, due to the SMOTE technique RF fixed the issue with " Smurf", RF reported 90% for precision, 93% Recall, 92% F1-score rate and finally 97% accuracy.

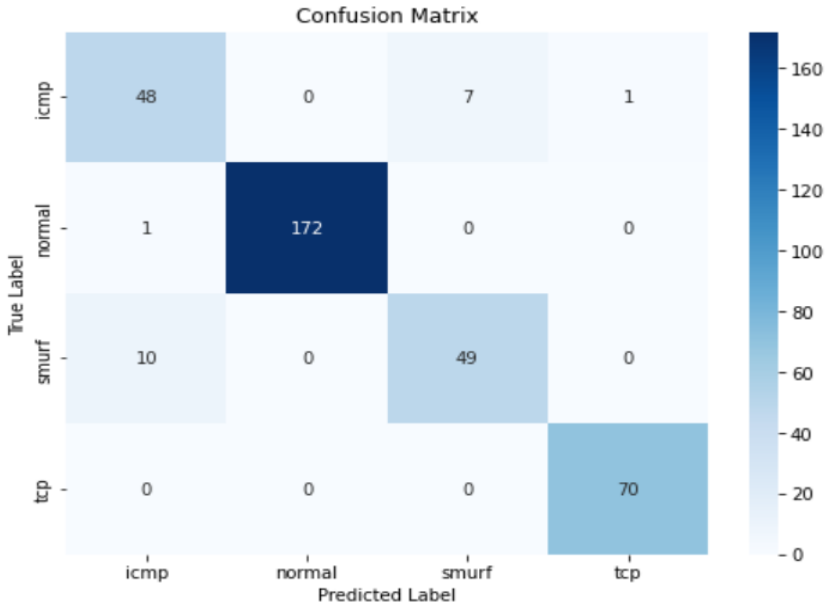


Figure 42 Confusion Matrix for Random Forest

The confusion Matrix illustrate that the RF classified all the “Tcp” instances correctly, Missed 10 instances from “Smurf” class and classified as “Icmp” class, depend on the figure 35 the Random Forest missed to classified 7 instances from “Icmp” and grouped as “Smurf” .

Table 14 the Result of Random Forest GS SMOTE Classifier

Model	Attacks	Pre	Rec	F1-score	Acc
RF_GridS_ SMOTE	ICMP	92	88	90	97
	NORMAL	100	99	100	97
	SMURF	90	93	92	97
	TCP	97	100	99	97
OVERALL		97	95	95	95

Chapter Five: Conclusion

Chapter Outline

5.1 Analysis of the Findings

5.2 Conclusion

5.3 Future Work

5. Chapter Five Discussion and Conclusion

The experiment's findings allow for the conclusion of the following

5.1 Discussion of the Results:

Imbalance Dataset:

Dealing with an imbalanced dataset, especially in a multi-class scenario like our dataset, is a common challenge in machine learning, Imbalanced data can significantly impact the performance of machine learning algorithms, often leading to poor classification of the minority class, to handle this issue we use SMOTE technique to oversampling the Minority Class, by increasing the number of instances in the minority to balance the dataset in our case “Smurf” class.

Create DoS

In this work we try to create different type of DoS attacks, each one relates to different root, in this work we create “Icmp” which belong to Volume based attack types, “TCP” and “Smurf” which belong to protocol based attack.

Feature Selection

In our dataset, which comprises 14 features (pk1, pk2, pk3, pk4, pk5, pk6, pk7, pk8, Stream, min, max, mean, median, std_deviation, entropy), the quantity of input features plays a crucial role in influencing both the performance and the capacity of the model. Each algorithm implemented endeavors to select the most impactful features that contribute positively. Notably, the Random Forest algorithm demonstrates its effectiveness in anomaly detection by efficiently utilizing these features.

Hyperparameters tuning

In our work we used grid search method to choose best parameters, best parameters will improve the performance of deep learning Models.

Performance metrics

Performance metrics are crucial in diagnosing issues and enhancing the effectiveness of a model, as well as facilitating comparative analyses. The Receiver Operating Characteristic (ROC) curve and the confusion matrix are instrumental tools for visualizing a model's performance. The Area Under the Curve (AUC) is an important metric that quantifies the classifier's ability to differentiate between classes.

Comparison with the literature Reviews

In this section I presents a comparison between my results and the previous work result based on the similarities and the differences, The comparison aims to contextualize my findings within the existing body of research and highlight any novel contributions made by my study

Table 15 Comparison with the litreture Reviews

Similarities						
My Study			Pre-Work			
Feature	Data set	Methods		Feature	Data set	Methods
Inter Arrival Time			(Abdalla et al., 2018)	Inter Arrival Time		
	SYN and ICMP	LSTM_ RNN	(Ullah et al., 2023)		DOS type: SYN, ICMP	-CNN-LSTM -CNN-RNN -
	UDP, ICMP flood	Random Forest	(Garsva et al., 2014)		UDP, ICMP flood	Random Forest
	Random Forest SVM		(Garcia & Blandon, 2022)			support vector machine SVM
Entropy	Random Forest		(Baldini et al., 2021)	Entropy		Random Forest
Differences						
	TCP, ICMP, SMURF Flood		(Abdalla et al., 2018)		CIC-IDS2017: this dataset has many attacks include the Dos Type (HTTP flood)	Support Vector Machine, Decision Tree, and Naive Bayes
	Smurf	DNN, SVM, LR Random Forest	(Ullah et al., 2023)		NSL-KDD: it has 4 types of attack include DOS UDP	-CNN-LSTM -CNN-RNN -CNN-GRU
IAT	Smurf	DNN, SVM, RNN	(Garsva et al., 2014)	IAT	UNSW-NB15 has many attacks not only DoS	
	Collected data only	DNN, SVM, RNN	(Kurniawan et al., 2021)		Collected data include	statistical analysis methods

	for DoS	Random Forest			d DoS Attacks	
	Collected data only for DoS	DNN, RNN, LR Random Forest	(Garcia & Blandon, 2022)		SNMP (Simple Network Management Protocol)	decision trees, one-class support vector machine, subspace clustering, replicator NN
Entropy and IAT	Real Data	DNN, RNN, LR, SVM	(Baldini et al., 2021)	Entropy	UNSW-NB15	

The comparative analysis between our study and the pre-work highlights several similarities and differences in terms of the features studied, datasets used, and methods applied. Both studies focus on analyzing features such as Inter Arrival Time (IAT), Entropy and the statistical features of each type of attack to understand cybersecurity threats behavior. In both cases, machine learning methods including Random Forest, Support Vector Machine (SVM), and Deep Neural Networks (DNN) are employed to process the data. This similarity in feature selection and analytical methods suggests a shared objective of identifying patterns and anomalies in network traffic to enhance cybersecurity measures.

However, significant differences emerge in the choice of datasets and the specific types of attacks considered Our research attempted to produce a few DoS assaults in a simulated setting and collect these attacks into a dataset on the other side the pre-work focuses on datasets like NSL-KDD and CIC-IDS2017, UNSW_NB15 which are more tailored to DoS attacks and include specific attack types such as TCP, ICMP, and

SMURF Floods. This discrepancy in dataset selection reflects differing research objectives and priorities in terms of attack detection and mitigation strategies.

Furthermore, the methods employed in each study vary, with our research incorporating a mix of traditional statistical analysis methods and advanced machine learning techniques, while the pre-work leans towards more complex algorithms like Deep Neural Networks (DNN) and Decision Trees, these methodological disparities suggest diverse approaches to addressing cybersecurity challenges with our study emphasizing interpretability and the pre-work focusing on predictive accuracy and scalability**.**

Overall, while both studies contribute valuable insights into cybersecurity threat detection, their differing methodologies and dataset choices offer complementary perspectives on the multifaceted nature of network security**.**

5.2 Conclusion

In this study, our objective is to evaluate multiple machine learning algorithms using actual DoS traffic data. We gathered this traffic by establishing a Mininet environment on a VMware machine. We generated different types of DoS flood attacks using hping3 and captured the network traffic using Wireshark, saving it in Pcap files. After labeling the dataset, we merged all Pcap files into a single CSV file. This file then applies preprocessing steps to prepare it for the application of both traditional machine learning and deep learning algorithms.

The primary objective of our research was to determine the effectiveness of using inter-arrival times and statistical features to detect DoS attacks in secure channel. To enhance performance, we employed advanced techniques such as Grid Search and SMOTE. However, certain algorithms encountered challenges in accurately identifying the minority class, which was partly attributed to the limited size of our dataset, Prior research in the area brought to light a few difficulties, including poor detection accuracy, a high rate of false alarms, and the failure to identify unidentified attacks.

The hyperparameters of the model need to be optimized to minimize training error. While the model's parameters are derived from the data, hyperparameters must be established beforehand. Techniques like grid search and random search are utilized to identify the most effective hyperparameters.

The best model addressed high performance metrics was RNN after Appling SMOTE technique to fix the imbalanced issue with 89% precision, 91.50% recall, 90.50% f1-score, 86.25% accuracy. Most of algorithms faced issue with the minority class although after applying the SMOTE technique.

In summary, Machine Learning algorithms stand out as potent instruments in the construction of predictive models. Their strength lies in their ability to discern patterns within data and effectively approximate functions that link inputs to desired outputs. These algorithms demonstrate high proficiency as classifiers, particularly in accurately detecting and identifying diverse patterns of anomalies in both familiar and unfamiliar network traffic

Additionally, the utilization of statistical features and the analysis of the time intervals between packets further contribute to the robust detection of anomalies.

5.3 Recommendations and Future Work

For future expansion of this study, several tracks could be explored. Firstly, the dataset size could be significantly increased. While our current dataset groups every eight packets into one stream, future datasets could consider streams comprising 128 or even 1024 packets. The overall dataset size could be scaled up to approximately 1 million entries. Secondly, future studies might incorporate the CICFlowMeter tool to compute a broader range of statistical features, potentially expanding the current set of 6 features. Thirdly, the scope of the dataset could be broadened to encompass more than three types of DoS attacks, providing a more comprehensive analysis.

We attempted to experiment both traditional machine learning algorithms and deep learning techniques. However, it might be more advantageous to explore algorithms specifically tailored for time series data, such as those involving time series analysis methods. This approach could potentially yield more accurate and efficient results for our dataset.

References

J. H. Seo, “Efficient digital signatures from RSA without random oracles,” *Inf. Sci. (Ny).*, vol. 512, 2020, doi: 10.1016/j.ins.2019.09.084.

I. Hidayat, M. Z. Ali, and A. Arshad, “Machine Learning-Based Intrusion Detection System: An Experimental Comparison,” *J. Comput. Cogn. Eng.*, vol. 2, no. July 2022, pp. 88–97, 2022, doi: 10.47852/bonviewjcce2202270.

Al-Morjan, “An investigation into a digital forensic model to distinguish between ‘insider’ and ‘outsider,’” pp. 1–299, 2010, [Online]. Available: http://www.libsearch.com/visit/608696%5Cnhttp://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:An+investigation+into+a+digital+forensic+model+to+distinguish+between+?insider?+and+?outsider?#0.

M. M. Rashid, J. Kamruzzaman, M. M. Hassan, T. Imam, and S. Gordon, “Cyberattacks detection in iot-based smart city applications using machine learning techniques,” *Int. J. Environ. Res. Public Health*, vol. 17, no. 24, pp. 1–21, 2020, doi: 10.3390/ijerph17249347.

R. Cziva, S. Jouët, D. Stapleton, F. P. Tso, and D. P. Pezaros, “SDN-Based Virtual Machine Management for Cloud Data Centers,” *IEEE Trans. Netw. Serv. Manag.*, vol. 13, no. 2, pp. 212–225, 2016, doi: 10.1109/TNSM.2016.2528220.

S. Qureshi and R. Braun, “Mininet Topology: Mirror of the Optical Switch Fabric,” *2019 29th Int. Telecommun. Networks Appl. Conf. ITNAC 2019*, 2019, doi: 10.1109/ITNAC46935.2019.9078014.

S. M. Mousavi and M. St-Hilaire, “Early detection of DDoS attacks against SDN controllers,” *2015 Int. Conf. Comput. Netw. Commun. ICNC 2015*, pp. 77–81, 2015, doi: 10.1109/ICCNC.2015.7069319.

M. Naveed *et al.*, “A Deep Learning-Based Framework for Feature Extraction and Classification of Intrusion Detection in Networks,” *Wirel. Commun. Mob. Comput.*,

vol. 2022, no. 3, 2022, doi: 10.1155/2022/2215852.

N. N. Dao, Q. D. Tran, M. Park, and S. Cho, “SDNbox: A portable open-source testbed for SDN study,” *Int. Conf. Inf. Commun. Technol. Converg. ICT Converg. Technol. Lead. Fourth Ind. Revolution, ICTC 2017*, vol. 2017-Decem, pp. 829–833, 2017, doi: 10.1109/ICTC.2017.8190793.

T. Abdullah, “Тестування Floodlight-Контролеру З Mininet В Sdn Топології,” *ScienceRise*, vol. 5, no. 2(5), p. 68, 2014, doi: 10.15587/2313-8416.2014.31734.

J. Mchugh, A. Christie, and J. Allen, “Defending Youself : The Role of IDS - IEEE software,2000,” *IEEE Trans. Netw. Serv. Manag.*, vol. 1, no. October, p. 10, 2000.

U. Amin, A. S Ahanger, F. Masoodi, and A. M Bamhdi, “Ensemble based Effective Intrusion Detection System for Cloud Environment over UNSW-NB15 Dataset,” *Scrs Conf. Proc. Intell. Syst.*, no. April, pp. 483–494, 2021, doi: 10.52458/978-93-91842-08-6-46.

K. A. Dhanya, S. Vajipayajula, K. Srinivasan, A. Tibrewal, T. S. Kumar, and T. G. Kumar, “Detection of Network Attacks using Machine Learning and Deep Learning Models,” *Procedia Comput. Sci.*, vol. 218, pp. 57–66, 2023, doi: 10.1016/j.procs.2022.12.401.

T. Sayyed, S. Kodwani, K. Dodake, and M. Adhayage, “Intrusion Detection System,” *Int. J. Aquat. Sci.*, vol. 14, no. 01, pp. 39–57, 2023, doi: 10.1142/9781848164482_0004.

E. Akyildirim, M. Gambara, J. Teichmann, and S. Zhou, “Applications of Signature Methods to Market Anomaly Detection,” *arxiv*, vol. 41, no. 3, pp. 1–32, 2022, [Online]. Available: http://arxiv.org/abs/2201.02441.

W. Bul’ajoul, A. James, and S. Shaikh, “A New Architecture for Network Intrusion

Detection and Prevention," *IEEE Access*, vol. 7, pp. 18558–18573, 2019, doi: 10.1109/ACCESS.2019.2895898.

M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should i trust you?' Explaining the predictions of any classifier," *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, vol. 13-17-Augu, pp. 1135–1144, 2016, doi: 10.1145/2939672.2939778.

M. Awad, S. Fraihat, K. Salameh, and A. Al Redhaei, "Examining the Suitability of NetFlow Features in Detecting IoT Network Intrusions," *Sensors*, vol. 22, no. 16, pp. 1–18, 2022, doi: 10.3390/s22166164.

V. Ndatinya, Z. Xiao, V. R. Manepalli, K. Meng, and Y. Xiao, "Network forensics analysis using Wireshark," *Int. J. Secur. Networks*, vol. 10, no. 2, pp. 91–106, 2015, doi: 10.1504/IJSN.2015.070421.

M. V. Pawar and J. Anuradha, "Network security and types of attacks in network," *Procedia Comput. Sci.*, vol. 48, no. C, pp. 503–506, 2015, doi: 10.1016/j.procs.2015.04.126.

Z. R. Alashhab, M. Anbar, M. M. Singh, I. H. Hasbullah, P. Jain, and T. A. Al-Amiedy, "Distributed Denial of Service Attacks against Cloud Computing Environment: Survey, Issues, Challenges and Coherent Taxonomy," *Appl. Sci.*, vol. 12, no. 23, 2022, doi: 10.3390/app122312441.

F. Hussain and D. Nashat, "Time Series Similarity for Detecting DDoS Flooding Attack," *Assiut Univ. J. Multidiscip. Sci. Res.*, vol. 51, no. 3, pp. 229–241, 2022, doi: 10.21608/aunj.2022.129373.1004.

F. O. Catak and A. F. Mustacoglu, "Distributed denial of service attack detection using autoencoder and deep neural networks," *J. Intell. Fuzzy Syst.*, vol. 37, no. 3, pp. 3969–3979, 2019, doi: 10.3233/JIFS-190159.

Y. Wang, J. Ding, T. Zhang, Y. Xiao, and X. Hei, “From Replay to Regeneration: Recovery of UDP Flood Network Attack Scenario Based on SDN,” *Mathematics*, vol. 11, no. 8, pp. 1–22, 2023, doi: 10.3390/math11081897.

O. E. Elejla, M. Anbar, S. Hamouda, S. Faisal, A. A. Bahashwan, and I. H. Hasbullah, “Deep-Learning-Based Approach to Detect ICMPv6 Flooding DDoS Attacks on IPv6 Networks,” *Appl. Sci.*, vol. 12, no. 12, 2022, doi: 10.3390/app12126150.

M. Darwish, A. Ouda, and L. F. Capretz, “Cloud-based DDoS Attacks and Defenses,” *Int. Conf. Inf. Soc. (i-Society 2013), 2013*, p. 6, 2013.

D. J. S. Raja, R. Sriranjani, A. Parvathy, and N. Hemavathi, “A Review on Distributed Denial of Service Attack in Smart Grid,” *7th Int. Conf. Commun. Electron. Syst. ICCES 2022 - Proc.*, vol. 16, no. 4, pp. 812–819, 2022, doi: 10.1109/ICCES54183.2022.9835859.

M. Dewidar, A. Selvaraj, and V. Lin, “Measuring Global Risk of ICMP Amplification Attacks.”

F. F. Ashrif, E. A. Sundararajan, R. Ahmad, M. K. Hasan, and E. Yadegaridehkordi, “Survey on the authentication and key agreement of 6LoWPAN: Open issues and future direction,” *J. Netw. Comput. Appl.*, vol. 221, no. December 2022, p. 103759, 2024, doi: 10.1016/j.jnca.2023.103759.

D. S. Wibowo, H. Z. Ilmadina, A. S. Ardi, and F. F. G. Insani, “Apache web server security with security hardening,” *J. Soft Comput. Explor.*, vol. 4, no. 4, pp. 213–221, 2023, doi: 10.52465/joscex.v4i4.230.

M. Wang, N. Yang, and N. Weng, “Securing a Smart Home with a Transformer-Based IoT Intrusion Detection System,” *Electron.*, vol. 12, no. 9, pp. 1–19, 2023, doi: 10.3390/electronics12092100.

I. H. Sarker, “Machine Learning: Algorithms, Real-World Applications and Research Directions,” *SN Comput. Sci.*, vol. 2, no. 3, pp. 1–21, 2021, doi: 10.1007/s42979-021-00592-x.

P. M. Guarango, “No DoS , vol. 54, no. 8.5.2017, pp. 2003–2005, 2022.

C. Robertson, J. L. Wilmoth, S. Retterer, and M. Fuentes-Cabrera, “Video frame prediction of microbial growth with a recurrent neural network,” *Front. Microbiol.*, vol. 13, 2023, doi: 10.3389/fmicb.2022.1034586.

J. Note and M. Ali, “Comparative Analysis of Intrusion Detection System Using Machine Learning and Deep Learning Algorithms,” *Ann. Emerg. Technol. Comput.*, vol. 6, no. 3, pp. 19–36, 2022, doi: 10.33166/AETiC.2022.03.003.

N. Moustafa and J. Slay, “UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” *2015 Mil. Commun. Inf. Syst. Conf. MilCIS 2015 - Proc.*, no. November, 2015, doi: 10.1109/MilCIS.2015.7348942.

T. Janarthanan, “Feature Selection in UNSW-NB15 and KDDCUP ’ 99 datasets,” *Sheff. hallam Univ.*, vol. 3, no. 2, p. 7, 2017.

G. Baldini, J. L. H. Ramos, and I. Amerini, “Intrusion detection based on gray-level co-occurrence matrix and 2d dispersion entropy,” *Appl. Sci.*, vol. 11, no. 12, 2021, doi: 10.3390/app11125567.

F. Ullah, S. Ullah, G. Srivastava, and J. C.-W. Lin, “IDS-INT: Intrusion detection system using transformer-based transfer learning for imbalanced network traffic,” *Digit. Commun. Networks*, 2023, doi: 10.1016/j.dcan.2023.03.008.

A. A. Kurniawan, Jusak, and Musayyanah, “Intrusion Detection System Using Deep Learning for DoS Attack Detection,” *JEECS (Journal Electr. Eng. Comput. Sci.*, vol. 6, no. 2, pp. 1087–1098, 2021, doi: 10.54732/jeecs.v6i2.203.

E. Garsva, N. Paulauskas, G. Grazulevicius, and L. Gulbinovic, “Packet inter-arrival time distribution in academic computer network,” *Elektron. ir Elektrotechnika*, vol. 20, no. 3, pp. 87–90, 2014, doi: 10.5755/j01.eee.20.3.6683.

B. M. A. Abdalla, M. Hamdan, M. S. Mohammed, J. S. Bassi, I. Ismail, and M. N. Marsono, “Impact of packet inter-arrival time features for online peer-to-peer (P2P) classification,” *Int. J. Electr. Comput. Eng.*, vol. 8, no. 4, pp. 2521–2530, 2018, doi: 10.11591/ijece.v8i4.pp2521-2530.

J. F. C. Garcia and G. E. T. Blandon, “A Deep Learning-Based Intrusion Detection and Preventation System for Detecting and Preventing Denial-of-Service Attacks,” *IEEE Access*, vol. 10, no. August, pp. 83043–83060, 2022, doi: 10.1109/ACCESS.2022.3196642.

R. R. Fontes, S. Afzal, S. H. B. Brito, M. A. S. Santos, and C. E. Rothenberg, “Mansdn-Cnsm15-Mininet-Wifi-Ramon.Pdf,” *IEEE Access*, 2015.

O. Flauzac, E. M. Gallegos Robledo, and F. Nolot, “Is mininet the right solution for an SDN testbed?,” *2019 IEEE Glob. Commun. Conf. GLOBECOM 2019 - Proc.*, no. December 2019, pp. 1–6, 2019, doi: 10.1109/GLOBECOM38437.2019.9013145.

F. Suthar and N. Patel, “A Survey on DDoS Detection and Prevention Mechanism,” *J. Adv. Inf. Technol.*, vol. 14, no. 3, pp. 444–453, 2023, doi: 10.12720/jait.14.3.444-453.

E. Papadogiannaki, G. Tsirantonakis, and S. Ioannidis, “Network Intrusion Detection in Encrypted Traffic,” *5th IEEE Conf. Dependable Secur. Comput. DSC 2022 SECSOC 2022 Work. PASS4IoT 2022 Work. SICSA Int. Pap. Compet. Cybersecurity*, vol. 3, no. June, 2022, doi: 10.1109/DSC54232.2022.9888942.

G. P. Fernando, A. A. H. Brayan, A. M. Florina, C. B. Liliana, A. M. Hector-Gabriel, and T. S. Reinel, “Enhancing Intrusion Detection in IoT Communications Through ML Model Generalization With a New Dataset (IDSAI),” *IEEE Access*, vol. 11, no. May, pp. 70542–70559, 2023, doi: 10.1109/ACCESS.2023.3292267.

A. Nichelini, C. A. Pozzoli, S. Longari, M. Carminati, and S. Zanero, “CANova: A hybrid intrusion detection framework based on automatic signal classification for CAN,” *Comput. Secur.*, vol. 128, p. 103166, 2023, doi: 10.1016/j.cose.2023.103166.

Y. Liu, S. Tang, R. Liu, L. Zhang, and Z. Ma, “Secure and robust digital image watermarking scheme using logistic and RSA encryption,” *Expert Syst. Appl.*, vol. 97, pp. 95–105, 2018, doi: 10.1016/j.eswa.2017.12.003.

A. Churcher *et al.*, “An experimental analysis of attack classification using machine learning in IoT networks,” *Sensors (Switzerland)*, vol. 21, no. 2, pp. 1–32, 2021, doi: 10.3390/s21020446.

N. Upreti, “DDoS Attack and Mitigation,” no. April, p. 42, 2019, [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/168652/DDoS Final Version.pdf?sequence=2%0Ametropolia.fi/en.

M. D. K, “by,” no. November, 2012.

الملخص

نظراً الى أهميه مصطلح حماية البيانات في عالم الشبكات والمعلومات فقط دأب المهتمين الى تشفير بياناتهم لتسهيل مرورها عبر القنوات الشبكية المشفرة , مما زاد من صعوبة تميز حمولة البيانات العادية عن اي حمولة اخرى ممكن ان تعتبر نوع من انواع (هجوم منع من الخدمه) ومن هنا حرص الباحثون على تطوير أنظمة كشف التسلل بمساعدة علم الخوارزميات العميقة لتقليل توليد الانذار الخاطئ.

للمساعده في حل المشكلة السابقة ولتسهيل عملية اكتشاف انواع هجمات منع الخدمه , قمنا بهذه الاطروحة ب انشاء انواع مختلفه من هجمة منع الخدمة و تطبيق خوارزميات معدله على البيانات بعد حساب المميزات الاحصائية بالاعتماد على الفرق الزمني بين الحمولات المارة عبر القناة المشفرة.

قمنا ب اختبار انواع متعدده من الخوارزميات التقليدية واخرى عميقة التعلم من اجل فحص كفاءتها بتمييز أربعة انواع من الهجمات, ومن اجل تحسين دقة الكشف تم حساب وهي فرق التوقيت الزمني بين كل حمولتين متتاليتين , ثم تم جمع كل ثمانية حمولات في قناة واحده وتم حساب المميزات الاحصائية لكل قناة وبناء على المميزات الاحصائية تم عمل تعريف لكل قناة وجمع هذه الانواع في ملف csv واحد ليتم معالجتها رياضيا و تطبيق خوارميات التعلم العميق و الخوارزميات التقليدية.

أكدت النتائج التجريبية أن خوارزمية RNN_SMOTE قد اثبتت كفائتها بكشف انواع الهجمات المختلفه بنسب عالية, وتتكون هذه الخوارزمية من طبقة ادخال واحده و ثلاث طبقات مخفية وطبقة اخراج واحده, حقق النموذج نسبة دقه 86.25% باكتشاف الاربعة انواع و لم يواجه النموذج صعوبة ب اكتشاف هجمه Smurf في حين أن معظم النماذج واجهو مشكلة في اكتشاف هذه الهجمه و يعود لك الى ان عددها كان قليل جدا في قاعدة البيانات و بالرغم من

تطبيق SMOTE الذي يعمل على نسخ عدد من مشاهدات النوع القليل ليتم مساواته بالانواع الثانية الا ان بعض الخوارزميات لم تستطع اكتشافها بكفاءة.

عند تطبيق Confusion Matrix على بعض الخوارزميات كانت هجمة Smurf تصنف على انها TCP بالغالب . أيضا تطبيق نموذج Random Forest كان له نسب عالية الدقه بالاضافه الى استعمال Grid Search مع بعض الخوارزميات لتحسين دقة الكشف عبر تحديد افضل المؤشرات.