University of Staffordshire

Detection of Volumetric ICMPv6 DDoS attack using Ensemble Stacking on Deep Neural Network

Om Vasu Prakash Salamkayala

A thesis submitted in fulfilment of the Doctorate of Philosophy in the School of Digital, Technology, Innovation and Business at the University of Staffordshire.

29 May 2025

Abstract

The internet serves as a vital hub for information exchange, seamlessly intertwining with our daily lives. Operating on IPv6 and IPv4 protocols, it facilitates connections between sources and destinations. However, these protocols harbour vulnerabilities, particularly evident in Internet Control Message Protocol version 6 (ICMPv6), making it susceptible to Distributed Denial of Service (DDoS) attacks inherent in IPv6 design. Despite ongoing advancements in Artificial Intelligence/Machine Learning (AI/ML) driven research, such attacks persist, inflicting significant losses on organizations. In response, this study introduces two distinct architectures within a Deep Neural Network (DNN) model. Model 1 integrates Convolutional Neural Networks (CNN) with Long Short-Term Memory (LSTM), inspired by Ahmed Issa's work. Meanwhile, Model 2 proposes an integration of Recurrent Neural Networks (RNN) with Gated Recurrent Units (GRU). The models were evaluated following Ahmed Issa's architecture using NSL-KDD, Sain Malaysian and Mendeley datasets, resulting in accuracies of 80%, 97.01%, 95.06%, 72.89%, and 64.94%, respectively. Notably, NSL-KDD and Mendeley datasets are IPv4-based, whereas the Sain Malaysian data is IPv6-based. These results were compared with those obtained using the NSL-KDD benchmark datasets. These results demonstrated that such combinations are effective for detecting ICMP DDoS attacks.

Further experiments were performed on the proposed model's architecture, and it was deployed using the Sain Malaysian datasets (IPv6-based). As a result, both models exhibited promising performance, achieving accuracies of up to 83.95% and 83.83%, respectively. Further ML techniques were also deployed using the proposed model. Three combinations were derived using the stacking technique for comparison: (1) CNN with LSTM + RNN with GRU, (2) various ML techniques, and (3) a combination of both (1) and (2) treated as ALL. The optimistic results obtained were 84.14%, 86.16%, and 86.19%, respectively. Additionally, two sets of ICMPv6 datasets are generated in two distinct environments, which helps to prove our research model is robust. The experiments continued to evaluate the robustness of the proposed model using Feature engineering from the physical and data link layers of the network to, windowing, Time Series split, Cross validation, ADASYN, LIME, SHAP, and AAD, measuring the model performance by metrics like Recall, F1 measure, Precision, ROC and AUC achieving promising results focusing more on Accuracy results. The results ranged from 81.56% to 99.998%, and in some cases reached 100%. The AAD and the inferences indicated that the Proposed model at base classifiers are not suitable for real-time implementation but recommended for Ensemble Stacking in real-time deployments.

Further, an Ensemble stacking technique is deployed on the proposed Model 1 and Model 2 as base classifiers along with the ADASYN technique, achieving outstanding results of accuracies of 99.89% and 99.97%, respectively. A critical evaluation based on datasets, features, and state-of-the-art research results validates our proposed model as a promising solution with a superior score for the detection and prediction of ICMPv6 DDoS attacks, particularly for Echo reply and request packets.

Keywords: *DoS/ DDoS attacks, ML, AI, DNN (CNN, LSTM, RNN, GRU), Ensemble Stacking, ADASYN, ICMPv6, IPv4, IPv6 and Datasets.*

Contents

LIST OF FIGURES

List of Tables

Declaration

I declare that, except where specific reference is made to the work of others or explicitly mentioned, the contents of this thesis are original and have not been submitted, either in whole or in part, for consideration for any other degree or qualification at this or any other university. This thesis is the result of my own work and includes no collaborative work.

Om Vasu Prakash Salamkayala

Acknowledgments

I am grateful to my Supervisor Dr. Saeed Shiry Ghidary. It is difficult to express the inspiration and the knowledge sharing I gained from him from the moment he was assigned and accepted to act as my supervisor. His interactions, guidance, valuable suggestions and advice motivated me to sustain my motivation throughout my course. I am fortunate to have his patience, understanding and support over the past 3+ years. I could not imagine the completion of this thesis without his exceptional mentorship. I feel very lucky to have such a great collaboration with him. My gratitude extends to my Second supervisor Mr. Christopher Howard, who has also supported me in this journey by providing his valuable suggestions and advice. His strong technical support, providing the required infrastructure to accomplish generating Datasets, made a great turning point that helped me to prove my proposed research work. I am once again fortunate to have such exceptional mentorship.

Further, I extend my gratitude to the Head of the Department Dr. Russell, Professor. Peter Kevern and Dr. Jane Wellens, who have supported me when I was going through hardship due to my father's serious illness from a kidney Tumour. I am also grateful to my fellow mate, Mr. Joideep, who has provided me with great moral support and valuable guidance in times when I needed it. The Income department and Graduate School's patience and understanding are evident in considering my critical situation and extending time for payment of fees.

I am indebted to my fellow research colleagues within the PhD lab. I am thankful to senior colleagues, especially Mr. Bob Hobs, Dr. Mostafa Tajdini and Mr. Stephen Cahill, who provided their valuable guidance and knowledge sharing.

I dedicate this work to my parents and family. Their overwhelming moral and financial support, from admission to the end of the course, has made this journey successful.

List of Publications

Papers:

Paper 1: Salamkayala, O.V.P., Ghidary, S.S. and Howard, C., 2024, July. Review of IDS, ML and Deep Neural Network Techniques in DDoS Attacks. In CS & IT Conference Proceedings (Vol. 14, No. 14). CS & IT Conference Proceedings. https://csitcp.net/abstract/14/1414csit24.

Paper 2: Salamkayala, O.V.P., Ghidary, S.S. and Howard, C., 2024, July. Detection of ICMPv6 DDoS Attacks using Hybridization of RNN and GRU. In CS & IT Conference Proceedings (Vol. 14, No. 14). CS & IT Conference Proceedings. https://csitcp.org/abstract/14/1414csit23.

Paper 3: Om Salamkayala, Saeed S.G, Chris. H, Russell. C, Joideep. B, 2024. Detection of ICMPv6 DDoS attacks using Ensemble stacking of hybrid Model-1 (CNN-LSTM) and Model-2 (RNN-GRU). Virtually presented the paper at the ICMLC conference in Japan. Published in IEEE. https://ieeexplore.ieee.org/abstract/document/10935151.

Primary Datasets:

Dataset 1: Salamkayala. Om (2024) “DDoS-Datasets”, GitHub Data repository, https://github.com/omvasu/DDoS-Datasets.git. (Om Salamkayala 18 June 2024)

Dataset 2: Salamkayala. Om (2024), “ICMPv6 DDOS - Dataset”, Mendeley Data, V1, doi: 10.17632/g583tzgv5s.1 https://data.mendeley.com/datasets/g583tzgv5s/1.(Salamkayala 20 June 2024)

1 INTRODUCTION

In recent years, Artificial Intelligence, like Deep Neural Networks (DNNs), has emerged as a significant application in various domains, delivering promising results. Concurrently, cyber threats such as DDoS attacks happen to be deadly threats to stable and secure network infrastructures. These attacks overwhelm targeted systems with massive amounts of traffic, rendering them inoperable and causing substantial economic and operational damage to organizations. By leveraging the advanced techniques and learning capabilities of DNNs, this research aims to develop robust, scalable, and adaptive solutions to effectively counteract the dynamic characteristics of modern DDoS attacks. This chapter introduces the thesis by explaining its evolution into a comprehensive research project, outlining the research hypothesis, and highlighting the contributions as inputs to this field of study. It also presents the research topics supported by relevant statistical evidence and describes the selected methodology in a structured format.

1.1 Background

With the proliferation of cutting-edge technologies in computing domains like Cloud Computing and the Internet of Things, the incidence of DDoS attacks has surged significantly. This escalating frequency poses a substantial threat, rendering DDoS attacks among the most formidable challenges in the realm of cybersecurity (H. Aydın, Orman, and M. A. Aydın 2022). This technology also opens up extensive avenues for various network attacks, specifically targeting critical services and causing system malfunctions, whether in servers or enterprise networks. Such disruptions lead to business paralysis, manifesting as downtime and resulting in significant financial losses. A denial-of-service (DoS) attack inundates a server with traffic, rendering a website or resource inaccessible. For a DDoS attack, multiple computers or machines collaborate to flood a specific target with overwhelming traffic, exacerbating the impact of the attack (Gaurav, Gupta, and Panigrahi 2022). This happens when attackers meticulously investigate unprotected entry points, such as vulnerabilities in software or system configurations, and skilfully exploit them. ICMPv6-based DDoS attacks, akin to their ICMPv4 counterparts, capitalize on vulnerabilities inherent in IPv6. Detection of such attacks traditionally relies on signature-based Intrusion Detection Systems (IDS). Nevertheless, researchers have introduced an adaptive intrusion detection system, leveraging machine learning (ML), which surpasses the efficacy of traditional signature-based IDS (Alghuraibawi et al. 2021).

Despite the implementation of sophisticated firewalls and IDS, DDoS attacks persist, as evidenced by recent statistics. In 2018, GitHub suffered a massive 1.3 Tbps DDoS attack. However, Imperva, a cybersecurity company, reported an even larger attack the same year, involving 500 million packets targeting an unnamed client and lasting 13 days. These attacks not only jeopardize security but also result in significant financial losses. According to the 2019 Annual Cyber Security Report by Bulletproof, a single DDoS attack could cost a small business 120,000 USD, with even greater financial impacts on larger enterprises (Dahiya and Gupta 2021). In February 2020, the largest DDoS attack on record occurred, peaking at 2.3 Tbps. This attack targeted CLDAP (Connectionless Lightweight Directory Access Protocol) web servers, surpassing the previous record of 1.3 Tbps set by the GitHub attack, which delivered 126.9 million packets per second (Alghazzawi et al. 2021). Similarly, one year after in February 2021, the cryptocurrency exchange EXMO experienced a surge in traffic, reaching 30 gigabits per second, rendering it inaccessible for 2 hours (Mittal, K. Kumar, and Behal 2023a). In 2022, Microsoft Corporation disclosed that it had been subjected to a DDoS attack, during which the network experienced an unprecedented traffic volume peaking at 3.47 terabits per second (Neira, Kantarci, and Nogueira 2023). According to Cloudflare's Q2 2024 report, Domain Name System (DNS) based DDoS attacks have become the most prominent attack vector, with their share among all network-layer attacks continuing to grow. It is clear that the share of DNS-based DDoS attacks increases up to 33.9%. Despite this surge, and due to the overall increase in all types of DDoS attacks, L3/4 attacks still account for 30% of the total. ICMP amplification attacks occupy 2.3%, alongside other types of attacks such as ACK, UDP, RST, etc. floods. Figure 1 shows the statistical view of the attack vectors from Cloudflare Q2 report (Cloudflare 2024).

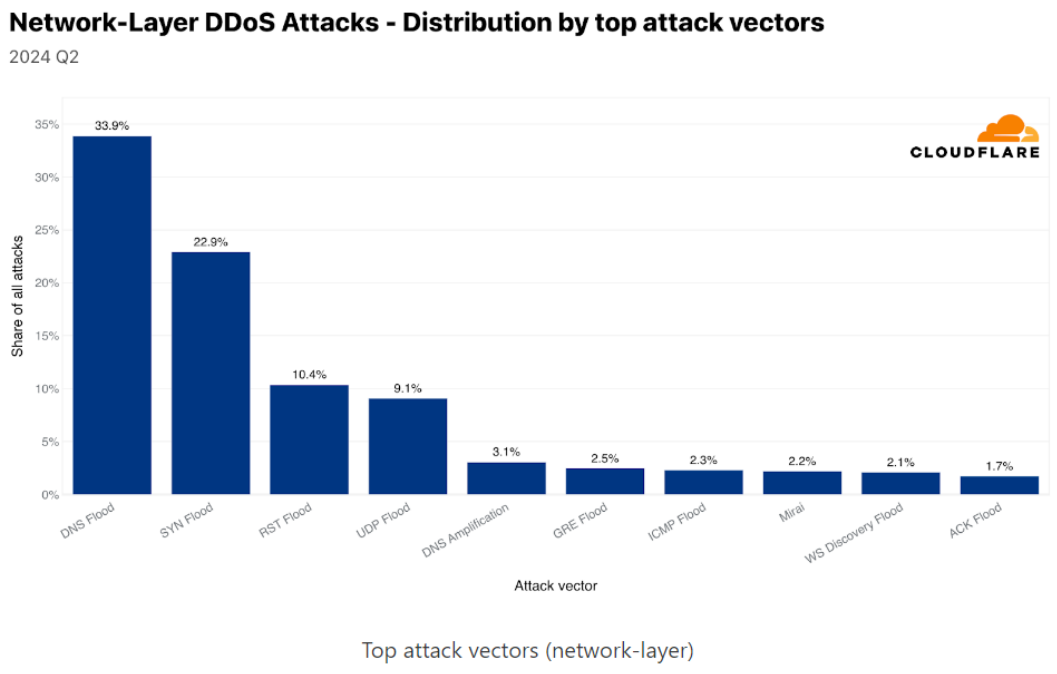


Figure 1: DDoS attack vector statistics (Cloudflare 2024)

1.1.1 Network Layers

Information transmission between systems involves multiple layers and protocols, transforming the data format during transit based on the OSI model and restoring it to its original form at the destination. The OSI model encompasses seven layers, and an overview of their functions can be summarised as follows.

1. The physical layer: Transmits information in a raw bit's stream format over a physical medium.
2. Data link layer: It specifies the format related to frames in the network. It is divided into 2 layers, the Media Access Control (MAC), which controls device interaction, and the Logical Link Control (LLC), which focuses on addressing and multiplexing.
3. Network layer: Determines the shortest path for packets to be routed in the network and concentrates on the IP protocol.

4. Transmission layer: This layer manages to ensures that the segments are in sequence with error-free. This includes port numbering and the use of TCP and UDP.
5. Session Layer: Manages sessions related to setup, authentication, connection, re-connection, and termination.
6. Presentation Layer: It is responsible for managing translating, encoding, decoding, encryption, and decryption information as required by an application.
7. Application Layer: This layer manages the information in user-readable format by accessing the services of the previous layer (Tanenbaum and J Wetherall 2010).

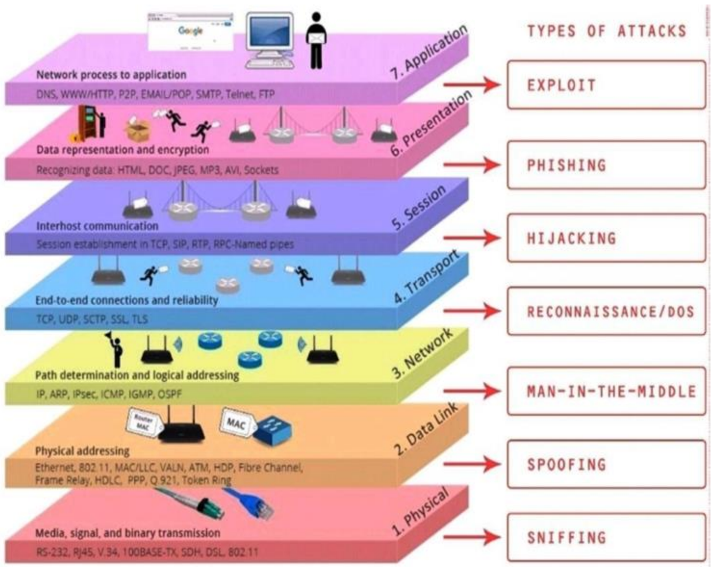

Google
7. Application
Network process to application
DNS, WWW/HTTP, P2P, EMAIL/POP, SMTP, Telnet, FTP
6. Presentation
Data representation and encryption
Recognizing data: HTML, DOC, JPEG, MP3, AVI, Sockets
5. Session
Interhost communication
Session establishment in TCP, SIP, RTP, RPC-Named pipes
4. Transport
End-to-end connections and reliability
TCP, UDP, SCTP, SSL, TLS
3. Network
Path determination and logical addressing
IP, ARP, IPsec, ICMP, IGMP, OSPF
2. Data Link
Router MAC
MAC
Physical addressing
Ethernet, 802.11, MAC/LLC, VALN, ATM, HDP, Fibre Channel, Frame Relay, HDLC, PPP, Q.921, Token Ring
1. Physical
Media, signal, and binary transmission
RS-232, RJ45, V.34, 100BASE-TX, SDH, DSL, 802.11
TYPES OF ATTACKS
EXPLOIT
PHISHING
HIJACKING
RECONNAISSANCE/DOS
MAN-IN-THE-MIDDLE
SPOOFING
SNIFFING

Figure 2: OSI 7 layers in Networking (Steingartner, Galinec, and Kozina 2021)

This research focuses on layer three, concentrating on the Internet Control Message Protocol version 6 (ICMPv6) concerning DDoS attacks using Deep neural networks to successfully detect and predict attacks. The scope of the feature selection is confined to the first three layers of information parameters that help to identify the ICMPv6 DDoS attacks that are based on Echo and Reply.

1.1.2 DDoS Classification of Attacks

Figure 3 illustrates the DDoS classification attacks. Some of the attacks used by threat attackers to launch ICMPv6 DDoS attacks are categorized under ICMPv6 DDoS attacks:

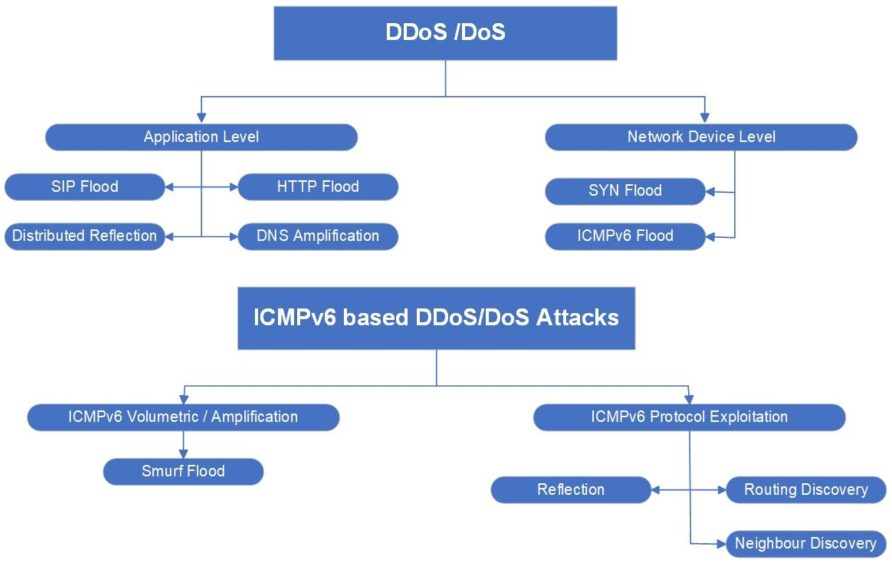


Figure 3: DDoS -ICMPv6 attack Classification (adapted from Bdair et al. 2020)

1. **Manual ICMPv6 Packet Generation:** Threat attackers can manually generate ICMPv6 packets using tools like hping or Scapy code. These tools enable attackers to create custom ICMPv6 packets with specific characteristics, such as source addresses, payload content, and packet size. By sending a high volume of these tailored packets to a target, threat attackers can overwhelm its resources and disrupt normal operations.
2. **Scripted ICMPv6 Packet Generation:** Threat attackers can develop scripts or leverage pre-existing tools to automate the generation and transmission of ICMPv6 packets. Automated attacks using scripts can achieve significantly higher volumes and sustained rates of packet transmission compared to manual methods.
3. **Botnet-based Attacks:** Threat attackers can exploit botnet networks of compromised computers under their control to launch ICMPv6 volumetric

attacks. By controlling a large number of compromised devices, threat attackers significantly expand the high volume of ICMPv6 traffic regulated at the target. These botnets are managed via command-and-control (C&C), enabling attackers to orchestrate coordinated and large-scale attacks.

4. **Reflective/Amplification Attacks:** In reflective or amplification attacks, Threat attackers spoof the source IP address of their ICMPv6 packets to make them believe as if they are genuine. They send these spoofed packets to misconfigured or vulnerable servers on the internet, which then respond with larger ICMPv6 packets. These amplified responses are directed back to the target, significantly increasing the volume of traffic, and maximizing the impact of the attack.
5. **Fragmentation Attacks:** Threat attackers can exploit fragmentation mechanisms in IPv6 to split ICMPv6 packets into smaller fragments. By sending a high volume of these fragmented ICMPv6 packets to the target, threat attackers can overwhelm the target's resources, as it must reassemble the fragments before processing the packets.
6. **Randomized Source Address Spoofing:** Threat attackers often resort to randomized source address spoofing techniques to thwart mitigation efforts aimed at filtering out malicious traffic. By continually altering the source addresses of their ICMPv6 packets, attackers can elude detection and bypass mitigation techniques that hinge on IP blacklisting or rate limiting.

(A. Alharbi and Alsubhi 2021).

Bdair emphasized the classification of DDoS attacks, broadly categorizing them into Application layer and Network layer attacks. SIP flood, HTTP flood, Distributed Reflection, and DNS amplification are some of the attacks present in the Application layer. On the other hand, Network layer attacks encompass SYN flood and ICMPv6 flood. Furthermore, ICMPv6 DDoS attacks are categorized into ICMPv6 Volumetric/Amplification attacks, which include Smurf Flood, and ICMPv6 Exploration, which encompasses Reflection, Routing Discovery, and Neighbour Discovery. (Bdair et al. 2020).

Besides hping and Scapy, there exist various tools that threat actors use to launch DDoS attacks. Some of the tools are listed in Table 1.

Table 1: Some of the tools used in DDoS attacks

Tool Name	Brief description
LetDown DDoS tool	It is effective for launching a TCP DDoS flood.
Hyenae tool	It is an effective tool to forge packet generators to launch ICMP, UDP, and TCP DDoS attacks
The Tribe Flood Network tool	It is an effective tool that contains client and daemon programs to deploy across the network. The nodes are usually located globally. This tool can launch ICMP, UDP, and SYN flood attacks besides Smurf and Blowfish to encrypt the list of IP addresses that are present in the process of attack.
Stacheldraht tool	It is very efficient in crashing the target devices using different transport and network layer packets. This tool uses Zombie architecture and handles encryption of clients while launching attacks related to TCP, UDP and ICMP DDoS attacks.
Trace6	It supports ICMPv6 echo request and TCP-SYN
Thc-ipv6	It is capable of multiple attacks that include DoS, DDoS, evasion of attack, etc.
Hping3	It is a tool based on Scapy to support IPv6 development and testing.
Scapy	Scapy is a program treated as a module in Python that gives users sophisticated options to develop scripts related to Networking

This research used a simple Scapy script to launch DDoS attacks while collecting or generating the datasets. The reason why this was selected is that Scapy is a well-equipped and powerful interactive packet manipulation program with numerous benefits for networking. Its ability to craft, manipulate, and analyse packets, combined with extensive protocol support and flexibility, makes it highly versatile. Scapy allows users to create custom packets, test network devices, and analyse network responses. It supports a wide range of protocols, making it suitable for various tasks, including security testing, penetration testing, and network diagnostics. Written in Python, Scapy is flexible and can be easily extended, integrated into automation scripts, and used for educational and research purposes. Its cross-platform compatibility and strong community support, coupled with extensive documentation, make Scapy an invaluable tool (Biondi 2008-2024).

1.1.3 DDoS Attack

A DoS attack inundates a server with traffic, rendering a website or resource inaccessible. In the case of a DDoS attack, multiple computers or machines collaborate to flood a specific target with overwhelming traffic, exacerbating the impact of the attack (Gaurav, Gupta, and Panigrahi 2022). Attackers meticulously investigate unprotected entry points, such as vulnerabilities in software or system configurations, and skilfully exploit them. Leveraging these entry points, they attempt to compromise the system by depleting its resources, thereby denying access to legitimate users. In alternative attack scenarios, malicious bots are deployed to inundate the target system with an overwhelming number of packets, ultimately leading to a server crash (A. Alharbi and Alsubhi 2021). Figure 4 outlines the steps involved in a Threat attacker's launching of a DDoS attack. The attacker begins by exploring methods like phishing to infiltrate a system and install malware. Once control of a compromised machine is secured, the attacker distributes bots to other systems through lateral movement. With control over these systems, the attacker deploys command and control, using scripts to command PowerShell to unleash a flood of ICMP packets at the target server. This flood overwhelms the server's resources, causing DDoS conditions, and preventing genuine users from establishing connections due to resource depletion. (Mateen and Shahzad 2021)

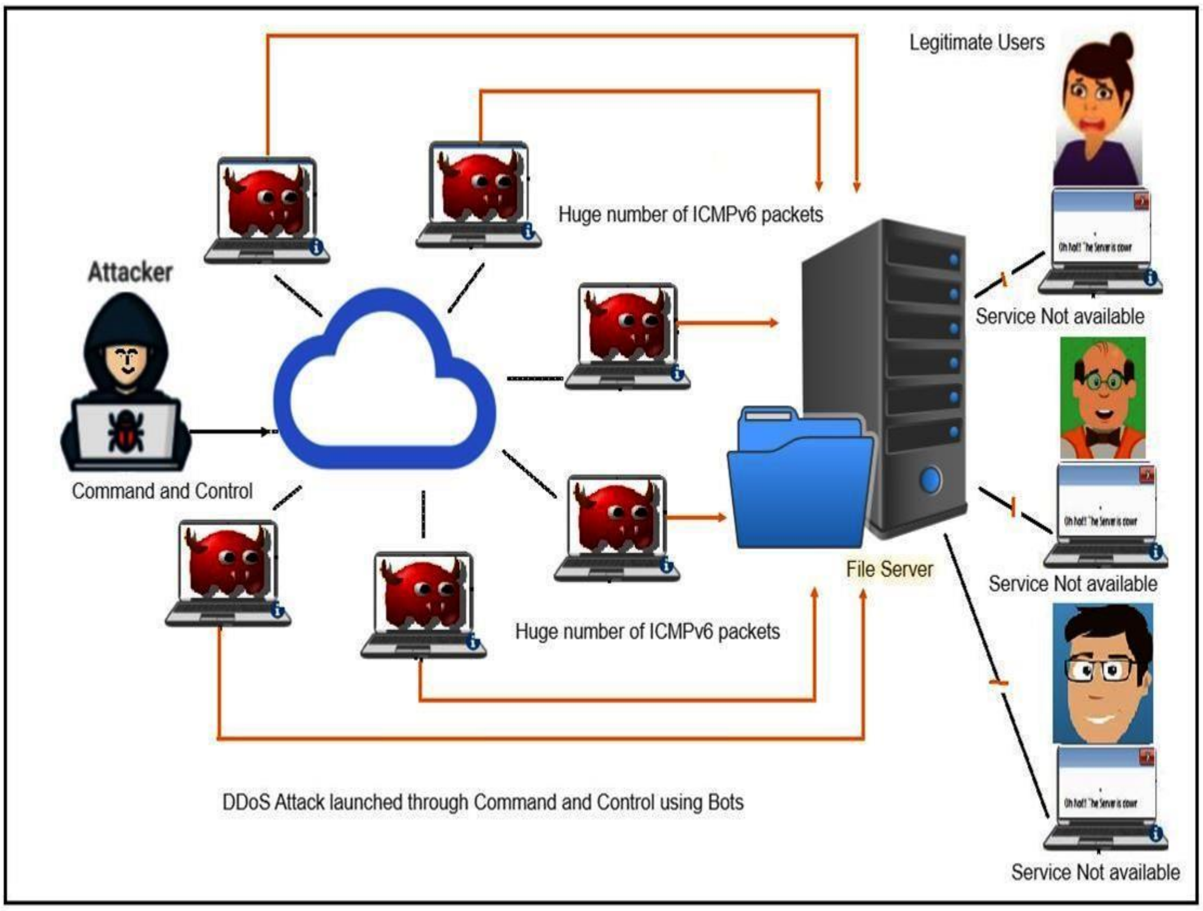

Legitimate Users
Huge number of ICMPv6 packets
Attacker
Service Not available
Command and Control
File Server
Service Not available
Huge number of ICMPv6 packets
DDoS Attack launched through Command and Control using Bots
Service Not available

Figure 4: DDoS attack Implementation of DDoS attack

Based on the incidents quoted and Figure 1, Cloudflare statistics 2024, DDoS attacks underscore their growing intensity, indicating a significant need for further research in this field. One specific area of concern is the use of the ICMPv6 protocol, which is vulnerable, and is usually used to diagnose the connection establishment or status of the destination system in the network. In this initial phase, threat attackers can launch attacks by exploiting the vulnerabilities in the network protocol, as they observe the mechanism of the traffic flow of the enterprise in and out to have a better understanding of the security settings. Addressing this gap is essential for enhancing network security and mitigating the impact of such sophisticated attacks.

1.2 Research Gap

Despite extensive research and various proposed solutions from previous researchers, DDoS attacks highlight their increasing intensity and frequency, presenting a significant opportunity for further research. Addressing this gap, particularly based on ICMPv6 protocol vulnerabilities, involves highlighting points considered to be good support for a more effective approach to enhance the possibility of detection of ICMPv6 DDoS attacks. These approaches, which are briefed in this section, are anticipated to achieve close to 100% accuracy.

The rapid proliferation of contemporary technology extends into various domains like IoT, autonomous vehicles, drones, and more, harnessing the internet and sophisticated intelligence scaling to support a myriad of smart devices equipped with Tiny Machine Learning at the hardware level, providing good security from threat attackers. However, despite these strides, significant gaps persist, contributing to diverse cyberattacks, and such vulnerabilities stem from the dynamic evolution of technologies or gaps in effectively implementing and securing them (Mishra and Pandya 2021). ICMPv6 falls within the Network layer, which is a connectionless protocol, and its essential purpose is for IP operations and network diagnostics. The stateless auto-configuration process relies on the Neighbour Discovery Protocol (NDP), which assists nodes in locating addresses and discovering other nodes. However, in the absence of IPsec security, ICMPv6 vulnerabilities emerge, creating opportunities for various attacks like DoS or DDoS defined at the beginning of this section. Malicious nodes can exploit these vulnerabilities to disrupt nodes in other network segments, crafting attacks to their advantage. By generating numerous ICMPv6 packets, attackers can significantly degrade network performance, especially by targeting nodes connected to the victim’s network segment (Mohmand et al. 2022). The ICMPv6 Ping of Death poses a significant challenge in combating DDoS attacks due to its capacity to overwhelm servers without depending on specific vulnerabilities. This type of attack exhausts the server’s processing capabilities by inundating a target with diverse ICMPv6 traffic, including ping requests, leading to a denial of service. Attackers exploit variations in IPv6 support and system configurations, such as differences in handling extension headers and time-to-live values, to evade detection. This scenario highlights the risk of malicious actors leveraging ICMPv6 to disruptive DDoS attacks, whether through sophisticated packet manipulation or straightforward methods like ICMPv6 information messages. The impact of such attacks is measured in Mbps /Tbps (bandwidth) and PPS (packets per second), underscoring their disruptive potential (Tajdini 2018). The threat actor employs covert intrusion techniques to operate discreetly. In the initial stages, they focus on crafting distinctive attack patterns devoid of identifiable signatures or exploiting vulnerabilities within network protocols and enterprise networks, eventually compromising the target server. For instance, they may manipulate out-of-order fragments, intentionally altering fragment values to deceive scanning systems, rendering the manipulated fragments seemingly genuine and thereby evading detection (Tan et al. 2022).

Despite the presence of various intelligent Intrusion Detection Systems (IDS), they often fall short in detecting such attacks due to the cunning approach of threat actors. When packets are scrambled with flag values lacking sequential order, IDS struggles to identify them, allowing attackers to evade detection more effectively (Tajdini 2018). Threat actors often leverage ICMPv6 packets in the initial stages to establish connections with systems or enterprises once they have the IP addresses. Subsequently, various vulnerabilities present a wide scope for launching attacks and some of the following are discussed very briefly:

- “ICMP messages can be manipulated to deceive the receiver into believing they originated from a different source than the actual originator.
- ICMP messages can be manipulated to redirect either the message or its reply to a destination other than intended by the message originator.
- ICMP messages are susceptible to alterations in message fields or payload.
- ICMP messages may be exploited in attempts to execute denial-of-service attacks by sending consecutive erroneous IP packets”.

(Deering and Conta 1998)

Given the identified gap, prudent to devise a contemporary strategy employing cutting-edge techniques, with the selection of AI Deep Neural Networks combination approach standing out as a promising solution. The subsequent chapter’s literature review underscores the significance of Neural Network methodologies explored by previous researchers, along with their inherent limitations. This research endeavours to transcend these limitations and make every effort to achieve a 100% accuracy rate in detecting and predicting ICMPv6 DDoS attacks.

1.3 Aim

This research aims to develop an advanced ICMPv6 DDoS attack detection method by leveraging feature engineering from the first two networking layers and interpreting feature contributions using SHAP, LIME, and permutation importance. To enhance detection accuracy, the study employs an ensemble stacking strategy that integrates CNN-LSTM and RNN-GRU deep learning architectures.

1.4 Hypothesis

Can these two combinations i.e. CNN with LSTM and RNN with GRU be successfully fused for the detection of ICMPv6 DDoS attacks with superior accuracy scores?

1.5 Related Questions

- Is there any similar kind of combination from existing researchers that supports the proposed model?
- Does Ensemble stacking enhance the accuracy metric in the context of a combination of DNN methods approach for predicting ICMPv6 DDoS attacks?

1.6 Objectives

1. Conduct a comprehensive literature review on ICMPv6 DDoS attacks, focusing on key methodologies relevant to the project. This includes deep learning models such as CNN-LSTM and RNN-GRU, ensemble techniques like stacking, and traditional machine learning approaches such as SVM. Additionally, the study of feature importance analysis using feature engineering, SHAP, LIME, windowing, Time series split, Train test split, ADASYN, etc. methods to enhance interpretability and model performance.
2. Designing a hybrid solution for the proposed model.

3. Use at least one benchmark dataset to deploy the proposed model and compare the results with other datasets.
4. Find specific ICMPv6 DDoS attack datasets from previous researchers and compare them with benchmark datasets (2&3).
5. Generating new ICMPv6 DDoS attack data sets using different network environments.
6. Validating the proposed techniques using performance measuring in terms of accuracy, precision, recall, and f-measure, comparing each technique based on the accuracy metric as well as their combination.
7. Evaluate the proposed method against the contemporary DDoS detection methods.

1.7 Contribution

1. **Development of a Novel Deep Learning-Based Detection Model:** Designed and implemented a hybrid deep neural network architecture that combines CNN-LSTM and RNN-GRU models using ensemble stacking, specifically tailored for detecting ICMPv6 DDoS attacks.
2. **Generation of Realistic ICMPv6 DDoS Datasets:** Created two original ICMPv6 DDoS datasets in distinct network environments, each capturing unique attack patterns, to support robust model training and evaluation.
3. **Feature Engineering and Interpretability Analysis:** Applied feature extraction from the first two layers of the network stack and utilized SHAP, LIME, and permutation importance to assess feature contributions at both global and local levels.
4. **Comprehensive Model Evaluation Against Benchmarks:** Evaluated the proposed model using both benchmark datasets (NSL-KDD, Mendeley, Sain Malaysian) and the newly generated primary datasets. The model achieved state-of-the-art performance, with accuracy scores reaching up to 99.97%.
5. **Scholarly Dissemination:** Published multiple research papers in peer-reviewed international

conferences, including a paper accepted and presented at an IEEE conference.

1.8 Research Methodology

This research method was proposed by Saunders, who explains that research methodology is always a challenge for a systematic approach to achieve the desired aim. Various methodologies exist, such as quantitative, qualitative, experimental, and applied research. Careful selection and adherence to a suitable technique are crucial for justifying the research design, following a scientific model, and ensuring coherent development. Research strategy is a method that defines the approach of research and the steps involved to follow. It provides strong beliefs, theories, and philosophical assumptions that create a shape to understand the research questions and select a procedure to use righteous methods. Research methodology is an integral part of a thesis, which helps to ensure the consistency between selected tools, techniques, models, and underlying theories (Goundar 2012).

1.8.1 Onion Method

In the quantitative phase, the research involves collecting and analysing existing techniques, and assessing their merits, demerits, and gaps concerning the current industry standards. This process includes conducting surveys to identify specific challenges that address and justify the research purpose. (Apuke 2017).

Figure 5 illustrates each layer with its unique guidance options, offering researchers the flexibility to make the best selection. This approach helps researchers progress toward achieving positive results while maintaining good consistency in their research design. The first four layers outline the guiding steps to transform research questions or hypotheses into a prospective project. The fifth layer involves project testing, which is essential for validating the hypotheses and obtaining robust evidence to support the project’s objectives. The final layer focuses on framing the thesis with supporting test results, demonstrating that the technique used is superior and effectively achieves the project’s aim (Alturki 2021)

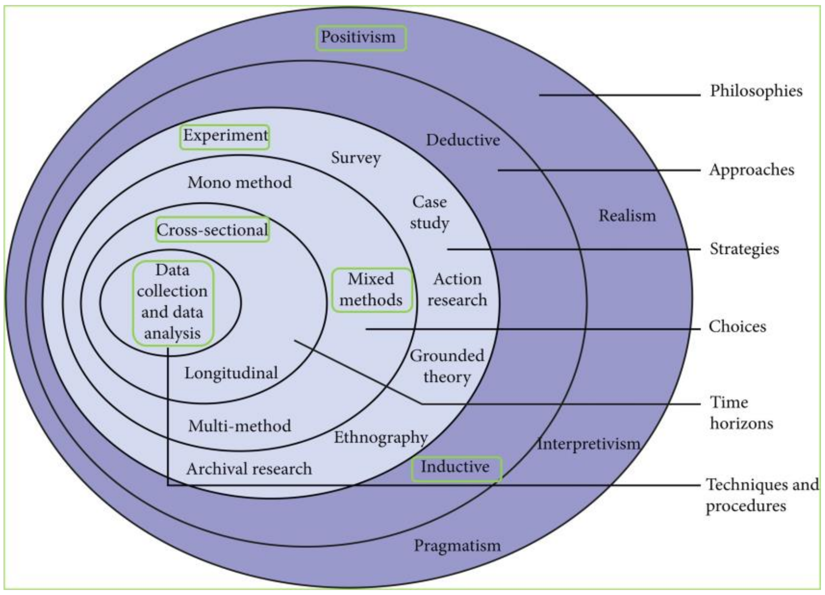


Figure 5: Onion research methodology with 6 layers (Alturki 2021)

1.8.2 Applying Onion Research Methodology.

This Onion research Methodology consists of 6 layers employed in the proposed research to the Hypothesis, Objectives, and contributions. The same is briefed in tabular form and is also given at the end of this section.

1. **Philosophy:** In this research, the positivist approach is adopted as the foundational layer of the Onion method. This approach is particularly suited for the study, which aims to scan packets and detect attacks using a Deep Neural Network (DNN) model designed to identify and mitigate DDoS attack packets, thereby safeguarding the network system. The following steps outline the nature of the idea, its development into a proposal, the scope of proving the proposal through simulation or a development framework, the strategic procedure, and the validation of the proposal (Mardiana 2020).
2. **Approach to theory development:** In the Onion method for designing Deep Neural Network (DNN) combinations, an inductive approach from the second layer was used. The literature review revealed DDoS attack detection at the Network layer using various DNN techniques. However, no research has combined RNN with GRU for monitoring and detecting ICMPv6 DDoS attacks at the enterprise Edge Router. This gap led to the development of a theory proposing DDoS detection using a DNN combination based on ICMPv6

echo/reply attributes. The objectives and hypotheses form a constructive base to identify packet features for detecting DDoS attacks at the Network layer (Melnikovas 2018).

3. **Methodological choice:** The Mono Method approach is utilized at the third layer of the Onion method, focusing on quantitative techniques. This approach involves conducting multiple experiments, tests, and assessments based on selected Deep Neural Network (DNN) techniques. The goal is to achieve outstanding performance measuring in terms of accuracy, precision, F1 measure, and recall (Apuke 2017).
4. **Strategy:** The Survey Experiment approach is used to develop solutions either by analysing collected datasets or by creating real-time virtual environments to gather datasets based on specific scenarios. Literature reviews suggest that for detecting DDoS attacks, packet header features for ICMPv6, in addition to essential frame features, should be collected at the Network layer. These features are then processed in the development of code designed to detect DDoS attacks on the first 3 layers in the OSI (Alturki 2021).
5. **Time Horizons:** This layer is about the cross-sectional or short-term of studying and focusing mainly on the collection, processing, analysing, etc. of data. Collecting data and understanding the insights of the data, like the context of attack, to data sets existing parameters or data extracted from the virtual platform of the DDoS attack scenario. Suitable parameters and frame features are extracted for analysis to check how they can fit into the technique. A systematic design and development of code is to be prepared to evaluate the tests and find the metrics (Melnikovas 2018).
6. **Techniques and Procedures:** The technical approach involves designing and developing a concrete solution using the Python programming language on the Google Colab platform. This entails utilizing specific repositories/ modules like Tensorflow, Sckit learn, Keras, Pandas numpy, etc. to implement a combination of Deep Neural Networks (DNNs), focusing on analysing features extracted from the ICMPv6 packet headers. The goal is to assess the efficiency of the developed code using accuracy, precision, F-measure, etc., to demonstrate the effectiveness of DDoS detection at the network layer. The

obtained results will serve to substantiate and defend the thesis, thereby achieving the objectives outlined in this proposal (Mardiana 2020).

1.8.3 Research Objectives Based on Onion Methodology

Table 2 provides an example of Research Methods for the Onion Methodology framework to ensure a thorough and robust investigation. It allows for precise experimentation, rigorous evaluation, and clear demonstration of the proposed model's effectiveness, providing compelling evidence of its superiority with impressive performance metrics.

Table 2: Research Objectives w.r.t Onion Methodology

Serial No.	Objective	Methodological Approach	Description
1	To investigate literature review on existing techniques, Gap and Solution analysis, related to ICMPv6 DDoS attacks detection (Systematic Literature Review (SLR), Rapid Review(RR)	Quantitative technique based on SLR.	SLR, RR streamline approach for producing evidence, typically for informing emergent decisions.
2	To study the ICMPv6 DDoS attacks, detection methodologies and DNN-related models.	Quantitative technique based on SLR.	SLR, RR. Identify gaps and solutions.
3	To investigate the application of DNN aiding in ICMPv6 DDoS detection.	Quantitative technique based on SLR.	SLR, RR. Identify gap-filling measures.

4	To propose and design a novel approach with a combination technique of DNN using the selected datasets.	Quantitative technique based on SLR.	Design the solution, plan and fill up the measures.
5	Develop test, implementation of a new framework based on the DNN model	Quantitative techniques, experiments, lab environments, virtualization of results in graphs and metric measures.	Laboratory experiments by simulation of datasets, benchmark datasets, verifying hypothesis theory, fine-tuning, feature selection, applying ADASYN, etc., for yielding high metric accuracy results.
6	Critical evaluation of the proposed model. Submission of the thesis with achieved results	SWOT(Strength, Weakness, Opportunities and Threats) analysis. Comparative Analysis. Statistical Analysis and Paper Publication	The proposed Novel approach of combination of DNN is proved by suppressing start of the art results

1.9 Implementation Method

This section brief about the implementation of the proposed research work in 6 phases:

- **Research and Analysis:** The research began with a systematic literature review of existing DNN techniques and their applications in addressing ICMPv6 DDoS attacks. This review identified the limitations, gaps, proposals, solutions, and achievements of various researchers. Additionally, the study encompassed a comprehensive analysis of the evolution from IPv4 to IPv6, including their relationships and associations with ICMPv6. The investigation also covered ML, AI, and DNN techniques, evaluating their merits and demerits.

- **Planning:** An effective research plan was devised, incorporating periodic schedules and milestones to ensure the set deadline in due course of the study.
- **Design and Configuration:** Effective model combinations were designed, utilizing CNN with LSTM and RNN with GRU employing ensemble techniques. Key datasets identified for this phase included NSL-KDD and Mendeley and Sain-Malaysian. During the research on ICMPv6, it was noted that ICMPv6 datasets were not openly available. This challenge was addressed by obtaining datasets from a secondary researcher who had conducted similar research.
- **Implementation and Deployment:** Additional ICMPv6 datasets were generated, and the model was developed using Python on the Google platform. Experiments were conducted to test the model.
- **Validation and Critical Evaluation:** High accuracy scores were targeted, and efforts were made to achieve these using ensemble stacking techniques. The model's performance was evaluated based on comparisons of scores across different datasets and against other researchers' achievements in similar research areas.
- **Conclusion and Thesis Preparation:** Upon successfully achieving the required results, a conclusion was drawn affirming the robustness of the proposed model in detecting ICMPv6 DDoS attacks. A thesis was prepared detailing the research process from inception to completion and was subsequently submitted.

1.10 Organization of Proposal Transforming into Thesis

The thesis is structured in the following Chapters format:

Chapter 1 provides the Introduction, Background, Research Gap, Aim, Hypothesis, Objectives, Contribution, and Research Methodology.

Chapter 2 provides a comprehensive literature review of research to provide a good understanding of the earlier researchers' work and their merits and limitations. A similar technique can help to support this research.

Chapter 3 provides the concept and design of the Model. Architecture, a Comprehensive review of DNNs used in the model, advantages, and applications.

Chapter 4 provides brief background information related to ICMPv6 Echo request / reply header parameters and a Comprehensive Review of Primary and Secondary Data sets.

Chapter 5 provides the evaluation of the results obtained and the state of the art that determines the best performance of the designed model based on metrics like accuracy, precision, F-measure, etc. It also provides discussions on the outstanding results of the model performance and the feature contributions that impact the model with different datasets that have different backgrounds.

Chapter 6 provides a summary of the entire project with the conclusion of successfully achieving the hypothesis mentioned and correlating to the first chapter i.e., aim, objectives, and contributions in the detection of DDoS using DNN, including the impact, limitations, and future research.

2 LITERATURE REVIEW

This chapter reviews previous research on solutions to DDoS attacks, ranging from traditional IDS methods to modern AI approaches. This review supports the current research, which aims to improve detection and prediction efficiency by incorporating a combination of Deep Neural Networks.

There are many attacks in the network targeting some important services and systems to break/crash resulting in the freezing of business thus causing great financial loss. The most targeted service is DoS, and DDoS is among them. There are many preventive techniques to mitigate DDoS, yet the attackers can succeed due to the change of approaches based on the vulnerabilities present in the victim's network, applications, protocols, and infrastructure. Most of the earlier researchers have come up with anomaly detection which is to find a pattern of a certain problem that is based on behaviour at the Network layer. Such patterns are often referred to as anomalies, outliers, discordant observations, etc. (Yang et al. 2022). Network management is maintained based on a rule-based system, for example, a Supervisory Control and Data Acquisition (SCADA) network is used for maintaining or troubleshooting (Saad, Anbar, and Manickam 2018). With the rapid advancements in emerging computing technologies, including Cloud computing and IoTs, DDoS attacks have a substantial surge in frequency. This escalating trend poses a significant and pervasive threat, establishing DDoS attacks as one of the most formidable challenges in the realm of cyber security. The widespread acceptance of cloud-based infrastructures and the interconnected nature of IoT devices contribute to the amplification of these attacks, underscoring the pressing need for robust cybersecurity measures to mitigate and counteract the escalating risks on the Internet (Mishra and Pandya 2021).

2.1 Related work

In the realm of DDoS Attacks using Malware, two primary methodologies exist, Signature-based detection and Anomaly-based detection. While Signature-based detection has historically been effective, it falters when confronting malicious scripts or bots due to their continual mutation. As these methods evolve, so do their signatures, rendering traditional detection methods ineffective against new variants. In contrast, Anomaly-based detection techniques, which operate on the premise that malicious behaviour deviates from normal traffic patterns, have gained prominence for their adaptability to detect emerging new variations in real-world scenarios due to which current IDS detections failed (Mishra and Pandya 2021b).

IPv6, known as Internet Protocol Version 6 is the latest version of the Internet protocol. The transition from the Internet Version 4 (IPv4) to IPv6 encountered new problems, and the most crucial one is vulnerabilities. Some of these vulnerabilities are Evasion attacks, DDoS, and Fragmentation attacks. As per RFC (Request for Comment) recommendations, there are potential attacks launched by threat attackers irrespective of any Operating System (OS). Due to different architecture and technology platforms between Operating Systems and their different behaviour, it can lead to evasions from IDS, Firewall evasion, OS fingerprinting, Network Mapping, DoS/DDoS attacks, Remote script execution, and command and control code execution attacks (Tajdini 2018).

Advanced Persistent Threat (APT) attacks represent a distinct form of network intrusion, leveraging coordinated human actions rather than automated scripts. APT attacks entail persistent monitoring and engagement with a target entity until specific objectives are met. In contrast, DDoS attacks seek to disrupt network functionality by overwhelming resources, often lacking further strategic goals. The Mirai botnet, a notable instance of a DDoS attack, incapacitated numerous websites like Twitter, Netflix, Reddit, and GitHub, for several hours in October 2016. Presently, Mirai variants are still present posing ongoing threats capable of inflicting substantial harm to networks (N. Wang et al. 2022).

2.2 Traditional IDS

In rule-based detection, the system monitors network traffic to identify potential vulnerabilities and detect abnormal events by comparing incoming traffic against a predefined set of rules that outline common attack patterns. This method involves continuous monitoring to spot deviations from normal traffic, raising alarms when such anomalies are detected. While effective for systematic attacks, it struggles with abrupt network behaviour changes and conditions beyond preset parameters. Their reliance on Boolean association rules for detecting irregularities can also slow down the process as the number of attributes increases, complicating rule management (Kaur, M. Kumar, and Bhandari 2017).

Bdair presented a concise examination of DDoS attacks, shedding light on the vulnerabilities of intrusion detection systems in the context of IPv6. Various detection mechanisms, such as anomaly, signature, and hybrid approaches, were outlined. An emphasis was placed on the growing interest in anomaly-based detection, utilizing rule-based methodologies. Additionally, he also pointed out the susceptibility of unsecured messages within the ICMPv6 protocol. Moreover, he hinted at proposing an Optimization Algorithm technique, aiming to enhance the intrusion detection system either through adoption or hybridization with a meta-heuristic algorithm, thus increasing its capability to detect DDoS attacks (Bdair et al. 2020).

Bahashwan provided an overview of IPv6 DDoS attack detection, emphasizing signature, anomaly, Rule, Entropy, Machine Learning and Deep learning-based mechanisms and techniques. It also brief about the approach to determine, distinguish, and reduce the attacks related to IPv6 and efficiently brief their merits and demerits (Bahashwan, Anbar, and Hanshi 2020). Figure 6 depicts the logical approach to capture the anomaly behaviour by an IDS without any ML or any AI employed.

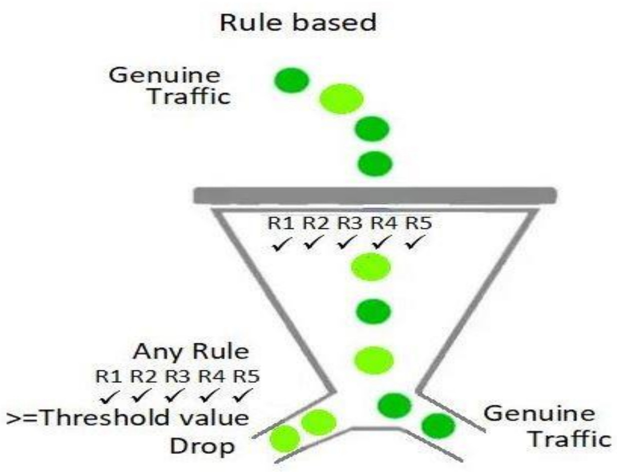


Figure 6: Logical approach of Rule base anomaly

- “**Rule 1:** One-way connection density in IPv6 networks which is inbound link utilization (bytes/s). The ICMPv6 packets that are sent without a corresponding response packet create a one-way connection (OC). ICMPv6 OC is the ratio of OC packets to all packets in a sampling time interval T. If this exceeds the predetermined value, then it implies, an abnormal behaviour is detected.
- **Rule 2:** Generally, in ICMPv6 flow, a packet set with the same five-component group (IPv6 source, IPv6 destination, source port, destination port and protocol), is used in the network analysis. The number of packets that belong to a certain ICMPv6 flow is called the length of the ICMPv6 flow. This rule is to detect the anomaly behaviour if the average length of ICMPv6 flow exceeds the given threshold.
- **Rule 3:** The ratio between inbound and outbound packets is usually steady. However, in an ICMPv6 anomalous behaviour attack, the ratio of this traffic increases rapidly. This rule is to determine as an attack if the anomaly behaviour that exceeds the given threshold value.
- **Rule 4:** The ratio of ICMPv6 echo request packet is the rate of set of ICMPv6 packet arrival to that of length of time interval (T1, T2, T3 ,..Tn). This rule is to detect the anomaly behaviour if the rate of ICMPv6 packet echo request

arrivals from a network to the set at the same length of time interval exceeds a threshold value Count.

- **Rule 5:** In this rule the count is used to determine the same number of sources IPv6 and the destination IP address. This rule is to detect the anomaly behaviour, if the number of packets has the same IPsrcont address source and IPdscont address destination exceeds the threshold value "

(Saad, Anbar, and Manickam 2018).

Rajat Tandon introduced AMON-SENSS, an open-source solution engineered to address the demands of scalable and precise DDoS detection, along with signature generation within expansive networks. AMON-SENSS adopts a hash-based binning strategy featuring multiple bin layers to ensure scalability, while simultaneously leveraging traffic analysis at various granularity. Additionally, it implements advanced techniques such as traffic volume and traffic asymmetry change-point detection to effectively pinpoint malicious activities. Consequently, their findings demonstrate AMON-SENSS's outperformance in accuracy, latency, and network signature quality compared to existing commercial alternatives (Tandon et al. 2022). Abnormal activities exceeding or deviating the threshold are concluded as an attack. The ICMPv6 has emerged based on the limitation of address space in the IPv4. However, IPv6 was developed with neighbour discovery protocol i.e. NDP which has vulnerabilities that can be exploited by attackers to launch an attack in the form of ICMPv6 which includes the lack of exchange of message authentication of NDP. Some of the attacks related to ICMPv6 are network reconnaissance attacks, routing headers, fragment headers and multi-casting. (Holkovic, Rysavy, and Dudek 2019).

The above discussion outlines the utilization of rule-based mechanisms for detecting and mitigating DDoS attacks, particularly focusing on ICMPv6 packets. However, as technology evolves, new attack vectors emerge, posing challenges at both hardware and application levels. As technology progresses, attackers become more sophisticated, adapting their methods to circumvent traditional defences. This evolution has spurred the adoption of automation, leading to the integration of machine learning (ML), artificial intelligence (AI), and deep neural networks (DNN) to enhance defence mechanisms against these evolving threats. Additionally, it seeks to identify novel approaches or methodologies that offer promising results while acknowledging their inherent limitations.

2.3 A Research Review on ML.

In the previous section, we discussed how IDS are designed using basic threshold and comparison mechanisms. These systems often fail to detect attack methods because they rely on predefined patterns and thresholds, making them ineffective against novel or sophisticated attacks that don't match existing signatures. Additionally, attackers can easily bypass these systems by slightly modifying their techniques to fall below the detection thresholds. This section provides a comprehensive overview of ML techniques employed by various researchers, exploring their efficacy in addressing DDoS attacks.

Ojugo conducted a study comparing machine learning methods for DDoS detection. They contrasted the Hidden Markov Model with an Experimental Hybrid (Memetic) Genetic Algorithm Trained Neural Network, which was based on a Rule-Generated and Fitness Function Model. Their evaluation utilized IDS datasets CIDDS-2017, comprising supervised network flow data for traffic based on the anomaly. They allocated 70% of the dataset for training and 30% for testing, achieving a fitness range between 0.8 and 0.865. Their results indicated an estimated 80% classification accuracy for detection (Ojugo and Eboka 2020). Liang Xiaoyu conducted a thorough realistic assessment of ML-based DDoS detection methods, with a primary focus on addressing the class imbalance problem. They underscored the importance of feature selection, advocating for a model-oriented approach. Their evaluation, employing datasets from CAIDA and DARPA, utilized the correlation coefficient across various algorithms including Decision Trees (DT), Support Vector Machines (SVM), Radial Basis Function SVM (RBF-SVM), Polynomial SVM (Poly-SVM), K-Nearest Neighbours (KNN), K-Means (KM), Naive Bayes (NB), Artificial Neural Networks (ANN), and D-Ward. Remarkably, their approach achieved a D-Ward score of 77.03%, outperforming other algorithms in the evaluation (Liang and Znati 2019). Yasser Alharbi developed an improved KNN algorithm, termed GR-AD-KNN, to enhance the detection of ICMPv6 DoS attacks. This algorithm utilizes the information gain rate to assign weights to different features, allowing them to have varying degrees of influence on the classification process. By integrating the concept of offset increment average distance, the measurement of the target point is refined, enhancing the algorithm's stability. This refinement specifically addresses the varying impact of lengthy and small distance samples to determine, leading to more reliable detection outcomes (Y. Alharbi et al. 2021). Zewdie has developed an evaluation framework utilizing machine learning techniques to detect DoS and DDoS intrusions. By applying algorithms such as K-Nearest Neighbours, Decision Trees, and Random Forests, she conducted experiments using the CIC-IDS2017 dataset. The results revealed impressive precision metrics, with accuracies ranging from 92.19% to 99.66% (Zewdie and Girma 2022). Manjula implemented three classifiers K-Nearest Neighbours (KNN), Random Forest, and Naive Bayes on datasets generated using Wireshark besides applying the LOIC attack tool. Among these, the Random Forest classifier achieved the highest accuracy of 96.75%, demonstrating the model's effectiveness in detecting ICMP, TCP, and UDP flood attacks (Manjula and Mangla 2023). Researchers with similar research work on DDoS attacks using ML are provided in Table 3. In summary, ML offers a more automated, scalable, and adaptable approach to DDoS detection compared to traditional IDS methods. They are particularly effective in managing the quick detection of network traffic attack anomalies. However, lack of performance ability, scalability, complexity and adaptability in large networks.

Table 3: ML summary review

SERIAL NO.	OBJECTIVE	ALGORITHM	DATASET	FS - APPROACH	CLASSIFICATION TYPE	IDS DOMAIN	PAPER REF.	ATTACK LAYER	LIMITATION
1	DDoS Attack(TCP-SYN & ICMP Flood) detection in SON-enabled ISP Netwo	KNN, XG Boos	CAIDA20	Based on the Time window monitoring and entropy calculation	Binary classification Normal & DDoS	Flow-based	N. N. Tuan, P. H. Hung, N. D.Nghia, N. Van Tho, T. Van Phan, and N. H. Thanh, A DDoS attack mitigation scheme in IP networks using machine learning based on SON," Electron., vol. 9, no. , pp. 1-19, 2020.	Transport layer	SDN-based networks and No Combination / Integration of algorithm Primary Datasets
2	The DDoS attack detection through machine learning and statistical methods in SDN	J48, Bayes Net, Random Tree, REP Tree, NB, LR.	UNB-ISCX, CTU 13, ISOT	Manual Selection based on neighboring nodes.	Binary Classification: Normal and DDoS.	Flow-based	A. Banttalebi Dehkordi, M. R. Soltanaghaei, and F. Z. Boroujeni, "The DDoS attacks detection through machine learning and statistical methods in SON," J. Supercomput., vol. 77, no.3, pp.2383-2415 ,2020	Network layer	SDN-based networks and No Combination / Integration of algorithm
3	Low-rate DDoS attack detection using ONOS controller and ML methods	J48, REP Tree,RF Random Tree, SVM, MLP	CIC-DDoS-2019	Manual selection	Binary Classification: Normal and DDoS	Flow-based	Perez-Diaz, J.A., Valdovinos, I.A., Choo, K.K.R. and Zhu, D., 2020. A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning. IEEE Access, 8, pp.155859-155872.	Application layer	Https-based attacks and No Combination / Integration of algorithm
4	SVM incorporated with selective IP traceback-based	SVM	NSL-KDD	Manual Selection	Binary Classification: Normal and attack	Flow-based	P. Hadem, 0. K. Saikia, and S.Moulik, "An SON-based Intrusion Detection System using	Application layer	Https-based attacks and No Combination /

	IDS mechanism for SON						SVM with Selective Logging for IP Traceback ," Computer. Networks, vol. 191, no. September 2020, p. 108015,2021		Integration of algorithm
5	DDoS attack detection using feature selection and ML-based techniques	SVM,ANN, KNN,NB	Self-Generated Simulated Data	Filter, Wrapper and Embedded based method	Multiclassification Normal, TCP, ICMP, and UDP	Flow-based	H. Polat and 0. Polat, "Detecting DDoS Attacks in Software Defined Networks Through Feature Selection Methods and Machine Learning Models," Sustainability, vol. 12, no. 3,1035,2020	Network Layer	SDN controller DDoS attack and No Combination / Integration of algorithm
6	DDoS attack detection using Hierarchical ML and Hyperparameter optimization	XGboost, LGBM, CatBoost, Random Forest (RF), and Decision Tree (DT)	CICIDS 2017	LASSO approach was used for feature selection	Decision Tree classification	Flow-based	Dasari, S. and Kalari, R., 2024. An effective classification of DDoS attacks in a distributed network by adopting hierarchical machine learning and hyperparameters optimization techniques. IEEE Access.	Network layer	No Combination / Integration of algorithm
7	Detecting DDoS Threats Using Supervised Machine Learning for Traffic Classification in Software Defined Networking	logistic regression, support vector machine, random forest, K-nearest neighbor, and XGBoost,	CICDDoS2019	---	Traffic class distribution based on Benign and attack	Flow-based	Hirsi, A., Audah, L., Salh, A., Alhartomi, M.A. and Ahmed, S., 2024. Detecting DDoS Threats Using Supervised Machine Learning for Traffic Classification in Software Defined Networking. IEEE Access.	Network layer	No Combination / Integration of algorithm
8	Online Network DoS/DDoS Detection: Sampling, Change Point Detection, and	SVM,DT,KNN.RFLDA,QDA and CPD	Multiple datasets NSL-KDD, CIC-IDS2017, and CSE-CICIDS2018	PCA	Classification based on attack instances normal instances	Multiple sampling like IP flow-based, Systematic, stratified, or	Owusu, E., Rahouti, M., Jagatheesaperumal, S.K., Xiong, K., Xin, Y., Lu, L. and Hsu, D.F., 2024. Online Network DoS/DDoS Detection:	Network layer	No Combination / Integration of algorithm

	Machine Learning Methods					random sampling…	Sampling, Change Point Detection, and Machine Learning Methods. IEEE Communications Surveys & Tutorials.		
9	Federated Learning Based DDoS Attacks Detection in Large-Scale Software-Defined Network	Federated Learning, XGBoost algorithm, gradient-boosted decision tree	InSDN, CICDDoS2019, and CICDoS2017	Manual Selection	DDoS attacks or normal	Flow-based	Fotse, Y.S.N., Tchendji, V.K. and Velempini, M., 2024. Federated learning-based DDoS attacks detection in large-scale software-defined network. IEEE Transactions on Computers.	Network layer	No Combination / Integration of algorithm
10	A Genetic Algorithm- and t-Test-Based System for DDoS Attack Detection in IoT Networks	Random Forest (RF), ExtraTree (ET), and Adaptive Boosting (AdaBoost)	ToN-IoT and HL-IoT binary datasets	PCC and novel ‘‘GAStats’’	Flood and low	time-series	Saiyed, M.F. and Al-Anbagi, I., 2024. A Genetic Algorithm-and-T-Test-Based System for DDoS Attack Detection in IoT Networks. IEEE Access, 12, pp.25623-25641.	Network layer	No Combination / Integration of algorithm

2.4 A Research Review on DNN

Kumar and his team conducted a comparative analysis of various Deep Learning techniques, including LSTM, Bidirectional LSTM, Stacked LSTM, and GRU. They structured the unstructured dataset CI-CDDoS2019, provided in CSV format, and pre-processing was performed by eliminating values such as NaN and infinity. Numerical values underwent standardization, while class values were encoded using label encoders. The pre-processed data was then fed into the model of Deep Learning techniques, allocating 80% for training and 20% for evaluation from the CSV file. Among these techniques, Stacked LSTM emerged as the most effective, achieving a remarkable accuracy of 99.55% compared to others (K. Kumar, Behal, et al. 2021). M. Asad and his team introduced a Deep Neural Network model employing a feed-forward back-propagation architecture, comprising seven layers to classify network flows and discern between attacks and normal traffic. The architecture includes three layers: input, hidden, and output. The input layer accommodates 66 features along with a bias factor, while the hidden layer initializes synaptic weights and connections to aid in classification computations. The output layer offers probabilities of benign traffic or a DDoS attack. They evaluated their model using the CIDC IDS 2017 dataset, achieving 98% accuracy (Asad et al. 2020).

Assis introduced a defence system focused on analysing records on a single IP flow, employing the GRU deep learning method to find DDoS and intrusion attacks. The model was assessed against various machine learning approaches using the CICDDoS 2019 and CICIDS 2018 datasets. An approach with a lightweight ability of mitigation was proposed and rated, with performance tests conducted on real network flow packets related to large-scale networks. The results obtained were outstanding in detection rates, achieving an accuracy of 97.1% (Assis et al. 2021).
Cil developed a DNN model with three hidden layers, each consisting of 50 neurons and utilizing sigmoid activation functions. This model was designed to detect DDoS attacks with the CICDDoS2019 dataset, achieving an impressive accuracy of 97.1%

(Cil, Yildiz, and Buldu 2021). Christian Callegari aimed at a deep learning-based method for network attack identification utilizing RNN, CNN, LSTM, and GRU. This approach was tested using datasets of traffic traces collected from the MAWI Lab archive, achieving an accuracy of 89.99% (Callegari, Giordano, and Pagano 2024). Researchers with similar research work on DDoS attacks using DNN are provided in Table 4.

Table 4:DNN Summary Review.

SERIAL NO.	PAPER TITLE	MODEL USED	NATURE OF LEARNING	DATASET	DETECTION OF ATTACKS	ATTACK LAYER	LIMITATION	CHARACTERISTICS	REF
1	A deep-learning model for detecting network attacks	RNN with autoencoder	Unsupervised	CICDDOS2019	SYN Flood, UDP, flood attacks, web DDoS attacks	Transport and application layer	No Combination / Integration of algorithm and no primary datasets used	Highest evaluation of metric -Recall, Fscore, Accuracy precision. Feature dimensionality reduction	Elsayed, M.S., Le-Khac, N.A., Dev, S. and Jurcut, A.D., 2020, August. Ddosnet: A deep-learning model for detecting network attacks. In 2020 IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM) (pp. 391-396). IEEE.
2	CNN- Based Network Intrusion Detection against Denial-of- Service Attacks.	CNN	Supervised	KDDCUP 99 and CSE-CIC-IDS 2018	DoS-Hulk, DoS Slow HTTP Test. DoS-Golden Eye, DDoS-LOIC-HTTP DDoS-HOIC. Neptune Attack. Smurf Attack	Network layer	No Combination / Integration of algorithm and no primary datasets used	Hyper-parameter tuning for designing an optimal model	Kim. J.. Kim, J„ Kim. H.. Shim. M. Choi, E. CNN-Based Network Intrusion Detection against Denial-of-Service Attacks. Electronics. 2020. 9(6). 916. https://doi. org/10.3390/electronics9060

3	An Effective Convolutional Neural Network based on SMOTE and Gaussian Mixture Model for Intrusion Detection in Imbalanced Dataset.	SGM-CNN (SGM-combination of Synthetic Minority Over-Sampling Technique (SMOTE) and under-sampling for clustering based on Gaussian Mixture Model (GMM))	Supervised	UNSW-NB15and CICIDS2017	DoS-Hulk. DoS Slow HTTP Test DoS-Golden Eye. DoS-Slowloris. DDoS-LOIC-HTTP DDoS-HOIC, Bot Net attacks. general DoS attacks (UDP.TCP)	Application, Transport, and Network layer	No Combination / Integration of algorithm and no primary datasets used	Ability to address class imbalance problem.	Zhang. H.. Huang. L.. Wu, C. a, Li. Z. An Effective Convolutional Neural Network based on SMOTE and Gaussian Mixture Model for Intrusion Detection in Imbalanced Dataset. Computer Networks. 2020, 177, 107315. https://d0i.0rg/l 0.1016/j.com net.2020.107315
4	Detection of DDOS Attack using Deep Learning Model in Cloud Storage Application	Feature Selection-Based Whale Optimization DNN	Supervised	CICIDS2017	DoS Slowloris. DoS Slow HTTP Test. DoS Hulk and DoS Golden	Network layer	No Combination / Integration of algorithm and no primary datasets used	Storing the non-attacked data in cloud to provide security and avoiding the entry of DDOS attacks	Agarwal, A., Khari, M., Singh, R. Detection of DDOS Attack using Deep Learning Model in Cloud Storage Application. Wireless Personal Communications. 2021, 1-21. https://doi.Org/10.1007/S1127 7-021-08271-z
5	A Multi-Classifier for DDoS Attacks Using Stacking Ensemble Deep Neural Network	CNN, LSTM, GRU and stacking ensemble	Supervised	CIC-DDoS2019	DDoS	Network layer	Combination / Integration of algorithm used but no primary datasets used	Ability to determine the various classifications of DDoS attacks and lack to identify similar categories.	M. I. Sayed, I. M. Sayem, S. Saha and A. Haque, "A Multi-Classifier for DDoS Attacks Using Stacking Ensemble Deep Neural Network," 2022 International Wireless Communications and Mobile Computing (IWCMC), Dubrovnik, Croatia, 2022, pp. 1125-1130
6	Detection and Characterization of DDoS Attacks Using Time-Based Features	GNB, DNN and SVM	Supervised	CICDDoS2019	DDoS, MSSQL, SSDP, SYN Flood, PORTMAP, DNS, LDAP, NETBIOS,	Network layer	No Combination / Integration of algorithm and no primary datasets used	Ability to identify and perform well on time-based features	J. Halladay et al., "Detection and Characterization of DDoS Attacks Using Time-Based Features," in IEEE Access, vol. 10, pp. 49794-49807, 2022,

					SNMP, TFTP, NTP, UDP Flood, or UDP-Lag				
7	An Efficient Hybrid DNN for DDoS Detection and Classification in Software-Defined IIoT Networks	(CNN-LSTM) & XGBoost	Supervised	CICDDoS2019	DDoS	Network layer	Combination / Integration of algorithm used but no primary datasets used	Ability to determine feature selection	A. Zainudin, L. A. C. Ahakonye, R. Akter, D. -S. Kim and J. -M. Lee, "An Efficient Hybrid-DNN for DDoS Detection and Classification in Software-Defined IIoT Networks," in IEEE Internet of Things Journal, vol. 10, no. 10, pp. 8491-8504, 15 May15, 2023
8	An Autoencoder-Based Approach for DDoS Attack Detection Using Semi-Supervised Learning	Autoencoder (AE) and Support Vector Machine (SVM)	supervised and semi-supervised	CICDDoS2019	DDoS	Network layer	No Combination / Integration of algorithm and no primary datasets used	Ability to perform well on unbalanced datasets	T. Fardusy, S. Afrin, I. J. Sraboni and U. K. Dey, "An Autoencoder-Based Approach for DDoS Attack Detection Using Semi-Supervised Learning," 2023 International Conference on Next-Generation Computing, IoT and Machine Learning (NCIM), Gazipur, Bangladesh, 2023, pp. 1-7
9	Lightweight Deep Learning Method based on Group Convolution: Detecting DDoS Attacks in IoT Environments	Autoencoders and CNN	supervised	CICIoT2023	DDoS	Network layer	No Combination / Integration of algorithm and no primary datasets used	The DGConv-IDS model has the ability to lower computational costs and provide better detection performance to improve security protection capabilities against DDoS attacks	S. Yan, H. Han, X. Dong and Z. Xu, "Lightweight Deep Learning Method based on Group Convolution: Detecting DDoS Attacks in IoT Environments," 2024 10th International Symposium on System Security, Safety, and Reliability (ISSSR), Xiamen, China, 2024,

10	Enhanced DDoS Attack Detection Using Advanced Deep Learning Techniques	LSTM, GRU,RNN,DCAE and CNN	supervised	CICDDoS2019	DDoS	Network layer	Combination/amalgamation of CNN - LSTM used and no primary datasets used	Ability to detect diverse DDoS attack patterns	C. Abdelkarim, M. Merouane and B. Lina, "Enhanced DDoS Attack Detection Using Advanced Deep Learning Techniques," 2024 International Conference on Advances in Electrical and Communication Technologies (ICAECOT), Setif, Algeria, 2024, pp. 1-4

This section summarizes about DNNs that are used mainly for their automated, adaptive and high-accuracy detection capabilities in the following areas:

1. **Complexity Handling:** Easily fit in for typical model complex, non-linear relationships due to their deep architecture and multiple layers.
2. **Performance and Accuracy:** They typically offer higher accuracy and better performance on large and complex datasets due to their ability to capture intricate patterns.
1. **Scalability:** It is devised to manage high-scale data effectively, yielding it for more appropriate real-time detection of DDoS attacks.
2. **Adaptability:** More easily retrained with new data to adapt to evolving attack patterns, offering better adaptability to changing threat landscapes.
3. **End-to-end Learning:** Support end-to-end learning, directly mapping raw input data to output predictions in a streamlined manner.
4. **Feature Engineering:** Automatically discover and capture attributes from raw data, reducing the need for manual intervention

(Mittal, K. Kumar, and Behal 2023).

Sections 2.2 to 2.4 of the literature review, including the ML-tables and DNN table, provide the mechanisms behind DDoS attacks and highlight various research efforts aimed at mitigating these attacks. These efforts have employed rule-based systems, ML, and DNN. The review delves into the limitations of these approaches and explores how researchers have innovatively adapted their methods over time to address these challenges.

2.5 A Research Review on ICMP

Based on our aim the research was narrowed down to DDoS attacks related to ICMP, TCP, and UDP attacks using ML and DNN by various researchers. Mohammad Tayyab reviewed DoS and DDoS attack detection in ICMPv6 using ML techniques, discussing single classifiers (e.g., SVM, KNN, Decision Trees, NB) and hybrid classifiers. They detailed how classifiers, trained on DARPA 1999 and generic datasets, achieved detection rates of 94.47% and 96.55%, respectively with performance, scalability, efficiency, benchmarks, imbalance, and evaluation metrics addressed. Additionally, Blockchain applicability for detecting ICMPv6 DDoS attacks was proposed as a new research direction (Tayyab, Belaton, and Anbar 2020). Ren-Hung presented a model that can learn unsupervised data using CNNs for the detection of traffic anomalies in the network at an early stage. This model automatically profiles traffic features from raw patterns, focusing on the first few packets in a flow to learn features and determine non-linear relationships, achieving partial end-to-end learning. Trained on raw data, it builds a classifier to differentiate benign traffic and detect anomalies accurately. Using the 277.1 GB Mirai-based DDoS dataset from Robert Gordon University, their evaluation with PyTorch and TensorFlow achieved nearly 100% accuracy, with less than 1% false alarms and false negatives, from just two packets and 80 bytes (Hwang et al. 2020).

Ahmed Issa introduced an innovative deep-learning classification approach by combining two widely used algorithms, CNN and LSTM. The model was designed with a 7-layer deep neural network consisting of a 1D CNN layer with kernel and stride parameters, followed by a MaxPooling 1D layer, using ReLU activation function, and SoftMax for the output layers. The model was assessed using the NSL-KDD dataset, which contains 40 features and includes various types of attacks. The model achieved an impressive accuracy rate of 99.20% (Issa and Albayrak 2023). Omar Elejla introduced an innovative method for identifying ICMPv6 flooding DDoS attacks in IPv6 networks. This approach leverages deep learning and incorporates an ensemble feature selection technique, utilizing chi-square and information gain ratio methods to identify crucial features for accurate attack detection. The model employs LSTM network trained on the selected features, resulting in impressive detection accuracy rates: 87.1% for RNN, 99.4% for LSTM, and 99.11% for a GRU (Elejla et al. 2019). Hasan provided good insight into ML and DL techniques focusing on ICMPv6 DDoS attacks and their usage to detect and mitigate. He also provided the differences between both and a review of the adaption of ML and DL techniques in AIDS for detecting IPv4 and IPv6 attacks, such as DoS and DDoS flooding attacks (Hasan Kabla et al. 2023). Researchers with similar work on ICMP DDoS attacks using ML and DNN are provided in Table 5. The literature review highlights the evolution from traditional IDS detection methods to the use of ML and AI, specifically deep neural networks (DNNs), in detecting DDoS attacks, noting their limitations and merits. It reveals that attackers are increasingly using sophisticated covert techniques to evade detection and successfully launch DDoS attacks across networks of all sizes. Despite many proposed techniques and approaches, existing methods face limitations that emphasize the need for advanced techniques like DNNs.

Selecting LSTM in the combination:

Sections 2.4 and 2.5 highlight that several researchers have utilized LSTM-based models for DDoS attack detection due to their effectiveness in handling sequential data. Kumar employed LSTM, Bidirectional LSTM, Stacked LSTM, and GRU models, achieving a remarkable accuracy of 99.55% in detecting DDoS attacks. Christian Callegari used a combination of RNN, CNN, LSTM, and GRU models for network attack identification, achieving an accuracy of 89.99%. Ahmed Issa integrated CNN and LSTM to detect various types of attacks, reaching an accuracy of 99.20%. Omar Elejla trained RNN, LSTM, and GRU models on selected features to detect ICMPv6 flooding DDoS attacks in IPv6 networks, with resulting accuracies of 87.1% for RNN, 99.4% for LSTM, and 99.11% for GRU. Similarly, Sayed applied CNN, LSTM, GRU, and a stacking ensemble approach for DDoS detection. A. Zainudin proposed a hybrid deep neural network using CNN-LSTM and XGBoost for DDoS detection and classification in Software-Defined Networks (SDNs). C. Abdelkarim employed a combination of LSTM, GRU, RNN, DCAE, and CNN to detect enhanced DDoS attacks.

Based on these studies, the LSTM technique is selected due to its superior performance across the following parameters:

Temporal patterns:

DDoS attacks typically involve bursts of traffic over time. They exhibit sequential behavior, such as rapid increases in packet rate or abnormal flow durations.

Table 5: ICMP summary review

SERIAL NO.	PAPER TITLE	MODEL USED	NATURE OF LEARNING	DATASET	DETECTION OF ATTACKS	ATTACK LAYER	LIMITATION	CHARACTERISTICS	REF
1.	LUCID: A practical, lightweight deep learning solution for DDoS attack detection	CNN	Supervised	ISCX2012, CIC2017 and CSECIC2018	DoS Slowloris, DDoS (TCP, ICMP)	Network layer	No Combination / Integration of algorithm	Reduction in execution and saving of processing power	Doriguzzi-Corin, R., Millar, S., Scott-Hayward, S., Martinez-del-Rincon, J. and Siracusa, D., 2020. LUCID: A practical, lightweight deep learning solution for DDoS attack detection. IEEE Transactions on Network and Service Management, 17(2), pp.876-889.
2.	A deep learning approach with Bayesian optimization and ensemble classifiers for detecting denial of service attacks	Ensemble models and AE-based deep learning classifiers	Unsupervised	Digiturk and Labris	SYN ack DDoS attacks, ICMP DDoS, HTTP Flooding	Network and application layer	No Combination / Integration of algorithm	Hyper Parampara uses Bayesian optimization to reduce and select optimal values for hyperparameters	Gormez, Y., Aydin, Z., Karademir, R. and Gungor, V.C., 2020. A deep learning approach with Bayesian optimization and ensemble classifiers for detecting denial of service attacks. International Journal of Communication Systems, 33(11), p.e4401.
3.	A comprehensive study of DDoS attack	GRU-BWFA	Supervised	SNMP-M IB dataset	TCP-SYN, UDP flood, ICMP-echo, HTTP flood, Slow	Application, Transport, and	No Combination / Integration of algorithm	To identify several attacks from the SNMP-MIB dataset and restore the	Gangula, R. Mohan. V.M. and Kumar. R., 2022. A comprehence study

	detecting algorithm using GRU-8WFA classifier				Loris. Slow Post a	Network Layer	and no primary dataset used	System in the shortest possible time.	of DOoS attack detecting algorithm using GRU- BWFA classifier Measurement: Sensors. 24. p.100570.
4.	Attack detection analysis in software-defined networks using various machine learning method	KNN, SVM, XGB ,ANN and Renyi joint entropy	Supervised	Generated in SDN environment	ICMP.TCP.UDP	Network and Transport layers	No Combination / Integration of algorithm	Efficacy and efficiency of ANOVA using the ML and ANN techniques	Wang. Y.. Wang. X, Ariffin, MM. Abotfathi, M , Alqhatani, A. and Almutairi, U 2023. Attack detection analysis in software-defined networks using various machine learning method. Computers and Electrical Engineering. 108. p 108655.
5.	Modified Flower Pollination Algorithm (MFPA) for ICMPv6-Based DOoS Attacks Anomaly Detection	MFPA	Supervised	Generated	ICMPv6	Network layer	No Combination / Integration of algorithms	To select the most relevant features from the ICMPv6 dataset to detect ICMPv6 DDoS attacks using MFPA	Alghuraibawi. AH8. Manickam. S.. Abdullah. R„ Alyasseri. Z.A.A, Jasim, H.M. and Sant N.S.. 2023. Modified Flower Pollination Algorithm for ICMPv6- Based DDoS Attacks Anomaly Detection. Procedia Computer Science. 220. pp.776-781.
6	An approach to on-stream DDoS blitz detection using machine	Naive Bayes. KNN and Random Forest	Supervised	Data set generated using Loic attacking tool	ICMP. TCP. or UDP	Network & Transport layer	No Combination / Integration of algorithm	To detect attacks related to any traffic protocols	Manjula. H.T and Mangia. N. 2023. An approach to on stream DDoS blitz detection using

	learning algorithms								machine learning algorithms. Materials Today: Proceedings. 80. pp 3492-3499
7	Zone-based stable and secure clustering technique for VANETs	K-means	Supervised	Data sets generated using network simulator. NS2.35	ICMP	Network layer	No Combination / Integration of algorithm	To monitor mode of RSU for the detection and mitigation of impersonation attacks in VANETs	Sharma. S. and Awasthi, S.K., 2024. Zone- based stable and secure clustering technique for VANETs. Simulation Modelling Practice and Theory. 130, p102863
8	A Real Time Deep Learning Based Approach for Detecting Network Attacks	MLP. RNN. CNN. LSTM and GRU	Supervised	MAW1 Data sets	HTTP, ICMP, TCP & UDP	Application, Network & Transport layer	No Combination / Integration of algorithm and No primary datasets used	To detect anomaly attacks using Deep learning techniques	Callegari. C„ Giordano. S. and Pagano. M . 2024 A Real Time Deep Learning based Approach for Detecting Network
9	Detection of ICMPV6 DDOS Attacks Using Ensemble Stacking of Hybrid Model-1 (CNN-LSTM) and Model-2 (RNN-GRU)	CNN with LSTM, RNN with GRU	Supervised	NSL-KDD, Sain Malaysian Mendeley, two distinct primary datasets.	ICMPv6	Network Layer	Combination / Integration of algorithm and primary datasets used	To detect ICMPv6 DDoS attacks	O. V. P. Salamkayala, S. S. Ghidary, C. Howard, R. Campion and J. Banerjee, "Detection of ICMPV6 DDOS Attacks Using Ensemble Stacking of Hybrid Model-1 (CNN-LSTM) and Model-2 (RNN-GRU)," 2024 International Conference on Machine Learning and Cybernetics (ICMLC), Miyazaki, Japan, 2024, pp. 58-64,

10	Detection and Mitigation of DDOS Attack in SDN Using Feature Based RF & MLP Approach	Genetic Algorithm and Chimp Optimization Algorithm	Supervised	SDN_DDoS_2020	TCP, UDP, and ICMP	Transport and Network layer	No Combination / Integration of algorithm and No primary datasets	Optimal feature selection methods to identify ICMP, TCP, and UDP attack traffic	S. K, S. M, A. M. R, G. N and S. Tamilselvi, "Detection and Mitigation of DDOS Attack in SDN Using Feature Based RF & MLP Approach," 2025 3rd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT), Bengaluru, India, 2025, pp. 457-461

Traditional models may fail to detect such time-based anomalies, but LSTM is explicitly designed to handle time series data.

Memory Capability:

LSTM networks are a type of Recurrent Neural Network (RNN) with gated memory units that can remember long-term dependencies.

- Detecting slow-evolving attacks.
- Capturing traffic trends leading up to the attack.

Anomaly Detection:

- In DDoS detection, it's crucial to identify subtle deviations in traffic over time.
- LSTMs can learn normal network behavior and detect when the sequence of events deviates significantly, suggesting an attack.

High-Dimensional Sequential Data:

- Network traffic can include many features: packet size, flow count, protocol type, etc.
- LSTMs can process multivariate time series data, allowing more holistic analysis than traditional static classifiers.

Performance:

- Early detection
- Reduced false positives
- Adaptability to new attack patterns

2.6 DDoS scope level

Threat attackers can launch DDoS attacks at various levels. This section briefly explains the different scopes of such attacks.

1. **Application Level:** In this level, the threat attacker aims to disrupt an application by exhausting its resources, such as the maximum number of processes or simultaneous connections it can handle. These kinds of DDoS attacks are mostly found in web applications, blocking user access through repeated invalid login attempts, etc (Velauthapillai 2014).
2. **Operating System level:** Operating system DDoS attacks are like application DDoS attacks. Example: SYN flooding attack using TCP, where the threat attacker floods TCP SYN packets to the victim without completing the 3-way TCP handshake, exhausting the target system's connection state memory (Tajdini 2018).
3. **Hardware level:** Attackers flood the hardware device with a high volume of traffic or requests, consuming its available resources such as CPU cycles, memory, or bandwidth. This can lead to device slowdown or complete unresponsiveness (Velauthapillai 2014).
4. **Network Layer Attacks:** This layer can be considered as a communication layer at a core level for packet transmission. ICMPv6 DDoS attacks primarily target the network layer. They can involve flooding a network with ICMPv6 packets, such as ICMPv6 Echo Request (ping) floods, ICMPv6 Router Advertisement floods, or ICMPv6 Neighbour Discovery floods. This results in overwhelming network bandwidth or consuming network resources, impacting the availability of network services (Droms 2014).

This research focuses on Network layer attacks, especially on ICMPv6 Echo-Reply attacks and the rest are out of scope.

2.7 IPv4 Protocol

The origins of IPv4 are traced back to the ARPANET, which was an experimental network funded by the U.S. Department of Defence’s Advanced Research Projects

Agency (ARPA). ARPANET was designed to explore packet-switching technology developed as part of the research and development efforts in computer networking that began in the late 1960s and early 1970s. As different networks began to emerge, there was a growing need for a standardized protocol that could enable communication between diverse systems. The development of a common protocol was necessary to ensure interoperability and the seamless exchange of data (Vint and Kahn 1974). As a result, IPv4 was born, and its improvisation was documented through a series of RFC documents. RFCs are a set of technical and organizational notes about the Internet, issued by the Internet Engineering Task Force (IETF). The initial proposal for the Internet Protocol was outlined in RFC 791, published in September 1981 by Jon Postel. This document defined the basic structure and functionality of IPv4, and Jon Postel was A key person in the development of IPv4, responsible for editing and publishing many of the early RFCs that defined the protocol (Internet Control Message Protocol 1981). IPv4 known as Internet Protocol version four, is widely used to identify devices on a network through an addressing system. The adoption of IPv4 facilitated the development of the World Wide Web, email, and other critical internet services.
Some of the key points of Ipv4 are:

1. **Address Format:** IPv4 is of a 32-bit address size, extending to a total of 2^{32} addresses (approximately 4.3 billion addresses).
2. **Address Representation:** Typically represented in dot-decimal format (e.g., 192.168.0.1).
3. **Header Size:** The header of an IPv4 packet is 20-60 bytes long.
4. **Address Exhaustion:** Due to the rapid expansion of the internet and the scaling of devices, IPv4 addresses are nearly drained.

(CISCO 2006).

2.8 IPv6

As the internet continued to grow and scale drastically, it was becoming difficult to cope with the 32-bit address space of IPv4 because of the shortage of IPs that are unable to meet the needs of an increasingly connected system in the digital world. This led to the development of IPv6 which was meant for larger address space. The transition to IPv6 was deployed officially in 2018, with both protocols coexisting and being used in parallel during the transition period.

Table 6: IPv6 address scheme

Address Type	Binary prefix	IPv6 Notation
Unspecified	00…0 (128 bits)	::/128
Loopback	00…1 (128 bits)	::1/128
Multicast	11111111	FF00::/8
Link-Local unicast	1111111010	FE80::/10
Global Unicast	(everything else)	--

IPv6 uses a 128-bit address scheme, which allows for approximately unique addresses. This vast address space is a significant improvement over IPv4 which was supporting about 4.3 billion addresses. IPv6 addresses are written in hexadecimal format and distinguished by colons (e.g.,2001:0db8:85a3:0000:0000:8a2e:0370:7334). For easier format, leading zeros can be skipped, and such sections of zeros can be replaced with a pair of colon (::) and (2001:0db8:85a3:::8a2e:0370:7334 or 001:0db8:85a3:0:0:8a2e:0370:7334).The IPv6 address format is in a hexadecimal number system, it starts from 0 to 9 as numbers and from 10 to 15 its notation is represented as A to F (CISCO 2006).

2.8.1 Identifying the IPv6 address based on Type and notation.

It is based on higher-order bits that are shown briefly with the necessary format and notations.
(Deering and Hinden 2006)

There are roughly 6 categories of IPv6 addresses that are briefed below:

1. **Unicast:** This address is mainly used for one interface. A packet that has a unicast address is transported to that specific interface which is recognised by that address. Further, it has 6 kinds that are used at different nodes at their

designated levels and sometimes with the combination of IPv4. IPv4 compatible IPv6 address (nearly deprecated). The remaining are listed below:

a. Global unicast address
b. Site-local unicast address
c. Link-local unicast address
d. IPv4 mapped with IPv6 address.
e. Special IPv6 Address

Interface identifiers in IPv6 unicast addresses are used on a link and are unique within a subnet prefix. It is advised that the same identifier is avoided while assigning to different nodes of a link, though they are unique on a wider scope. An interface identifier can be determined from its link-layer address and should be used on multiple interfaces of a single node provided in different subnets. For instance, a Global Unicast address can be applied to a local scope identifier, and a Link-Local address can be applied to a universal scope identifier
(Tajdini 2018).

2. **Multicast:** It is an IPv6 address type that is used for multiple identifiers for a group of interfaces in the network. In a multicast address, there exist 4 flags consisting of 0, R, P and T are used to restrict the possibility of the multicast group. The higher order flag is reserved and assigned to "0". The values are interpreted as
    - "R = 1 indicates a multicast address that embeds the address on the RP. Then P must be set to 1, and consequently T must be set to 1,
    - R = 0 indicates a multicast address that does not embed the address of the RP
    - P = 0 indicates a multicast address that is not assigned based on the network prefix
    - P = 1 indicates a multicast address that is assigned based on the network prefix.

- T = 0 indicates a permanently assigned
- T = 1 indicates a non-permanent

Further, multicast is based on the scope of the 4-bit value of groups that are listed as:

- reserved
- Interface-Local scope: Its scope is limited to a single interface on a node and is only useful for loopback transmission of multicast.
- Link-Local scope: Its scope covers the same topological region as the corresponding unicast scope.
- reserved
- Admin-Local scope: Its scope is the smallest scope that requires administrative configuration, as it is not automatically derived from physical connectivity or other non-multicast-related configurations.
- Site-Local scope: Its scope is limited to a single site.
- (unassigned)
- (unassigned)
- Organization-Local scope: Its scope is designed to span multiple sites within a single organization.
- (unassigned)
- A (unassigned)
- B (unassigned)
- C (unassigned)
- D (unassigned)
- E Global scope: It does not have any boundary.
- F reserved
- Unassigned scopes are available for administrators to define additional multicast regions".

(Deering and Hinden 2006)

Within the reserved multicast address range from FF00:: to FF0F:: the following addresses represent the scope of the nodes:

- “FF01::1—All Nodes within the node-local scope (its only for that host)
- FF02::1—All Nodes on the local link (link-local scope).
- FF01::2—All Routers within the node-local scope
- FF02::2—All Routers on the link-local scope
- FF05::2—All Routers in the site (site-local scope)
- FF02::1:FFXX:XXXX—Solicited-Node (where XX:XXXX represents the final 24 bits in the IPv6 address of a node)”

(Droms 2014).

3. **Anycast:** An address that typically represents a set of interfaces that belong to various nodes. Anycast addresses are not syntactically distinguishable from unicast addresses. An exception in the Anycast scope is that Multiple interfaces can have a single unicast address assigned to them when they are used for load sharing over multiple physical interfaces. The same is true when multiple physical interfaces are treated as a single interface at the Internet layer. Routers using unnumbered interfaces on point-to-point links are not assigned IPv6 addresses, because the interfaces do not function as a source or destination for IP datagrams (Deering and Hinden 2006).

4. **Global unicast address:** IPv6 Global Unicast Addresses are akin to IPv4 public addresses and serve as the IPv6 addresses used on the internet. They are unique and routable, essential for global connectivity and crucial in the realm of IoT. Assigned by IANA, these addresses encompass the entire range of available IPv6 devices. The prefix for IPv6 Global Unicast Addresses is 2000::/3, where the high-level 3 bits are fixed as 001, allowing addresses starting with hex digits 2 or 3 depending on the fourth-bit value (0010.. for 2000::/3 and 0011.. for 3000::/3) (Tajdini 2018).

5. **Site-local unicast address:** These addresses were originally intended for intra-site addressing without requiring a global prefix. However, new implementations must now treat this prefix as Global Unicast and should not

support its original special behaviour. Existing implementations and deployments are allowed to continue using this prefix as they have been (Deering and Hinden 2006).

6. **Link-Local IPv6 Unicast Addresses:** These addresses are intended for addressing within a single link and are used primarily for tasks like automatic address configuration and neighbour discovery, especially in the absence of routers. Routers are prohibited from forwarding any packets with Link-Local source or destination addresses to other links (Tajdini 2018).

2.8.2 A brief comparison of IPV4 vs IPv6 is shown in Table 6:

Table 7: IPv4 vs IPv6

Serial No	Parameters	Internet Protocol Four	Internet Protocol Six
1	Version	IPv4	IPv6
2	Address size	32 bits or 2^{32}	128 bits or 2^{128}
3	Notation	Integer IP address format. Four lots of three-digit numbers, separated by full stops. Ex: 192.168.0.1	Hexadecimal format. Eight lots of four-characters hexadecimal numbers, separated by colons. Ex: 2600:1600:5b3::4bd3
4	Loopback address	127.0.0.1	::1
5	Requires address translation	Network Address Translation (NAT).	No
6	Packet addressing	Unicast, Broadcast, Multicast.	Unicast, Anycast, Multicast.

7	Address configuration	Manual DHCP Configuration	Default Autoconfiguration is stateless. It also supports stateful DHCPv6 Configuration
8	Header size	Variable: 20 bytes and can be increased up to 60 bytes when optional fields and flags are added.	Fixed 40 bytes: The size of separate extension headers varies.
9	Header Checksum	Yes	No
10	Optional Extras	Limited support for options controls	Numerous extension headers are available to enhance routing, fragmenting, quality of service, etc.
11	Privacy	IP address masking to hide the last eight bits of an address	IP privacy extensions are used for temporary addresses
12	Fragmentation	Controlled by routers	Controlled by the originator
13	Routing efficiency	Controlled in headers	Controlled in routing tables
14	Mobile support	Manual assignment	Default

2.9 IPv6 Header

Figure 7 provides a pictorial view of the IPv6 head structure comprises 8 fundamental fields essential for routing and delivering packets across networks. These fields include the version, source and destination addresses, traffic class, flow label, payload length, hop limit, and next header. Understanding the IPv6 field format is crucial for managing and troubleshooting modern IP networks.

1. **Version:** It provides the Protocol version which is always IPv6 and its size is 4 bits with the 0110 binary sequence.

2. **The Traffic Class:** This field in IPv6 is 8 bits and indicates the priority of the packet, aiding routers in traffic management. During congestion, routers discard packets that have values indicating the lowest priority. Currently, only 4 bits are in use and corresponding values are from 0 to 7 meant for controlling congestion traffic and likewise for values from 8 to 15 for uncontrolled traffic like video or audio streaming. Initially, the source node does the setting of priorities; however, in transmission, the routers can change them (Tajdini 2018).
3. **Flow Label:** This field is of size 20 bits and used by the source to label packets for handling special instances like non-default quality of service or real-time service by intermediate IPv6 routers. Routers differentiate flows using the source address, destination address, and flow label. Multiple flows can exist between a source and a destination due to concurrent processes. For routers or hosts that do not support flow label functionality, this field is set to 0. The source must also specify the flow's lifetime when setting the flow label (CISCO 2006).

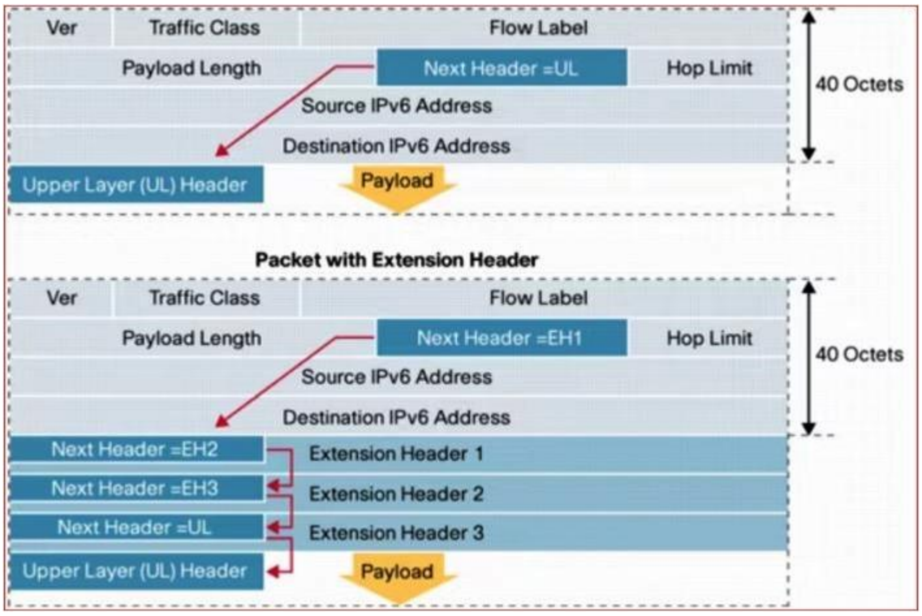


Figure 7: IPv6 header with optional Extension headers chain(CISCO 2006)

4. **Payload Length:** This 16-bit field is an unsigned integer informing routers about the packet's payload size that includes upper-layer packet and extension headers. The field is set to zero in case the payload exceeds the

size of 65,535 bytes, and the Hop-by-Hop options extension header in case of the jumbo payload (Gont and LIU 2022).

5. **Next Header:** This 8-bit field is for the type and indication of the presence of a forward extension header after the IPv6 header. In some cases, TCP or UDP is specified as the protocol within upper-layer packets. (Tajdini 2018).
6. **Hop Limit:** It is like Time To Live (TTL) in the IPv4 version, determining the number of intermediate nodes an IPv6 version packet can travel. Each forwarding node decrements its value by one, and the packet is discarded if it reaches 0. This mechanism prevents packets from getting stuck in an infinite loop due to routing errors (CISCO 2006).
7. **Source address:** This field indicates the source of the packet that contains a 128-bit IPv6 address (Gont and LIU 2022).
8. **Destination address:** This field indicates the original destination of the packet that contains a 128-bit IPv6 address. All the nodes carried by this packet go through this field to route the packet to the destination (Tajdini 2018).
9. **Extension Headers:** This field acts as a chain of next extension headers associated with the packet, overcoming the limitations of the IPv4 option field. It plays a significant role in the IPv6 architecture (CISCO 2006).

A sample packet that contains one or more extension headers should maintain the recommended order and sometimes it can also contain zero headers which is given below.

Table 8:IPv6 Header

1	Basic IPv6 Header	-
2	Hop by Hop options	0
3	Destination options with routing information	60
4	Routing header	43
5	Fragmentation header	44
6	Authentication header	51
7	Encapsulation security payload header	50
8	Destination options	60
9	Mobility header	135
	No next header	59

Upper Layer TCP		6
Upper Layer TCP	UDP	17
Upper Layer TCP	ICMPv6	58

2.9.1 Internet Control Message Protocol Version 6 (ICMPv6)

ICMPv6 serves a crucial role in network testing and diagnostics, enabling functions like ping and traceroute to verify end-to-end connectivity and identify node errors within the network. While ICMP can be disabled in IPv4 for security reasons, doing so in IPv6 can severely impact network diagnostics and packet transmission consistency (Deering and Conta 1998). Therefore, disabling ICMPv6 could lead to significant network issues and disrupt the reliable flow of packet data. Threat attackers take this service or utility as an advantage and exploit it to launch a deadly attack by deploying command and control or by a simple script (Salih 2017). Error messages and Information messages are the two categories present in ICMPv6 messages with their Type codes.

1. **Error Message:**
    a. Destination unreachable -1
    b. Packet too big-2
    c. Time out-3
    d. Parameter problem-4
    e. Private experimentation-100
    f. Private experimentation-101
    g. Reserved for expansion of ICMPv6 message-127
2. **Information message:**
    a. echo request-128
    b. echo reply-129

(Davies and Mohacsi 2007).

As per RFC4890, ICMPv6 messages function differently on Firewalls and Routers, and some of them are briefed below:

a. IPv4 consists of an address resolution protocol that converts a MAC address to an IP address and a reverse resolution protocol that is quite the opposite, i.e converts an IP address to a MAC address. This is not the same in IPv6, and for such a similar kind of functionality, we have i.e. Neighbour Discovery Protocol (NDP), Neighbour Advertisement (NA), and Neighbour Solicitation (NS).
b. To determine node information, network prefix, default gateway, etc. in the LAN Router Advertisement(RA) and Router Solicitation (RS).
c. Ping6 utility provides an Echo request and Echo reply.
d. Path Maximum Transmission Unit Discovery (PMTUD) provides the information related to MTU size for communication.
e. IP multicast presence and absence can be determined through Multicast Listener Discovery (MLD)
f. To find multicasting in routers, Multicast Router Discovery (MRD) is used.
g. Information about nodes among them is obtained using Node Information Query (NIQ).
h. Secure Neighbour Discovery (SEND) provides secure communication between neighbour devices like routers.

(Davies and Mohacsi 2007), (Deering and Conta 1998).

2.9.2 ICMPv6 vulnerabilities

Internet Control Message Protocol version 6 (ICMPv6) is an essential part of the IPv6 protocol suite, responsible for error messages and operational queries. Despite its critical role in network operations, ICMPv6 is susceptible to various vulnerabilities that can be exploited by attackers and some of them are provided below:

The Internet Assigned Numbers Authority (IANA) has established ICMPv6 parameters, categorizing ICMPv6 type numbers into two groups: error messages (0-127) and informational messages (128-255) as provided in the 2.9.1 section. Each type number is further defined with a sub-section of type code fields, as defined in RFC 4443, and the parameters can be useful references for interpreting ICMPv6 anomalies (Davies and Mohácsi 2007). RFC 4443 reinforces the use of multicast addressing (MA) and Multicast Listener Discovery (MLD), which are employed by IPv6 devices like routers and switches to track IPv6 multicast listening nodes on the link-local network. Similarly, Protocol Independent Multicast (PIM) is used by switches to send multicast packets. Although IPv6 multicast messages are limited to the local network prefix with a hop limit of 1, vulnerabilities exist where an attacker connected to the local network could send spoofed packets to multicast groups, potentially exploiting these groups to launch further attacks (Droms 2014), (M. Gupta and Conta 2006). Additionally, ICMPv6 error responses, such as "destination unreachable," are not limited to the local network, posing a challenge for filtering malicious ICMPv6 attacks (Li et al. 2022).

Duplicate Address Detection(DAD) is used in IPv6 networks when a node is first allocated an IPv6 address. The node sends an RS message to the all-routers multicast group (ff02::2) to obtain an IPv6 prefix, then an NS message to the all-nodes multicast group (ff02::1) to check this address is in use. A threat attacker exploits DAD by sending spoofed NA responses to NS requests, falsely indicating the address is in use, preventing any IPv6 addresses from being allocated on the network (CISCO-2020). Router discovery is a neighbour discovery process where a node sends a RS message to all router's multicast groups to obtain a RA response, receiving the connected network prefix and other autoconfiguration options necessary for network access. A Threat attacker can exploit this by deploying a malicious device to respond with spoofing forcing nodes to communicate through a malicious route and potentially launch DDoS and man-in-the-middle attacks (CISCO- 2012).

Threat attackers can exploit extension headers to craft new attack methods, using ICMPv6 to generate response or error messages that mask these attacks successfully evading from firewalls. ICMPv6 includes a parameter of type 137 and code 0 to advertise a new IPv6 router address for routing redundancy if a router is inaccessible. An attacker could spoof this redirect message, providing a fake gateway address for nodes on the link-local network, enabling a man-in-the-middle replay attack to capture packets and intercept sensitive information (Li et al. 2022). Bahashwan highlights that DoS attacks are the most common cyber-attacks targeting

ICMPv6. These attacks often use specific ICMPv6 message numbers and codes, with spoofed source addresses to hide the attacker's identity. One common method is ICMP flooding, where hosts receive numerous echo requests (type 128, code 0), forcing them to respond until network resources are exhausted, degrading performance. Amplification can be achieved by sending ICMPv6 echo requests to a multicast group address with the target's address spoofed as the source. Additionally, IPv6 routing loop vulnerabilities can allow DDoS attacks, detectable using the 'Time out' ICMPv6 message (Bahashwan, Anbar, and Hanshi 2020).

In this chapter, sections 2.6 to 2.9.2 of the literature review offer comprehensive insights into IPv6 and ICMPv6, covering their purposes, evolution, development, and features that facilitate broader usage. However, alongside their benefits, these technologies also exhibit vulnerabilities that can be exploited by attackers. To mitigate these risks and maintain a secure environment, a proposed model is presented in the next chapter to minimize the attacks especially related to ICMPv6 DDoS Echo-Reply attacks.

3 DESIGN OF ARTIFACT

This chapter addresses the design of the artifact aimed at bridging the gap identified in Chapter 1, Section 1.2. Initially, it delves into the merits of Deep Neural Networks (DNNs), offering a comprehensive understanding of how these networks contribute to the proposed solution. Following this, the chapter elaborates on the model architecture, and it concludes with a logical algorithm that is used to develop and apply to the secondary datasets and primary datasets for expected results.

3.1 Attack Scenario

Assume an attacker from the internet attempts to bypass the firewall to establish a connection. The attacker compromises systems in the E43 block of Figure 8(a)-ASN and initiates a DDoS attack by deploying Command and Control (C2) servers to flood the network with volumetric ICMPv6 packets. This flooding exhausts the server's resources, causing it to crash and malfunction in E42-DMZ. To counter this, a Solution Model is proposed as depicted in Figure 8(b), which employs a Deep neural network. This model integrates with CNN & LSTM as CNN and RNN &GRU as RNN trained to identify and detect malicious packets of a DDoS attack. Genuine packets are allowed to enter the enterprise network, while malicious packets identified by the Stacking-Regression learning mechanism are dropped and further predict such attacks in advance. This proposed model can be deployed in Medium to Large-scale Enterprise systems.

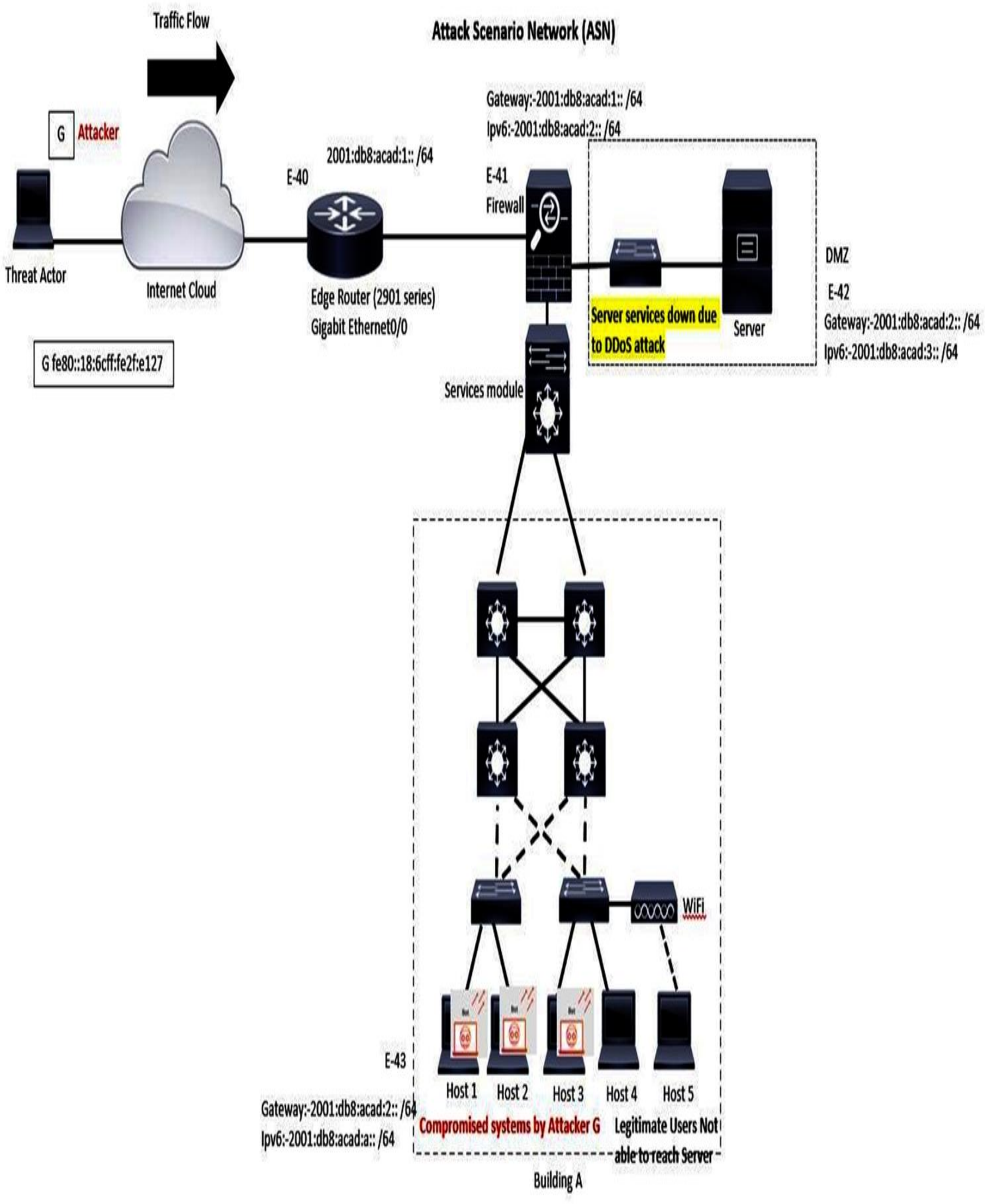

Traffic Flow
Attack Scenario Network (ASN)
G Attacker
Threat Actor
Internet Cloud
G fe80::18:6cff:fe2f:e127
E-40
2001:db8:acad:1:: /64
Edge Router (2901 series)
Gigabit Ethernet0/0
Gateway:-2001:db8:acad:1:: /64
Ipv6:-2001:db8:acad:2:: /64
E-41
Firewall
Server services down due to DDoS attack
Server
DMZ
E-42
Gateway:-2001:db8:acad:2:: /64
Ipv6:-2001:db8:acad:3:: /64
Services module
WiFi
E-43
Gateway:-2001:db8:acad:2:: /64
Ipv6:-2001:db8:acad:a:: /64
Host 1
Host 2
Host 3
Host 4
Host 5
Compromised systems by Attacker G
Legitimate Users Not able to reach Server
Building A

Figure 8 (a): Attack Scenario Network

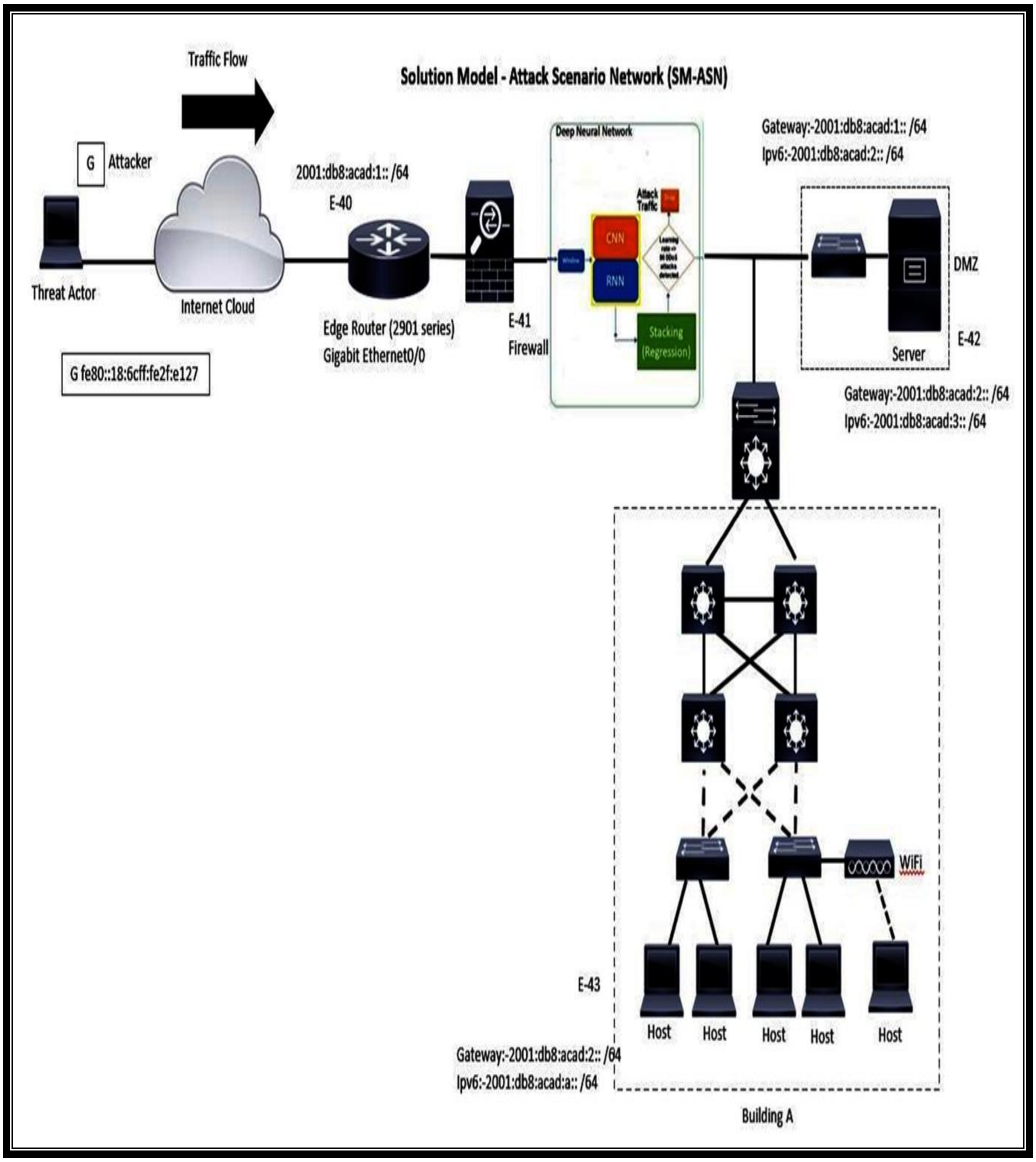

Traffic Flow
Solution Model - Attack Scenario Network (SM-ASN)
G Attacker
Threat Actor
Internet Cloud
G fe80::18:6cff:fe2f:e127
2001:db8:acad:1:: /64
E-40
Edge Router (2901 series)
Gigabit Ethernet0/0
E-41
Firewall
Deep Neural Network
Window
CNN
RNN
Attack Traffic
Learning rate => 80 DDoS attacks detected
Stacking (Regression)
Gateway:-2001:db8:acad:1:: /64
Ipv6:-2001:db8:acad:2:: /64
DMZ
E-42
Server
Gateway:-2001:db8:acad:2:: /64
Ipv6:-2001:db8:acad:3:: /64
WiFi
E-43
Host
Host
Host
Host
Host
Gateway:-2001:db8:acad:2:: /64
Ipv6:-2001:db8:acad:a:: /64
Building A

Figure 8 (b) Solution Model for ASN

3.2 Logical flow of a proposed model

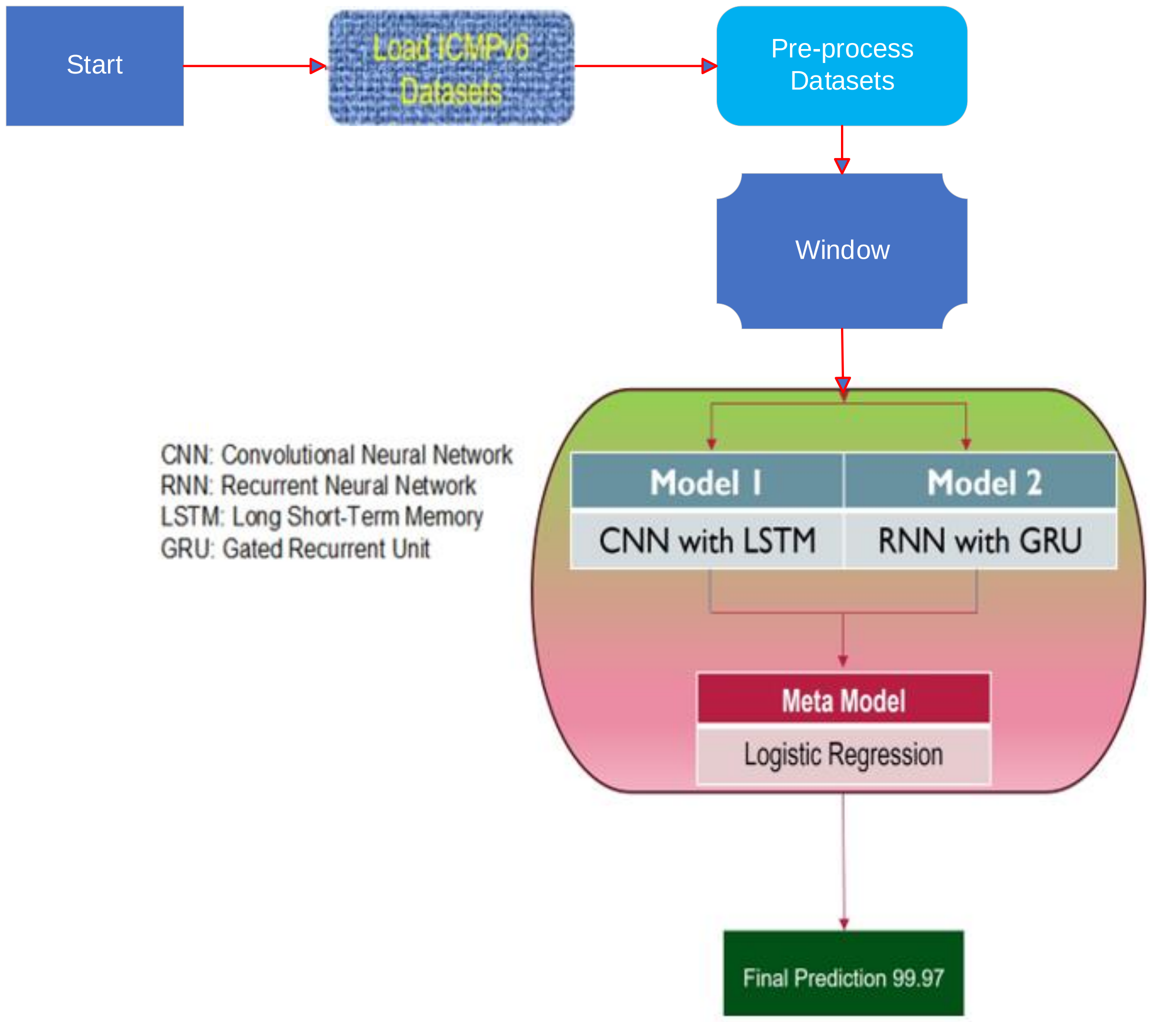


Figure 9: Logical diagram of the proposed Model

Figure 9 illustrates a Logical block diagram of the proposed Model. Specifically, Models 1 and Model 2 are selected for scrutinizing the processed datasets, with a focus on identifying ICMPv6 volumetric DDoS attacks. Following the application of these techniques, the outputs from both models are amalgamated and input into a regression model to enhance the efficacy of DDoS attack packet detection and prediction.

3.2.1 Convolution Neural Networks(CNN)

The principle behind CNNs is inspired by the human visual system's ability to process and recognize visual patterns. CNNs are part of deep neural networks used for processing and analyse large data like images and videos. They are highly efficient for tasks like image recognition, object detection, image segmentation, etc. The key principle of CNNs is the use of convolutional layers, which perform convolution operations on the input data. Convolution involves sliding a small filter (also called a kernel) over the input image and computing dot products between the filter and local patches of the image. This operation enables the network to discover and identify several features like edges, textures, and more complicated sequences.

A CNN should have at least one layer relating to convolution operations in its architecture. CNNs are successful in learning local attributes. CNNs are highly fast to run during training and conjecture due to shared kernels. CNNs use one-dimensional to have minimum computational cost and good performance for simple classification challenges (Ma, Zhou, and S. Wang 2019). CNN has a good understanding of the sensory neuronal response field and shares weights that successfully minimize the parameters of training, besides reducing the complication of a network architecture model. The training of CNN mainly uses the forward propagation algorithm to learn the connected layer weights, bias, and other parameters (J. Chen et al. 2019). The structure of CNN mainly consists of the following layers, which are briefly explained in Figure 10

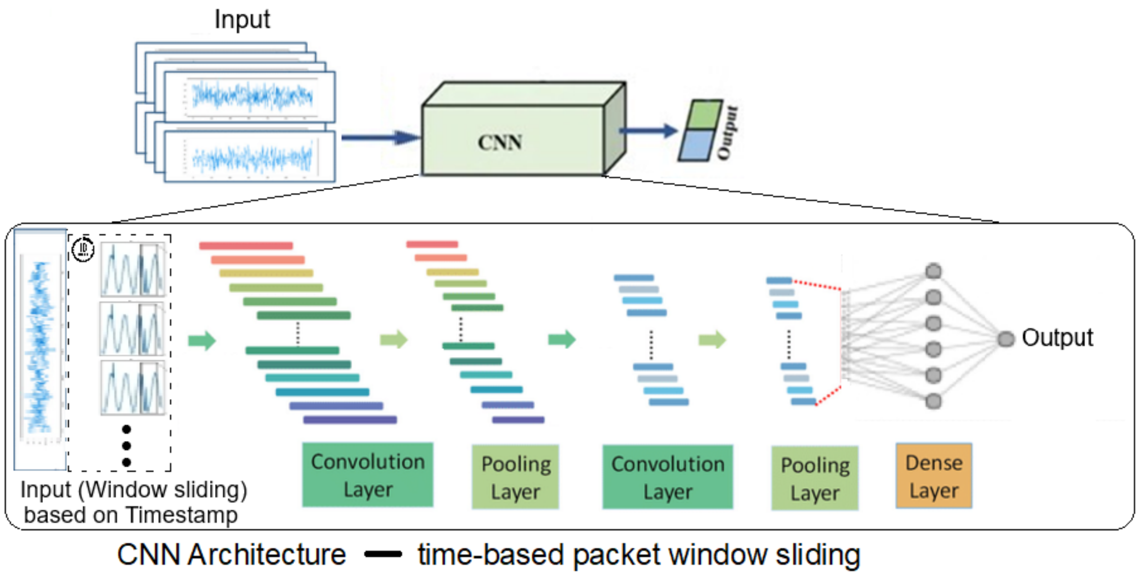

Input
CNN
Output
Input (Window sliding) based on Timestamp
Convolution Layer
Pooling Layer
Convolution Layer
Pooling Layer
Dense Layer
Output
CNN Architecture — time-based packet window sliding

Figure 10: Illustrates a Simple CNN block

a. **“Input layer:** The CNN needs the network status i.e. a m×m array describing the traffic of each link in a network containing m nodes. In addition, the Input Layer will combine several network statuses in a time sequence to demonstrate a trend of continuous network changes.

b. **Convolution layer:** It is a significant part of the entire CNN network layers. Each input matrix in the Convolution Layer is the result of the previous output. It can be calculated based on the computing formulae:-

$$Ym,n = f(\sum_{i=0}^{F=1}\sum_{j=0}^{F=1}\sum_{t=0}^{t-1} Xm+i,n+j,t * Fi,j,t + Br)$$

Where Ym, n is the eigenvalue calculated by the network traffic matrix via Convolution, and f is the ReLU or Ø (Sigmoid) activation function that is defined as max(0,x), Xm+i,n+j,t which is newly entered matrix, or the result of the last Convolution calculation. Fi,j,t and Br are filters. The learning method here uses matrix multiplication and calculates based on each result of the previous output to be input into the Convolution Layer. This calculation will result in a linear relationship between the output and the input, but practically, the occurrence of the attacks is not completely linear. Therefore, to make the training model more perfect and accurate, it is necessary to adjust the nonlinear excitation function. Based on the activation function, in case the input is a positive value, then it is directly output. If the input is not a positive value, the output is 0. The activation function is effective in solving the gradient explosion and the gradient disappearance problem compared with the excitation functions such as Sigmoid and tanh. Complex mathematical operations such as indices greatly reduce the computational complexity and greatly increase the speed of convergence.

c. **Pooling layer:** This layer is to remove the similarity and the excess of overfitting. The matrix is divided into several rectangular regions, and in each rectangular region, it selects the largest value as the output. The remaining smaller values are rounded off, thus reducing the excessive feature parameters and increasing the computing efficiency.

d. **Non-Linearity (Activation):** After each convolutional or pooling layer, a non-linear activation function is applied elementwise to the output. ReLU (Rectified Linear Unit) is a commonly used activation function in CNNs.
e. **Fully connect layer:** In this layer, after converting the Filter array of the Convolution Layer into a one-dimensional matrix, the forward propagation approach is used to calculate the relevant error value, update its weight value, and output the result to the Output Layer.
f. **Training and Backpropagation:** CNNs are trained using backpropagation, which involves iteratively adjusting the network's biases and weights to reduce the difference of predicted outputs and basic targets. This is usually accomplished using flawless algorithms such as Stochastic Gradient Descent (SGD).
g. **Weight Sharing and Local Connectivity:** CNNs utilize weight sharing and local connectivity to minimize the number of attributes and create a network efficient based on the computing. Weight sharing allows a single feature detector (filter) to be used at multiple spatial positions, and local connectivity ensures that each neuron present in the layer is connected to a modest region of the earlier layer.
h. **Hierarchical Representation:** CNNs learn to represent features hierarchically, starting from low-level features (e.g., edges, corners) and gradually progressing to more complex and abstract features (e.g., object parts, high-level structures).
i. **Output layer:** This layer is the output of a fully connected Layer that distinguishes between the probability of attack and the probability of non-attack based on the selection of the highest probability value in the Output Layer thus using the predicted value of the CNN to complete the judgment of attack detection"

(Y.-H. Chen et al. 2020).

3.2.1.a Merits of CNN

1. "**Spatial Hierarchy:** CNNs use convolutional layers to process small, localized regions of the input, capturing spatial hierarchies. This is critical for

jobs like image identification where local features are important. Convolutional layers share weights across different parts of the input, significantly reducing the number of parameters and computations required.

2. **Translation Invariance:** CNNs are designed to identify particular sequences irrespective of their status in the visual field. This process known as translation of invariance, is achieved through pooling layers that minimize the spatial elements while preserving the significant features.
3. **Feature Extraction:** CNNs automatically learn and extract hierarchical features from the input data, which are crucial for classification, object detection, and other vision tasks. Lower layers might detect edges and textures, while higher layers might detect shapes and objects.
4. **Reduction of Parameters:** Using convolution and pooling operations, CNNs reduce the number of parameters compared to fully connected networks. This makes training more efficient and reduces the risk of overfitting.
5. **Handling Large Inputs:** CNNs are well-suited for handling large inputs like high-resolution images. The hierarchical structure allows them to break down the complexity and focus on local patterns, making it computationally feasible to process large images.
6. **Strong Performance in Practice:** CNNs have demonstrated superior efficiency in a wide area of domains and relevant applications, including image classification, and segmentation, They have become the de facto standard in these domains.
7. **Flexibility and Adaptability:** CNN architectures can be adapted and extended to various tasks beyond image recognition, such as video analysis, natural language processing (with 1D convolutions), and even playing board games.
8. **Robustness to Variations:** CNNs are robust to variations in the input, such as changes in lighting, orientation, and partial occlusions. This robustness is a key factor in their success in real-world applications.
9. **End-to-end Learning:** CNNs support perfect learning, i.e., they are capable of learning to map raw input data directly to the desired output. This simplifies the learning process and reduces the need for manual feature engineering.

10. **Transfer Learning:** Prior to training CNN models, they can be fine-tuned on fresh tasks with limited data, leveraging the knowledge gained from large datasets. This transfer learning capability accelerates training and improves performance on related tasks"

(Mittal, K. Kumar, and Behal 2023).

Overall, the principle of CNNs is to automatically learn hierarchical and translation invariant features from the input data, which makes them highly effective for tasks involving huge data.

3.2.2 Long Short-Term Memory

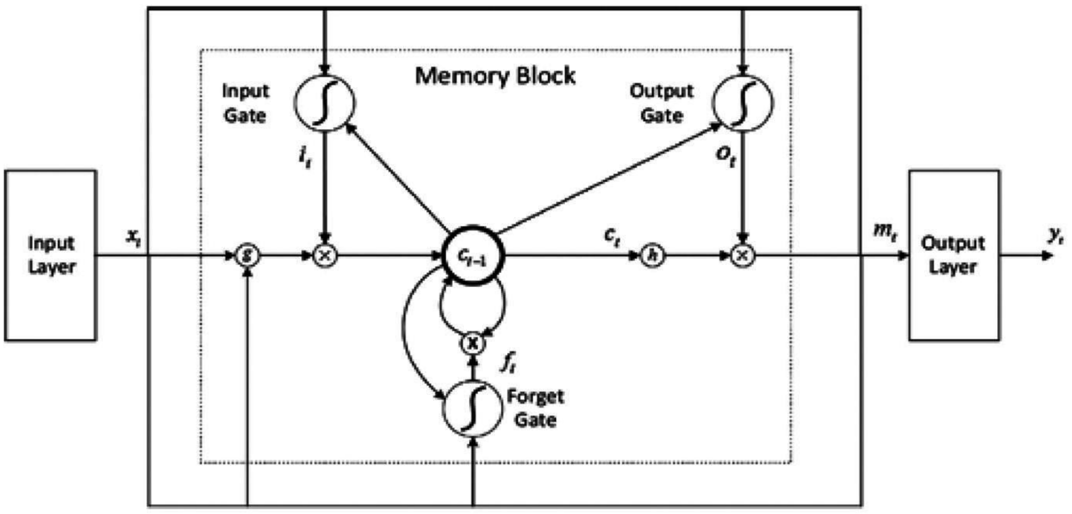


Figure 11: LSTM Architecture (Alguliyev, Aliguliyev, and Abdullayeva 2019)

In Figure 11, there are three gates namely input, output, and forget which are the regulatory gates for the information flow that is into and out in a cell which are memorized over arbitrary time intervals. The important functionality of the LSTM cell is to decide which value of the gate should be deleted or retained. For this, the forget gate plays a role in information coming from a prior unknown state (ht-1) from the current input (xt). On the combination of these values, it applies the *σ* function to get the values between 0 and 1. Any information value closer to 0 will be dropped otherwise it will be forwarded to the next step to process with the earlier cell state (ct-1). The final result of (ht-1+xt) is passed through another *σ* and tan h activation function respectively (Alshra'a, Farhat, and Seitz 2021).

3.2.3 Recurrent Neural Networks

In RNN, there are two types, one is a Feed-forward propagation, and the other is feed-backward propagation or Bidirectional. RNN performs the same action for every input data, whereas the output of the current input varies based on the earlier calculation. Once it produces the output, it is then fed back into the recurrent network. In Feed-forward RNN learns through the previous steps and produces the output as an input to the after node in the unseen layer. The memory of nodes stores the required information to be used for learning in future time steps. In a Bidirectional RNN, it connects two unseen layers to run in opposite directions. This allows the unseen layers to have information related to the previous and the next state (Nazih et al. 2020).

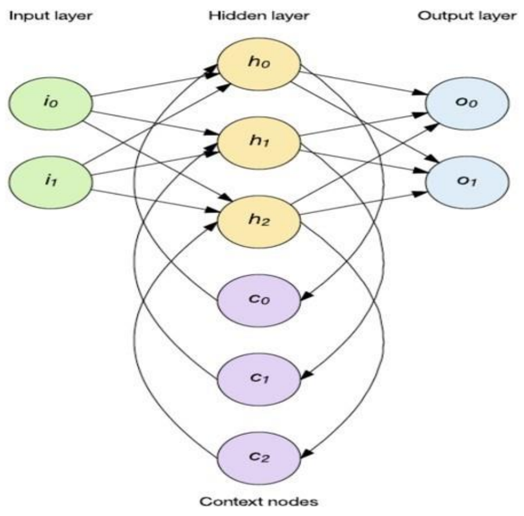


Figure 12: Recurrent Neural Network (RNN) (Alguliyev, Aliguliyev, and Abdullayeva 2019)

In theory, RNN operation can be illustrated as a memory for storing unbounded history based on previously processed elements shown in Figure 12. This means that at every point in time and instance, the stored history is used to predict the next output from the process. For instance, if W indicates the weight of neurons relating unknown state S. V indicates the weight of neurons relating to the unknown state S and the output O. U indicates the weight of neurons relating the inputs X and the unknown state S. Notice that all the 3 weights at any point in the process of the RNN operating will have the same value but the values will be different in the case of Conventional Neural Network. Because the same task is processed at each step with different input attributes. This reduces the total number of attributes that RNN will learn. To revise these weights, Back Propagation Through Time (BPTT) is to be used (Alguliyev, Aliguliyev, and Abdullayeva 2019).

Simple example: C0 = John / June, C1 = France / French. These values are assigned to C0 and C1. If John lived in France for a long time and could speak French fluently. Who lived in France? John; Similarly, June can also speak French fluently as it is her mother tongue. Whose mother tongue is French? June. Assigned values will change based on the context, and those need to be retained or discarded accordingly.

3.2.3.a Merits of RNN

RNN is an artificial neural network designed for processing sequential data.

1. The key principle behind RNNs is their ability to maintain a hidden state that captures information about previous elements in a sequence and uses it to process the current element. This hidden state acts as a form of memory that allows RNNs to collect temporal dependencies and patterns in sequential data.
2. The fundamental operation of an RNN involves iterating over a sequence of inputs while updating the hidden state at each step.
3. This hidden state is passed along from one step to the next, allowing the network to maintain information about previous inputs. Mathematically, the unknown state at instance time step t is a function of the input at given instance step t and the earlier unknown state: $ht = f(ht - 1, xt)$ Where: ht is the unknown state at given time step t. $ht - 1$ *is the unknown state at time step t-1 (earlier hidden state).* xt is the input at instance step t.
4. RNNs can be used for various tasks related to sequential data, like natural language processing (NLP), speech identification, music creation, video analysis, and more. However, conventional RNNs have weaknesses in

capturing extended dependencies because of the vanishing gradient problem, which can make them struggle with retaining information from distant past time steps.

5. To address these limitations, more advanced RNN architectures have been developed, such as LSTM networks and GRUs. These architectures incorporate gating mechanisms that enable the network to control the flow of information and gradients, making them better suited for capturing long-term dependencies

(S. Sumathi and Lokesh 2021).

In summary, the principle underlying RNNs is their ability to process sequential data by maintaining an unknown state that encodes information from earlier time steps, allowing them to capture temporal patterns and dependencies.

3.2.4 Gated Recurrent Unit (GRU)

In RNN, a vanishing Gradient problem occurs when RNN starts with backpropagation multiplying a tiny number by the weight values, which results in diminishing. To avoid such issues, GRU is used. It has only two gates. Figure 13 illustrates the Architecture of GRU.

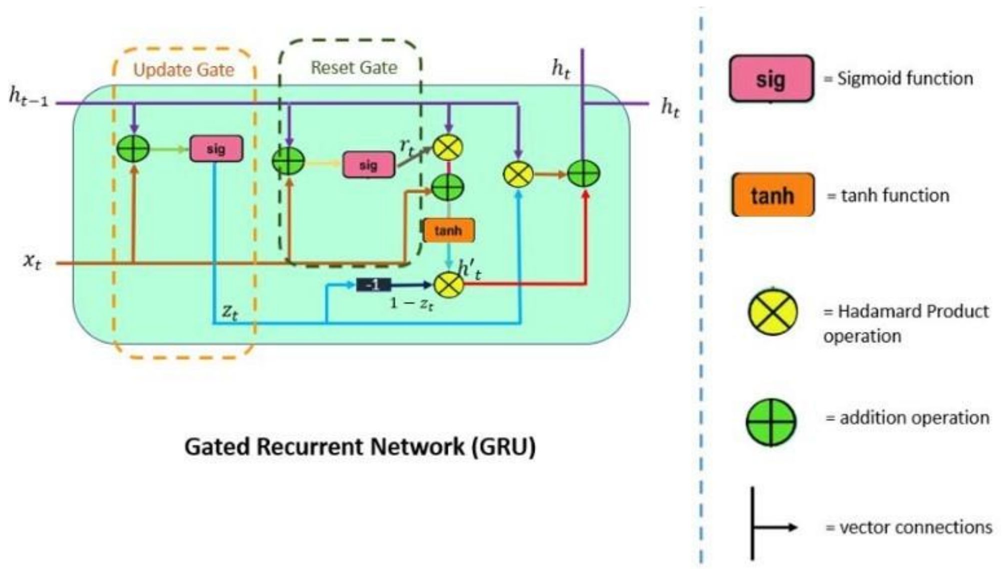


Figure 13: GRU Architecture (I. Ahmad, Z. Wan, and A. Ahmad 2023).

The core function of a GRU is to maintain a hidden state vector, which stores information about the sequence it has seen so far. The hidden state is updated at the instance of the time step based on the present input and the earlier hidden state. GRUs incorporate a gating mechanism that helps control the flow of information through the network. This gating mechanism consists of two gates:

a **A Reset Gate:** Determines which information from the previous state should be forgotten or reset.

b **Update Gate:** Controls how much of the new state is a blend of the old state and the candidate's new state.

GRU is responsible for deciding which parts of the previous hidden state to forget and which parts to update. It does this by considering the current input and the earlier unknown state. If the reset gate value is close to 1, it means that the model should mostly use the earlier unknown state. If it's close to 0, it means that most of the information should be replaced with the new input. The update gate helps the model decide how much of the new candidate state to retain. A value close to one implies retaining a lot of new information, while a value close to zero implies relying more on the old state. Candidate New State: The candidate's new state is computed based on the current input and the earlier unknown state, modified by the reset gate. This candidate state is used to update the unknown state (Alshra'a, Farhat, and Seitz 2021).

3.2.5 Ensemble Stacking

Advanced ensemble techniques enhance predictive performance by amalgamating multiple machine-learning models. Stacking elevates ensemble learning by training a meta-model, or second-level model, to predict based on the outputs of various base models. It's a versatile method enabling experimentation with different base models and meta-models, empowering the construction of robust and highly accurate machine learning models (Sayed et al. 2022).

The reason for opting for stacking in the proposed model is that, from Figure 8 at the beginning of this chapter, if the detection accuracy of individual models is close to each other or if any model has a lower score, stacking can be employed as it is a powerful ensemble learning technique improves predictive performance by combining the strengths of multiple models (S.Sumathi and Lokesh 2021). Wan employed the Stacking method in the stock market by integrating TCN, CNN, LSTM, and GRU classifiers into a stacked model, creating an ensemble learning approach. Their study focused on utilizing CSI300 index stock market datasets with a primary objective of minimizing the prediction error, specifically the Mean Absolute Error (MAE). Their proposed model achieved an accuracy rate of 86.3% (Wan et al. 2022). Sayed introduced a multi-classifier model with stacking ensemble deep neural networks to discern different types of DDoS attacks, targeting to tackle prevalent challenges in this domain. Their innovative hybrid model consists of CNN, LSTM, and GRU architectures. The study demonstrates the effectiveness of the ensemble technique in enhancing the performance of the model, particularly when evaluated with extensive datasets like CIC-DDoS2019, achieving an impressive accuracy rate of 89.4% (Sayed et al. 2022). Ali introduced an innovative ensemble approach based on stacking, demonstrating its superior performance over the current contemporary Network Intrusion Detection Systems (NIDS). His assessment encompasses diverse attack scenarios, on a network topology crafted using the Graphical Network Simulator-3 (GNS-3). Leveraging the CIC Flow Meter, they extracted key flow features for each attack, subjecting them to comprehensive analysis. By employing various machine learning approaches on the extracted traffic dataset features, with their respective performances. Notably, his findings reveal that the stacking-based ensemble approach emerges as the most promising accuracy score of 98.24% (Ali et al. 2023). The stacking mechanism is briefly explained with a pictorial block diagram in Figure 14.

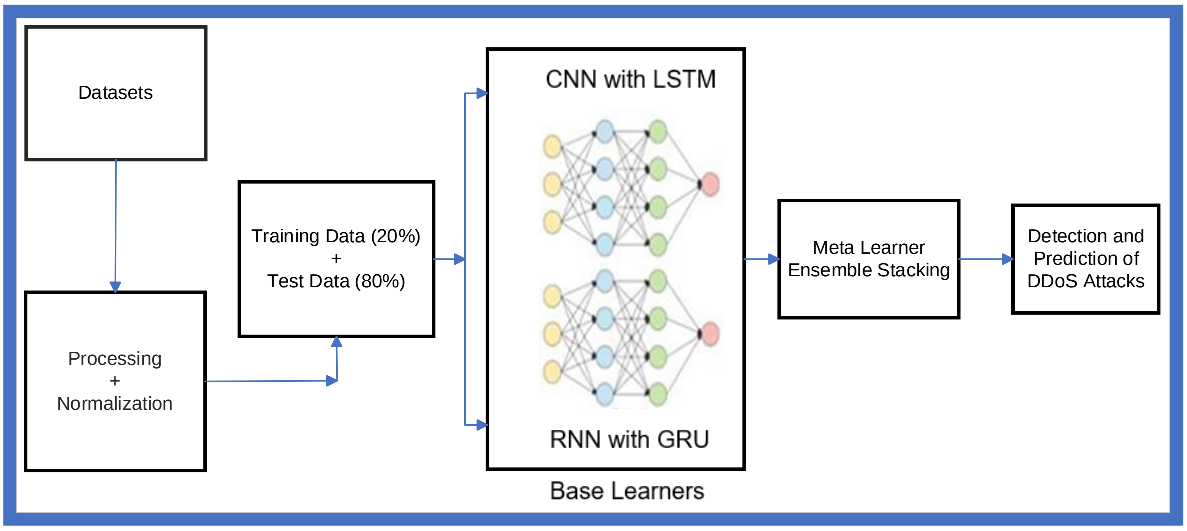

Datasets
Processing + Normalization
Training Data (20%) + Test Data (80%)
CNN with LSTM
RNN with GRU
Base Learners
Meta Learner Ensemble Stacking
Detection and Prediction of DDoS Attacks

Figure 14: Pictorial block diagram of Stacking

1. **"Train Base Learners:** Multiple base learners are trained on the same dataset. These can be different types of models (e.g., ML, neural networks, both) or the same type of model with different hyperparameters.
2. **Generate Predictions:** Once trained, each base learner makes predictions on the training data (or on a validation set if using cross-validation). These predictions become the input features for the next step.
3. **Train Meta-Learner:** The meta-learner is trained taking the output of the base learners as its input values treating it as features and the original target variable as the target. The meta-learner learns how to conglomerate the predictions from the base learners to improve overall performance**"**.

(Mohammed and Kora 2023)

3.2.4.a Merits of Stacking

Stacking is an ensemble learning technique that combines multiple models for better predictive performance. Merits of stacking are given below:

1. Increased accuracy, model diversity, and handling of heterogeneous data.
2. Stacking reduces overfitting by combining models with different sources of error.
3. Flexibility in model selection allows experimenting with various algorithms based on dataset characteristics.
4. Stacking adapts well to model changes, enabling continuous improvement with new and improved models.
5. It excels in capturing complex, nonlinear relationships in data, making it effective for intricate datasets.
6. Stacking improves robustness by reducing the impact of outliers or noisy data (Zhao et al. 2022).

3.3 Model-1 and Model-2

Issa has proposed a model related to a combination of CNN and LSTM having 7 layers indicating high quality of detecting DDoS attacks. Initially, he took CNN and Maxpooloing repeating twice and placed the LSTM layer deriving to an output layer using the SoftMax activation function. He assigned Filter = 10 Kernel size =3 and strides = 1 using ReLu as an activation function for the internal layers. He employed this combination neural network on NSL-KDD (IPv4 data) and achieved 99.20% accuracy. Taking his research work has a good support, in the current research, Model-1 was implemented with the same layers with different values and different activation functions and received a decent result of up to 80% accuracy using on different datasets including NSL-KDD. The combination of this neural network retained the same, however, changed the layer position and its values with different activation functions. On receiving good results and taking this as a good inspiration a similar kind of combination i.e. RNN with GRU is designed and tried using the same datasets achieving 84% of accuracy. Figure 15 a and b illustrates Model-1 and Model-2.

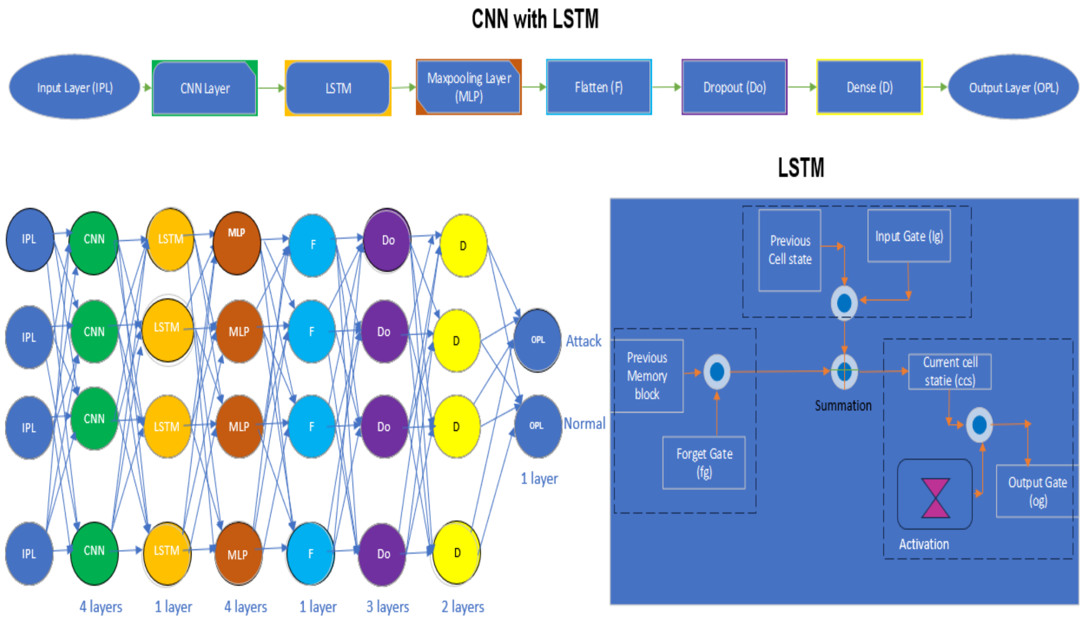


Figure 15-(a) Model-1 Block diagram

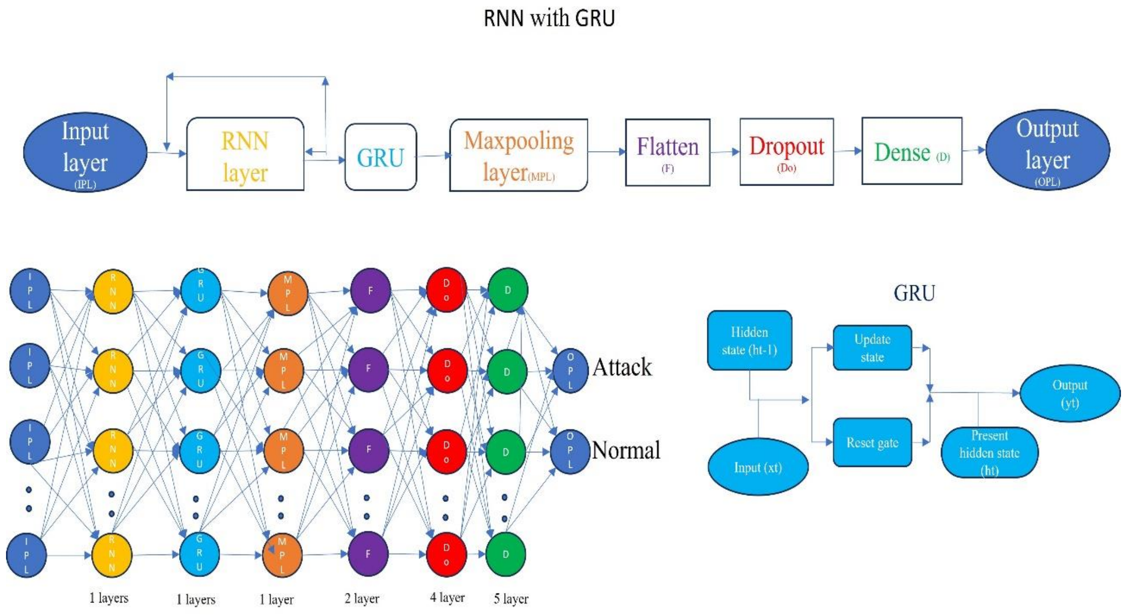


Figure 15-(b) Model 2 Block diagram

Normally in each model Input Layer (IPL) and output layer (OPL) are present. In between there exist hidden layers of the model designed by arranging them depending upon the chosen algorithms and respective activation function Relu at the required layer. The 14 parameters based on the datasets used are the traffic packet parameters meant to improvise the model's output. At any given time, t, the present input is a combination of IPL(t) and IPL(t-1). The output at any given time is carried back to the network to improvise the output.

1. **The model-1** architecture starts with an initial input layer consisting of a CNN with a kernel filter size of 128, using the ReLU activation function. Following this, an LSTM layer is employed with 100 filters, maintaining sequential order. The third layer comprises a MaxPooling layer and a one-dimensional convolutional layer with a pooling size of 1. Subsequently, repeating these CNN and Maxpooling for the next 3 times with CNN decreasing to 32, 16 and 8 values at each layer reshaping and focusing on the parameter's filtration, then followed to Flatten layer that reshapes the tensor into a vector, facilitating seamless transition between connected layers, particularly interfacing with two Dense layers paired with two Dropout layers alternatively. Dropout regularization is then applied, randomly deactivating 10% of neurons in a layer during training to mitigate overfitting. The 3rd Dense layer designed for classification tasks act as output layer typically undergoes a sigmoid activation function, generating probabilities for each class, such as ”Normal” and ”Attack.”
2. **The model-2** architecture starts with an initial input layer consisting of a RNN with a kernel filter size of 128, utilizing the ReLU activation function. Following this, a GRU layer relates to 64 filters, maintaining sequential order. The third layer comprises a MaxPooling layer with a pooling size of 1. Subsequently, 2 flattened layer reshapes the tensor into a vector, facilitating a seamless transition between connected layers, particularly interfacing with the 4 Dropout layers and 5 Dense layers alternatively pairing, deactivating 10% of neurons in a layer during training to mitigate overfitting. The final layer is a dense layer designed for classification tasks. The output of this dense layer typically undergoes a SoftMax activation function, generating probabilities for each class, such as ”Normal” and ”Attack.”

3.4 Models Learning

During the training phase, the weights were adjusted using the forward propagation in Model 1 and the back-propagation technique in Model 2. This process utilized the Sparse Categorical Cross-entropy or Binary Cross-entropy loss function to calculate error loss, which was then propagated across the network. All intermediate nodes between layers were interconnected, contributing their error values to the forward propagation. The entire network was enveloped by both forward and backward propagation mechanisms in respective models. For weight updating, the stochastic gradient descent optimizer for Adaptive Moment Estimation (ADAM) (Reyad, Sarhan, and Arafa 2023) was employed with a learning rate of 0.01, and parameter tuning set a minimum delta of 0.000001. To ensure effective training, the networks underwent 5 epochs, where each epoch involved one pass forward and backward of all data in the training set, or a comprehensive training cycle with a batch size of 5000. This iterative process enabled the network to gradually refine its weights and improve its performance over each epoch.

3.5 ES-Model Architecture

In this proposed model, Stacking is an ensemble learning technique that uses Logistic Regression as the meta-learner that combines predictions from various classifier models during the learning process, creating an adaptive mechanism i.e uses the predictions of base learners as input and then makes the final prediction.

In the training phase, the 3-fold cross-validation approach is followed so that the base learners are trained on 2 folds of training data while the prediction is made on the 3rd fold, and this process is iterated to obtain the prediction corresponding to the entire training set. This synthesis of classifications from multiple models enhances the overall predictive accuracy by leveraging the diverse strengths of individual learners, ultimately resulting in more precise predictions. Figure 16 illustrates the Architecture of the ES-Model.

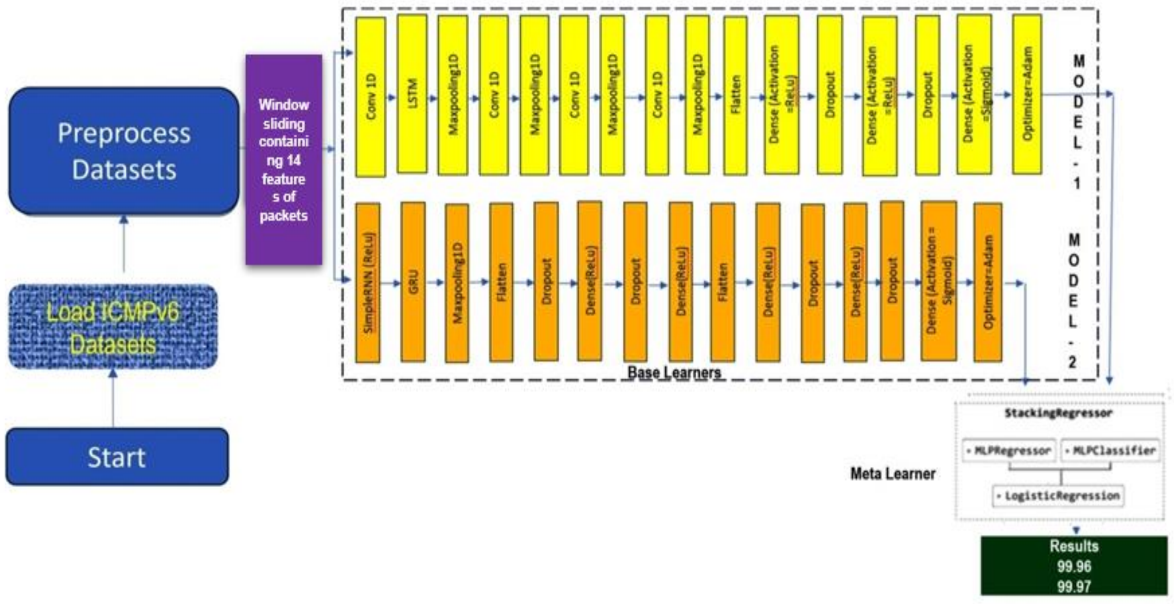


Figure 16: ES-Model Architecture

1. **CNN:** A CNN should have at least one of its layers involving convolution
2. operations. CNN is effective for learning local features. CNN is relatively fast to run during training and inference due to shared kernels. Mostly One-dimensional CNNs are suggested instead of two-dimensional convolution, as they have low computational cost and good performance for simple classification problems.
3. **LSTM:** The primary function of LSTM is to effectively learn from sequences by maintaining and updating information over extended periods. It has 3 gates, an Input gate to decide which values should be updated and added to the cell state. Forget gate to controls which information from the cell state should be forgotten or removed and the Output gate to determines what part of the cell state should be used to produce the output at the current time step.
4. **RNN**: It encompasses two key architectures: feedforward and bidirectional propagation. RNNs process data inputs iteratively, with outputs relying on past computations. In feed-forward RNNs, learning progresses sequentially, with each output feeding into the subsequent hidden layer node, retaining essential

information for future tasks. Conversely, bidirectional RNNs employ two hidden layers operating in opposing directions, accessing both preceding and succeeding states, thus enriching hidden layers with contextual information. Computational processes involve input vectors guided by equations, activation functions, and adjustments to weights via backpropagation. Despite their strengths, RNNs face challenges like the vanishing gradient problem, stemming from diminishing weight updates over time.

5. **GRU:** It stands as a formidable asset in handling sequential data tasks like language modelling, speech recognition, and time series prediction. At its core, it maintains a hidden state vector that evolves with each time step, influenced by both the present input and the preceding state. Key to its functionality is two gating mechanisms: the reset gate, which determines what to discard from the prior state, and the update gate, regulating the infusion of fresh state information. These gates strike a delicate equilibrium between assimilating new data and retaining pertinent historical context, effectively addressing the challenge of capturing distant dependencies while mitigating the vanishing gradient predicament.
6. **MaxPooling Layers:** Max pooling and average pooling are common techniques used to down-sample the feature maps. This helps reduce computational complexity and enhances the model's ability to recognize important features.
7. **Flatten:** The flattened vector effectively preserves the relevant features and provides a continuous input for the subsequent layers of the neural network.
8. **Dense layers:** These are associated with the connections between neurons in the fully connected layers. These weights determine the strength of the link between neurons and are discovered during training to map input features to the output predictions.
9. **Dropout:** It is a process of ignoring certain nodes in a layer at random during training. A dropout is a regular approach that avoids overfitting by assuring that no neurons are co-reliant with one another.
10. **Kernal or Filters:** It is a key parameter that is the weight of a small matrix employed across the data related to input to extract attributes in each layer or

the architecture. Each Kernal learns to detect a specific pattern or feature in the input data.

11. **Biases:** Each neuron (or feature map) typically has an associated bias term. These bias terms are added to the weighted sum of inputs before passing through an activation function.
12. **Adam:** It is derived from stochastic gradient descent (SGD) that combines ideas from RMSprop (Root Mean Square Propagation) and momentum. It is abbreviated as Adaptive Moment Estimation which is a flawless technique and highly known as an optimization technique mostly used in optimizing machine learning models, specifically used in ML and deep learning applications.
13. **Rectified Linear Unit (ReLu):** It is an activation function defined as f(x)=max(0,x) which means it returns “0” for any negative input value and returns the self-input value for any positive input. Mathematically, it’s a linear function where the output is linear for positive values and “0” for negative values.
14. **Sigmoid:** It is an activation function, often denoted as $\sigma(x)$, and represents a mathematical notation that derives any real-valued number to a value range of zero and one. In the context of artificial intelligence. and neural networks, the σ function is commonly used as an activation function.

The two outputs of the base layers in the ES-Model are fed to the Meta learner by a stacking technique for better predictive performance. The main advantages of stacking are to increase accuracy, model diversity, and handle heterogeneous data. It reduces overfitting by combining models with different sources of error. It provides flexibility in model selection and allows experimenting with various algorithms based on dataset characteristics. Stacking adapts well to model changes, enabling continuous improvement with new and improved models. It excels in capturing complex, nonlinear relationships in data, making it effective for intricate datasets and improving robustness by reducing the impact of outliers or noisy data.

3.5.1 Reason for selection of LSTM :

Rationale Behind Choosing the Model Combination, especially the LSTM (Long Short-Term Memory) model, was driven by the fundamental characteristics of the dataset and the nature of the research problem, which involves sequential data and temporal dependencies. Several factors were taken into consideration. Some of them are:

- **Nature of the Data – Temporal and Sequential Patterns:** The dataset comprises sequence-dependent features, where the order and timing of data points are crucial for accurate prediction or classification, a kind of continuous flow of packets. Traditional machine learning models, such as Random Forests or Support Vector Machines, treat each instance independently and fail to capture the underlying temporal correlations that may significantly influence outcomes.

- **Model Capability – Handling Long-Term Dependencies:** LSTM networks are a specialized type of Recurrent Neural Network (RNN) designed to retain information over long sequences. Unlike basic RNNs, LSTMs effectively mitigate the vanishing gradient problem and are capable of learning long-term dependencies through their gating mechanisms. This property aligns well with the problem at hand, where past events or feature states can influence future behaviour over a span of time.

- **Comparative Evaluation – Justification Over Alternatives:** From the research study, alternative architectures such as traditional RNNs, GRUs (Gated Recurrent Units), and CNNs (Convolutional Neural Networks) were considered. While GRUs offered a simplified alternative to LSTMs with comparable performance in some tasks, preliminary evaluations showed that LSTM slightly outperformed GRU in capturing nuanced temporal dependencies in this particular dataset. CNNs, although effective in pattern recognition, lacked the sequential memory needed for this task. Based on these points a novel combination CNN with LSTM and RNN with GRU architecture was designed.

- **Research Integrity – Theoretical and Empirical Support:** The choice of LSTM is supported by a substantial body of literature where similar sequence modelling tasks have successfully leveraged LSTM networks. Additionally, empirical validation through experiments on this dataset confirmed the model's robustness and superior performance compared to non-sequential models.

- **Compatibility with Analytical Framework – Interpretability Tools:** The LSTM model was integrated into a broader interpretability framework using SHAP, LIME, and permutation importance. Although neural networks are often criticized for their "black-box" nature, the use of these interpretability techniques ensured the model remained transparent and aligned with the principles of explainable AI, allowing for a clear understanding of which features influenced predictions over time.

3.6 Algorithms

==

Using Deep Learning Neural Network feature list trained and tested for CNN and LSTM using MLP :

Input packets : Packet traffic / flow Output packets : Good Packets Main key process of the technique:

==

CNN and LSTM

==

```
1:Load dataset
2: Split dataset into 80% training and 20% testing
3: T rain data ← dataset[0 : 80%]
4: T est data ← dataset[80% :]
5: Xtrain, Ytrain ← Train data
6: Xtest, Ytest ← Test data
7: while Batch = 5000 and Epoch = 5 do
8:          Conv1D layer
9:          LSTM layer
10:         for weights in hidden layers do
11:              MaxPooling1D
```

```
12:                 Flatten
13:                 Activation function: ReLU
14:                 Dense layer
15:                       Apply Dropout
16:                 if layer is final output layer then
17:                       Activation function: Sigmoid (σ)
18:                 end if
19:           end for
20:           Optimization function: Adam
21:     end while
22: CNN MLP ← Final output
```

==

RNN and GRU

==

```
1: Split dataset into 80% training and 20% testing
2: Train data ← dataset[0 : 80%]
3: Test data ← dataset[80% :]
4: Xtrain, Ytrain ← T rain data
5: Xtest, Ytest ← T est data
6: while Batch = 5000 and Epoch = 5 do
7:          Apply SimpleRNN layer
8:          for weights in hidden layers do
9:                Apply MaxPooling1D
10:                     Apply GRU layer
11:                     Apply Dense layer
12:                     if layer is final output layer then
13:                           Apply Activation function: softmax
14:                     end if
15:               end for
16:               Set Optimization function: Adam
17: end while
18: RNN MLP ← Final output
```

==

Stacking

==

```
1: Input: CNN-MLP, RNN-MLP
2: Estimator List ← {(CNN-MLP, classifier), (RNN-MLP, regressor)}
3: Stack Model ← StackingClassifier(Estimator List)
4: Logistic regressor = Stackmodel
5: Prediction:
```

6 : Ytrain pred ← Stack Model (Xtrain)
===

The above algorithm can be mathematically explained based on the proposed architectures and the ensemble stacking approach model.

1. Data Representation:

Assume the input network traffic data is represented as a sequence of features over time. Let X be the input data, where each instance xi∈X is a sequence of T time steps, and each time step has F features. Thus, the input data can be represented as a 3D tensor of shape (N,T,F), where N is the number of samples.

2. Model 1: CNN-LSTM

This model combines Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to capture both spatial and temporal patterns in the network traffic data.

- **Convolutional Layers:** The 1D convolutional layers extract local patterns from the input sequence. For a given input sequence $x \in \mathbb{R}^{T \times F}$ the output of a 1D convolutional layer with K filters of size w can be represented as:

$$h_j^{(l)}[t] = \sigma\left(\sum_{i=0}^{w-1}\sum_{f=1}^{F} W_{ji}^{(l)} x[t+i,f] + b_j^{(l)}\right)$$

  where $h_j^{(l)}[t]$ is the activation of the j-th filter at time step t in the l-th layer, $W_{ji}^{(l)}$ are the weights of the i-th element of the j-th filter in the l-th layer, $b_j^{(l)}$ is the bias of the j-th filter in the l-th layer, and σ is the activation function (e.g., ReLU). The model has multiple such convolutional layers with varying numbers of filters (64, 32, 16, 8) and a fixed kernel size of 2.
- **LSTM Layer:** The LSTM layer processes the temporal dependencies in the features extracted by the CNN. An LSTM unit at time step t takes the input ht (output from the previous layer) and the previous hidden state ct−1 and ht−1 to compute the current hidden state ht and cell state ct through a series of gates:

```
\begin{align*}
i_t &= \sigma_g(W_{xi}h_t + W_{hi}h_{t-1} + b_i) \quad \text{(Input Gate)} \
f_t &= \sigma_g(W_{xf}h_t + W_{hf}h_{t-1} + b_f) \quad \text{(Forget Gate)} \
o_t &= \sigma_g(W_{xo}h_t + W_{ho}h_{t-1} + b_o) \quad \text{(Output Gate)} \
\tilde{c}t &= \tanh(W{xc}h_t + W_{hc}h_{t-1} + b_c) \quad \text{(Cell Input Activation)} \
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad \text{(Cell State)} \
h_t &= o_t \odot \tanh(c_t) \quad \text{(Hidden State)}
\end{align*}
```

where W are weight matrices, b are bias vectors, σg is the sigmoid function, tanh is the hyperbolic tangent function, and ⊙ denotes element-wise multiplication. The model uses an LSTM layer with 100 units. The return_sequences=True argument indicates that the LSTM outputs a sequence of hidden states for each time step, which is then processed by the subsequent MaxPooling layer.

- **MaxPooling Layers:** The MaxPooling layers downsample the feature maps, reducing dimensionality and providing translational invariance. For a 1D input sequence s, the output of a MaxPooling layer with pool size p is:

$$y[i] = \max_{j=1}^{p} s[i \cdot p + j]$$

  The model uses multiple MaxPooling1D layers with a pool size of 1, which in practice doesn't perform downsampling but might be used for architectural consistency or as a placeholder.
- **Flatten Layer:** The Flatten layer transforms the multi-dimensional output of the convolutional and pooling layers into a 1D vector.
- **Dense Layers (Fully Connected Layers):** These layers perform linear transformations followed by a non-linear activation function. For an input vector x, a dense layer with m neurons and activation function σ computes the output vector $y \in \mathbb{R}^m$ as: $y = \sigma(Wx + b)$.
- **Dropout Layers:** Dropout layers randomly set a fraction (e.g., 0.2) of input units to 0 during training to prevent overfitting.
- **Output Layer:** The final dense layer with a sigmoid activation function outputs a probability score between 0 and 1, representing the likelihood of

the input traffic being a DDoS attack (1) or normal traffic (0). $\hat{y} = \sigma(w^T z + b)$ where z is the output of the previous dense layer, w and b are the weights and bias of the output layer, and σ is the sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$

- **Loss Function and Optimizer:** The model is compiled with binary cross-entropy loss, which is suitable for binary classification problems: $L(y, \hat{y}) = -(y \log(\hat{y}) + (1-y)\log(1-\hat{y}))$ where y is the true label (0 or 1) and $\hat{y}$ is the predicted probability. The Adam optimizer is used to update the model's weights during training to minimize this loss.

3. Model 2: RNN-GRU

This model utilizes a combination of SimpleRNN and Gated Recurrent Unit (GRU) layers to capture temporal dependencies.

- **SimpleRNN Layer:** A SimpleRNN layer computes the hidden state h_t at time step t as: $h_t = \sigma_h(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$ where xt is the input at time step t, h_{t-1} is the previous hidden state, W_{xh}, W_{hh} are weight matrices, b_h is the bias vector, and σ_h is the activation function (ReLU in this case). The model uses a SimpleRNN layer with 128 units.
- **GRU Layer:** The GRU is a more sophisticated recurrent unit with update and reset gates that help in capturing long-range dependencies more effectively than SimpleRNN. The computations within a GRU unit are:

```
\begin{align*}
z_t &= \sigma_g(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \quad \text{(Update Gate)} \
r_t &= \sigma_g(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \quad \text{(Reset Gate)} \
\tilde{h}t &= \tanh(W{xh'}x_t + W_{hh'}(r_t \odot h_{t-1}) + b_{h'}) \quad \text{(Candidate
Hidden State)} \
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \quad \text{(Hidden State)}
\end{align*}
```

  The model uses a GRU layer with 64 units.
- The rest of the layers (MaxPooling1D, Flatten, Dropout, Dense with ReLU, and the final Dense with sigmoid) function similarly to those in Model 1. The model is also compiled with binary cross-entropy loss and the Adam optimizer.

4. Ensemble Stacking

Ensemble stacking combines the predictions of multiple base models to improve overall performance. In this case, the base models are a CNN-LSTM model (represented by the CNN function), an RNN-GRU model (represented by the RNN function), and a Logistic Regression classifier.

- **Base Models:** Let M1(X), M2(X), and M3(X) represent the prediction outputs (probabilities for DDoS attack) of Model 1 (CNN-LSTM), Model 2 (RNN-GRU), and Model 3 (Logistic Regression) for a given input X.
- **Meta-Learner:** The stacking approach involves training a meta-learner (in this case, implicitly suggested but not explicitly defined in the provided snippet as a separate trainable model) on the predictions of the base models. A common approach is to use a simple model like Logistic Regression as the meta-learner.
- **Prediction with Stacking:** For a new input x, the prediction process involves:
  - Obtaining predictions from each base model: $\hat{y}_1 = M_1(x)$, $\hat{y}_2 = M_2(x)$, $\hat{y}_3 = M_3(x)$.
  - Combining these predictions into a new feature vector: $z = [\hat{y}_1, \hat{y}_2, \hat{y}_3]$.
  - Feeding this feature vector z to the meta-learner M_{meta}: $\hat{y}_{ensemble} = M_{meta}(z)$ If the meta-learner is a Logistic Regression model, its prediction would be: $\hat{y}_{ensemble} = \sigma(w_1\hat{y}_1 + w_2\hat{y}_2 + w_3\hat{y}_3 + b)$ where w_1, w_2, w_3 are the weights and b is the bias learned by the meta-learner during training.

In summary, DDoS detection is achieved through the following mathematical processes:

1. **Feature Extraction:** CNN layers in Model 1 extract spatial features from the time series network traffic data.

2. **Temporal Dependency Modeling:** LSTM layers in Model 1 and RNN/GRU layers in Model 2 learn long-range temporal dependencies in the extracted features or the raw input data.
3. **Dimensionality Reduction and Abstraction:** MaxPooling and Flatten layers reduce the dimensionality of the feature maps and prepare them for fully connected layers.
4. **Classification:** Dense layers learn complex non-linear relationships between the extracted features and the target class (DDoS or normal), with the final sigmoid activation providing a probability of a DDoS attack.
5. **Regularization:** Dropout layers help prevent overfitting by randomly dropping out neurons during training.
6. **Ensemble Aggregation:** Ensemble stacking combines the predictions of multiple diverse models (CNN-LSTM, RNN-GRU, and Logistic Regression) using a meta-learner to produce a more robust and accurate final prediction. The meta-learner learns the optimal way to weigh or combine the individual model predictions.

The entire process involves learning the optimal weights and biases of all the layers in each model through backpropagation and optimization algorithms like Adam, based on a labeled training dataset of network traffic. The goal is to train models that can accurately map input network traffic patterns to the probability of it being a DDoS attack.

3.7 Implementation Specifications:

The algorithm developed for the proposed model was implemented using Python on an HP laptop with an i7 processor, 64 GB RAM, and primarily using the T4 GPU runtime of Google Colab. Key modules such as Scikit-learn, Keras, and TensorFlow were applied. Model-1 consisted of a base classifier with 15 layers and an output layer, while Model-2 consisted of a base classifier with 14 layers and an output layer, where both models used the Adam optimizer. Each model was trained with 5 epochs with a batch equal of 5000. The outputs from these models were then used as inputs for an ensemble stacking meta-classifier, employing a regression classifier with cross-validation up to 3 folds, resulting in promising final predictions. Detailed information about the datasets, experiments, and validation is discussed in the following chapters.

3.8 Handling New and Unseen Networks:

Handling New and Unseen Attacks

A key challenge in real-world deployment of detection of DDoS attacks is their ability to generalize to new or unseen types of attacks. In this context, the proposed system's performance on such attacks would depend on several factors, including the diversity of the training data, the robustness of the learning algorithm, and the system's architecture.

While the model is primarily trained on known attack categories from the training set, it incorporates mechanisms that enhance generalization:

- **Feature-based generalization:** The use of high-level abstracted features (especially those with high importance across interpretability methods such as SHAP, LIME, and permutation importance) increases the likelihood that the system can recognize patterns indicative of anomalous behaviour, even if the specific attack vector is novel. (The discussion about this point is elaborated in Chapter 5 Experiments under Phase 4, which is mainly focused on the proposed model behaviour based on the results obtained).
- **Anomaly detection capability**: Though the system is primarily a supervised classifier, additional mechanisms (e.g., thresholding prediction confidence, monitoring rare feature patterns) can be integrated to flag low-confidence or unfamiliar inputs as potential anomalies, which can then trigger further investigation. This rare feature pattern investigation is outside the scope of this research and can be extended for future research.

3.9 Responding to Attacks Not Seen During Training:

For attacks that the model has not encountered during training (zero-day or zero-shot attacks), the system is able to respond appropriately. This research is focused mainly on detection of DDoS attack and other attacks is outside the scope of this research and can be extended for future research.

3.10 Handling Concept Drift:

In dynamic environments, data distributions may shift due to changes in network behaviour, user activity, or emerging attack vectors, this phenomenon is referred to as concept drift. The system addresses concept drift through the following strategies:

- **Monitoring model performance over time:** By evaluating metrics (e.g., precision, recall, F1-score) on recent traffic data in a sliding window, the system can detect performance degradation that may indicate concept drift.
- **Incremental or periodic retraining:** Using new labelled data or pseudo-labeled data from high-confidence predictions, the model can be updated regularly. This allows it to adapt to evolving patterns in both normal and malicious traffic.
- **TimeSeries-aware model validation**: The use of TimeSeriesSplit during model evaluation and tuning provides better robustness to temporal shifts, making the system more resilient to changes in traffic over time, which is implemented in Chapter 5, Experiments Phase 4, and justified through results.

Chapter 3 presents the design artifact of the proposed model, detailing each component block along with the mathematical formulations used to classify network traffic as benign or malicious for DDoS detection. It includes a step-by-step representation of the model’s workflow using algorithmic notation and explains the rationale behind the deep neural network combination. Additionally, the chapter outlines strategies for handling new and previously unseen traffic, ensuring adaptability to evolving threat patterns.

4 DATASETS

This chapter discusses the datasets used in this research, highlighting their importance in providing empirical evidence to support hypotheses, validate models, and draw conclusions. Datasets, being structured collections of data, are essential for analysis and research across various fields. The dataset consists of network traffic recordings obtained from previous research endeavours aligned with the objectives of this proposal. These datasets, widely disseminated by diverse institutions, research groups, and relevant entities, have been made publicly accessible, thereby enabling contemporary researchers to incorporate them into ongoing projects. It is worth noting that a majority of these datasets are specifically associated with IPv4. Organizations and institutes use datasets from real-time attacks for research to find mitigating solutions, while others generate datasets to address identified research gaps. They extend their services by openly providing these datasets, allowing researchers to perform similar experiments, validate their models, and draw conclusions based on the results. Below is a list of some of these organizations and a summary of their contributions.

1. The Centre for Applied Internet Data Analysis (CAIDA) was established in 1997 and administers network research and develops research groups to support huge data gathering, maintenance, and data sharing to the scientific research groups. It is an independent research group and analysis centre based at San Diego Supercomputer Centre, University of California. Its main focus areas are Measurement & Infrastructure, Research, Analysis and Data Collection. CAIDA Dataset 2007 is mainly used for DDoS and intrusion detection projects that were used earlier and hosted for new research projects and the page was updated in 2022. Summary CAIDA can be found at (CAIDA 2022).

2. CIC, based at the University of New Brunswick, is a leading Canadian hub for cybersecurity. Focused on innovation, disruptive technology, and groundbreaking research, the institute explores diverse domains, including big security data analytics, visualization, analysis, risk management, intrusion detection, malware

analysis, and more. It has various datasets like CIC-DDoS2019, ISCXIDS2012, NSL-KDD, CIC-IDS2017, and CIC-IDS2018, addressing different cybersecurity challenges such as intrusion detection and DDoS attacks. A summary of CIC can be found at (CIC 2021).

3. MAWI Lab, established and managed by Dr. Kensuke Fukuda, is primarily situated in Tokyo, Japan. The lab is dedicated to curating datasets focused on Distributed Denial of Service (DDoS) anomalies, specifically those associated with packet features. These datasets categorize anomalies into distinct labels, including anomalous, suspicious, notice, and benign. The lab diligently and consistently gathers data from researchers to contribute to ongoing research efforts in the field. A summary of MAWI Lab can be found at (MAWI et al. 2010).

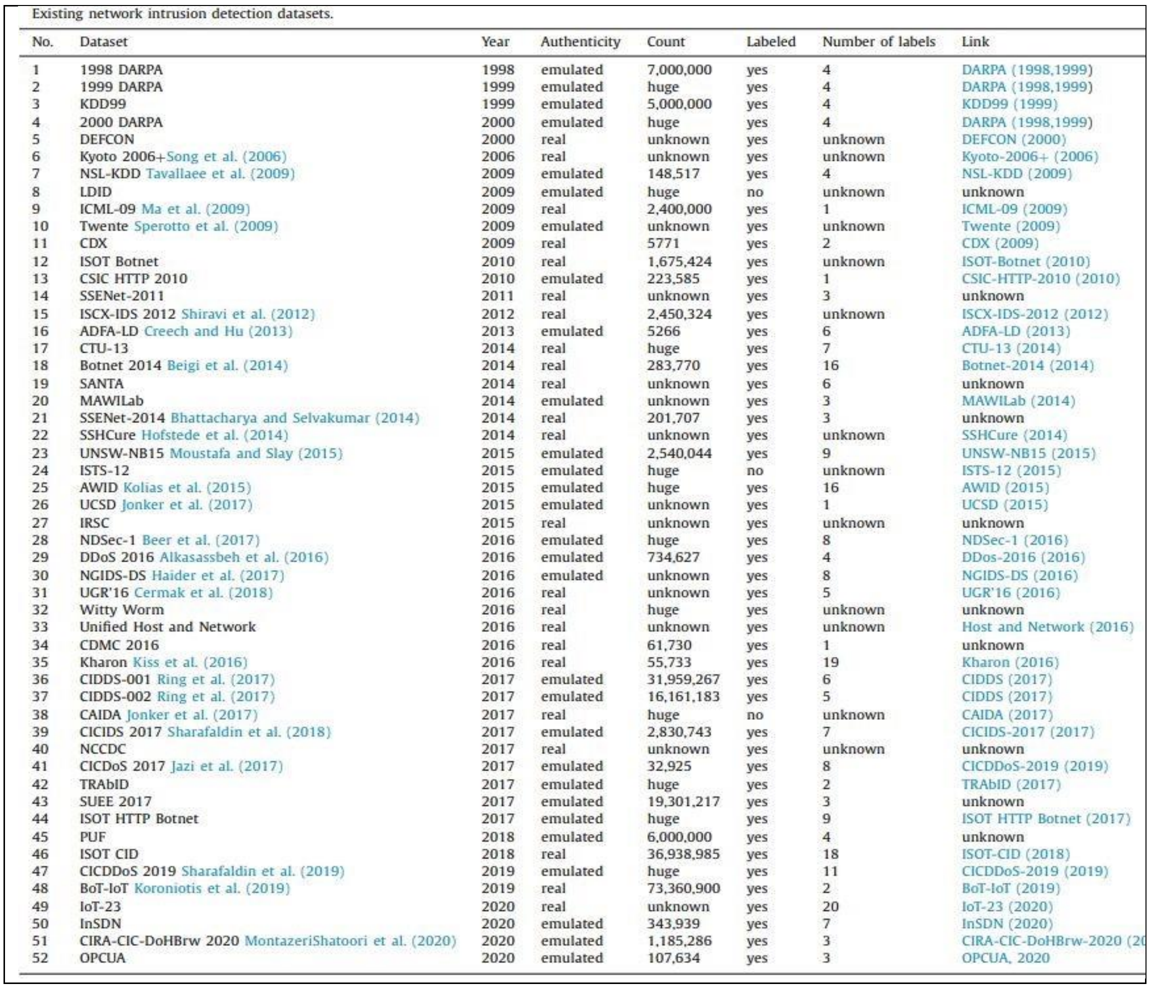
4. The KDD Cup 1999 Data set, created for the Third International Knowledge Discovery and Data Mining Tools Competition, is publicly hosted and archived by the University of California, Irvine. Designed in 1999, the dataset aimed to facilitate the development of network intrusion detectorspredictive models capable of differentiating between malicious intrusions or attacks and normal connections. Simulated in a military network environment, the dataset provides a standard set of auditable data, featuring a diverse range of intrusions. Since its inception, researchers have widely utilized this dataset in various projects to assess and refine their proposed techniques (KDDCup October 28, 1999).

5. The University of MIT's Lincoln Laboratory R&D openly hosts a diverse array of DARPA datasets, featuring a substantial collection of archived data. These datasets are made readily available to emerging researchers, enabling them to assess and refine their techniques. By utilizing these datasets, researchers can effectively evaluate the suitability of their projects related to traffic flow, specifically tailored to various attack scenarios. A summary of DARPA can be found at (DARPA July 2000).

Below are some of the researchers who have used the datasets from the above Organizations / Institutes for their research. Cheng et al. utilized CAIDA's "DDoS

Attack 2007” dataset, employing SVM, SMKL, and GMKL to enhance DDoS attack detection using a novel method ”Optimized Generalized Multiple Kernel Learning” thus achieving 86.2% accuracy in distinguishing attack flow from normal traffic (Cheng et al. 2019). Omer Kasim introduced an innovative approach for detecting Distributed Denial of Service (DDoS) attacks by leveraging Deep Learning and Autoencoder Support Vector Machine (AE SVM) techniques. The evaluation of his proposed method involved utilizing the CICIDS and NSL-KDD datasets. His approach demonstrated a high accuracy of 99.1% (Omer 2020). Laurens D’hooge and his team presented a study titled ”Inter-data Set Generalization Strength of Supervised Machine Learning Methods for Intrusion Detection,” which explores the assessment of generalization capabilities of inter-dataset using supervised ML methods in ID. The study specifically utilizes the CIC-IDS2017 and CSE-CIC-IDS2018 data sets. The research involved the application of a varied range of algorithms, where Support Vector Machine (SVM) yielded 90% accuracy during their evaluation (D’hooge et al. 2020). Kurniabudi Stiawan introduced a novel approach for Anomaly Detection by employing Data Set Feature Analysis with Information Gain, specifically using the CICIDS-2017 dataset. Focusing on substantial features within extensive network traffic, their analysis significantly enhances the accuracy of anomaly detection They evaluated their experiments using Random Forest, J48 classifier achieving 99.86 and 99.87 accuracy rates respectively (Stiawan et al. 2020). Jing has proposed a novel approach for countering DDoS flooding attacks through ”Network traffic fusion and analysis” using a Chinese Remainder Theorem-based Reversible Sketch (CRT-RS) and Modified Multi-chart Cumulative Sum (MM-CUSUM) applied to the WIDE Internet (MAWI) datasets -2015050, 20150304, and CICIDS2017. In their pursuit of heightened accuracy, they conducted additional experiments employing Multi-scale Principal Component Analysis (MSPCA) and information theory-based detection methods thus resulting in 79.36 percent accuracy on comparison (Jing et al. 2019). Ahmed Issa introduced an innovative deep-learning classification approach by combining two widely used algorithms, CNN and LSTM. The model was evaluated using the NSL-KDD dataset achieving an impressive accuracy rate of 99.20% (Issa and Albayrak 2023). Ieracitano introduced an innovative intrusion detection approach, combining statistical analysis with Autoencoder (AE) technology. In their study, they conducted a comparative analysis against LSTM and RNN techniques. Their experiments were carried out on the NSL-KDD dataset resulting in 84.21%, 87% accuracy representing binary classification and multi-classification respectively (Ieracitano et al. 2020). Wesam introduced a novel methodology named Clustering Using Representative (CURE), which underwent comprehensive evaluation through comparisons with alternative methods such as SVM, Fuzzy Estimator, Fuzzy Logic, Growing Hierarchical SOM (GHSOM), and k-means. The assessment utilized DARPA2000, CAIDA2007, and CAIDA2008 datasets, demonstrating an impressive accuracy rate of 97.25% specifically when applied to DARPA2000 datasets (Bhaya and EbadyManaa 2017). Besides the discussed datasets, further datasets that were used by different researchers are briefly provided in a tabular form shown in Table 7 (Yang et al. 2022).

Table 9: Review of Datasets Summary (Yang et al. 2022)



Existing network intrusion detection datasets.

No.	Dataset	Year	Authenticity	Count	Labeled	Number of labels	Link
1	1998 DARPA	1998	emulated	7,000,000	yes	4	DARPA (1998,1999)
2	1999 DARPA	1999	emulated	huge	yes	4	DARPA (1998,1999)
3	KDD99	1999	emulated	5,000,000	yes	4	KDD99 (1999)
4	2000 DARPA	2000	emulated	huge	yes	4	DARPA (1998,1999)
5	DEFCON	2000	real	unknown	yes	unknown	DEFCON (2000)
6	Kyoto 2006+Song et al. (2006)	2006	real	unknown	yes	unknown	Kyoto-2006+ (2006)
7	NSL-KDD Tavallaee et al. (2009)	2009	emulated	148,517	yes	4	NSL-KDD (2009)
8	LDID	2009	emulated	huge	no	unknown	unknown
9	ICML-09 Ma et al. (2009)	2009	real	2,400,000	yes	1	ICML-09 (2009)
10	Twente Sperotto et al. (2009)	2009	emulated	unknown	yes	unknown	Twente (2009)
11	CDX	2009	real	5771	yes	2	CDX (2009)
12	ISOT Botnet	2010	real	1,675,424	yes	unknown	ISOT-Botnet (2010)
13	CSIC HTTP 2010	2010	emulated	223,585	yes	1	CSIC-HTTP-2010 (2010)
14	SSENet-2011	2011	real	unknown	yes	3	unknown
15	ISCX-IDS 2012 Shiravi et al. (2012)	2012	real	2,450,324	yes	unknown	ISCX-IDS-2012 (2012)
16	ADFA-LD Creech and Hu (2013)	2013	emulated	5266	yes	6	ADFA-LD (2013)
17	CTU-13	2014	real	huge	yes	7	CTU-13 (2014)
18	Botnet 2014 Beigi et al. (2014)	2014	real	283,770	yes	16	Botnet-2014 (2014)
19	SANTA	2014	real	unknown	yes	6	unknown
20	MAWILab	2014	emulated	unknown	yes	3	MAWILab (2014)
21	SSENet-2014 Bhattacharya and Selvakumar (2014)	2014	real	201,707	yes	3	unknown
22	SSHCure Hofstede et al. (2014)	2014	real	unknown	yes	unknown	SSHCure (2014)
23	UNSW-NB15 Moustafa and Slay (2015)	2015	emulated	2,540,044	yes	9	UNSW-NB15 (2015)
24	ISTS-12	2015	emulated	huge	no	unknown	ISTS-12 (2015)
25	AWID Kolias et al. (2015)	2015	emulated	huge	yes	16	AWID (2015)
26	UCSD Jonker et al. (2017)	2015	emulated	unknown	yes	1	UCSD (2015)
27	IRSC	2015	real	unknown	yes	unknown	unknown
28	NDSec-1 Beer et al. (2017)	2016	emulated	huge	yes	8	NDSec-1 (2016)
29	DDoS 2016 Alkasassbeh et al. (2016)	2016	emulated	734,627	yes	4	DDos-2016 (2016)
30	NGIDS-DS Haider et al. (2017)	2016	emulated	unknown	yes	8	NGIDS-DS (2016)
31	UGR'16 Cermak et al. (2018)	2016	real	unknown	yes	5	UGR'16 (2016)
32	Witty Worm	2016	real	huge	yes	unknown	unknown
33	Unified Host and Network	2016	real	unknown	yes	unknown	Host and Network (2016)
34	CDMC 2016	2016	real	61,730	yes	1	unknown
35	Kharon Kiss et al. (2016)	2016	real	55,733	yes	19	Kharon (2016)
36	CIDDS-001 Ring et al. (2017)	2017	emulated	31,959,267	yes	6	CIDDS (2017)
37	CIDDS-002 Ring et al. (2017)	2017	emulated	16,161,183	yes	5	CIDDS (2017)
38	CAIDA Jonker et al. (2017)	2017	real	huge	no	unknown	CAIDA (2017)
39	CICIDS 2017 Sharafaldin et al. (2018)	2017	emulated	2,830,743	yes	7	CICIDS-2017 (2017)
40	NCCDC	2017	real	unknown	yes	unknown	unknown
41	CICDoS 2017 Jazi et al. (2017)	2017	emulated	32,925	yes	8	CICDDoS-2019 (2019)
42	TRAbID	2017	emulated	huge	yes	2	TRAbID (2017)
43	SUEE 2017	2017	emulated	19,301,217	yes	3	unknown
44	ISOT HTTP Botnet	2017	emulated	huge	yes	9	ISOT HTTP Botnet (2017)
45	PUF	2018	emulated	6,000,000	yes	4	unknown
46	ISOT CID	2018	real	36,938,985	yes	18	ISOT-CID (2018)
47	CICDDoS 2019 Sharafaldin et al. (2019)	2019	emulated	huge	yes	11	CICDDoS-2019 (2019)
48	BoT-IoT Koroniotis et al. (2019)	2019	real	73,360,900	yes	2	BoT-IoT (2019)
49	IoT-23	2020	real	unknown	yes	20	IoT-23 (2020)
50	InSDN	2020	emulated	343,939	yes	7	InSDN (2020)
51	CIRA-CIC-DoHBrw 2020 MontazeriShatoori et al. (2020)	2020	emulated	1,185,286	yes	3	CIRA-CIC-DoHBrw-2020 (2
52	OPCUA	2020	emulated	107,634	yes	3	OPCUA, 2020

4.1 Secondary Datasets

Initially, secondary datasets, derived from existing literature, were utilized. These secondary datasets provided a foundational basis for the research. The details of these datasets including their sizes, sources, etc. are provided.

4.1.1 Sain Malaysian Dataset

Omar Elejila generated ICMPv6 datasets based on the network topology illustrated in Figure 16, serving as the foundational framework for his research. The data generation process extended over a period of 2 hours, capturing network traffic (Figure 17), and resulting in a dataset size of 15.8 MB.

A subset of this traffic, presented as a 2.8 MB Excel sheet, was already present as training and testing datasets in the form of .csv files. These files consist of 11 features outlined in Table 8 (Elejla et al. 2019), each provided with a description. Both Datasets consist same 11 features where the "training.csv" file was utilized as an 80:20 train-test split ratio and employed in the experiments. These data sets were pre-processed where an additional column titled" Class" was added. This column categorizes the network traffic as either" NORMAL" or "ATTACK," providing on the whole classification aspect of the datasets (Elejla et al. 2019).

Table 10: Feature list of Sain Malaysian University datasets (Elejla et al. 2019)

Serial no	Feature	Description	Condition of Threshold
1	ICMPv6Type	Sent packets in the traffic flow	Originally in the flow as a key
2	Packet Number	Number of sent packets in the traffic flow	Counting the number of packets

3	Transferr ed Bytes	Number of bytes sent from the source to the destination	Summation of Packet length
4	Duration	Time length of the traffic flow	Last packet time – first packet time
5	Ratio	The ratio of bytes transferring during the traffic flow duration	Transferred bytes/duration
6	Length_SD	The variation in the length of traffic flow packets	The standard deviation of packet flow Length
7	Flowlabel_SD	The variation in the traffic flow label of packets	The standard deviation of the packet flow label
8	HopLimit_SD	The variation in the hop limits of traffic flow	The standard deviation of the hop limits
9	TrafficClass_SD	The variation in the traffic Class of packet flow	The standard deviation of the traffic Class of the packet
10	NextHeader_SD	The variation in the Next header of the traffic packet flow	The standard deviation of the Next header packet
11	Payloadlength_SD	The variation in payload length of traffic packet flow	The standard deviation of the packet payload length

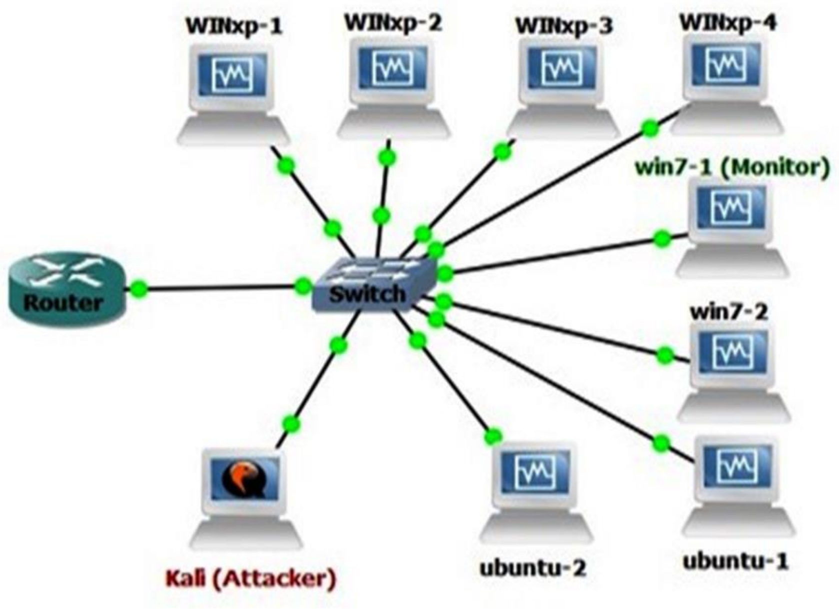
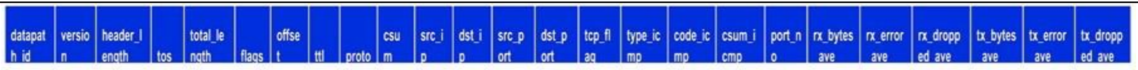


Figure 17: Omar Elejila's Network Topology (Elejla et al. 2019)

4.1.2 Mendeley Dataset

These datasets comprise IPv4 data generated and utilized by Housman from Universitas Muhammadiyah Malang for research purposes. The data encompasses DDoS attacks within SDN, encompassing ICMP, TCP, and UDP flood incidents. The attacks were simulated in the Mininet Emulator, employing Scapy and TCP Reply. (Scapy details are briefed in Chapter 1 section 1.1.2). The resulting traffic was captured as .pcap file format, amounting to a size of 34.7 MB, named TRAIN-DATA .pcap. This file was generated through the packet generation process utilizing the Scapy library. Each malicious packet contains a randomly generated IP source address, targeting H4. The RYU controller was extended to possess the capability of storing attack information in a .csv file, with a size of up to 37.9 MB, named TRAIN-DATA.csv. This process was replicated for TESTDATA. However, only the TRAIN-DATA.csv file was employed in the experiment as both files share the same 25 features that are provided in Table 9 (Housman Oxicusa Gugi 2020) in tabular form, with an 80:20 train-test split ratio. These data sets were also pre-processed and a new column" label" was added to categorize traffic into DDOS-ICMP, DDOS-TCP, DDOS-UDP, NORMAL-ICMP, NORMAL-TCP, and NORMAL-UDP (Housman Oxicusa Gugi 2020).

Table 11: 25 features from the Mendeley dataset (Housman Oxicusa Gugi 2020)



datapath id	version	header_length	tos	total_length	flags	offset	ttl	proto	csum	src_ip	dst_ip	src_port	dst_port	tcp_flag	type_icmp	code_icmp	csum_icmp	port_no	rx_bytes ave	rx_error ave	rx_dropped ave	tx_bytes ave	tx_error ave	tx_dropped ave

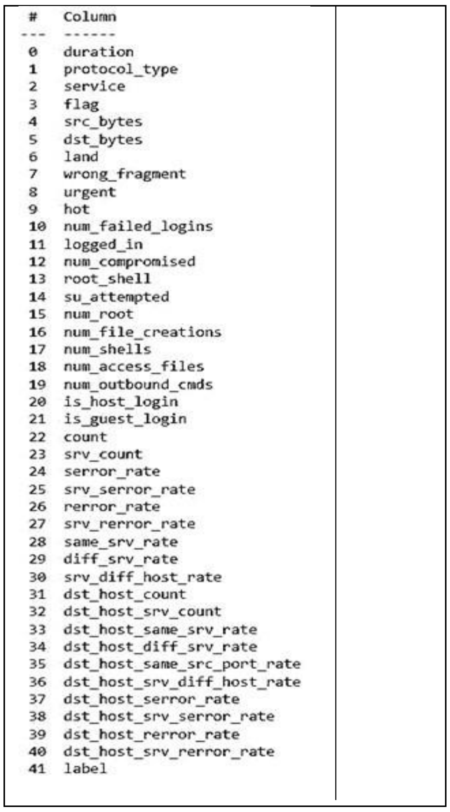
4.1.3 NSL-KDD (Benchmark dataset)

According to Mahbod Tavallaee, the KDD dataset is essentially a compilation of data gathered during the DARPA’98 Intrusion Detection System (IDS) evaluation program. The DARPA’98 dataset comprises around a compressed of 4 GB of raw in binary TCP dump, reflecting network traffic of 7 weeks. This extensive dataset can be transformed into about 5 million connection records, each containing approximately 100 bytes of information. The testing phase of DARPA’98 spans two weeks and generates approximately 2 million connection records. The KDD training dataset contains nearly 4,900,000 vectors that are based on a single connection and contain 40 attributes and is categorized as normal or an attack. The NSL-KDD dataset are improvised dataset over the original KDD dataset (DARPA July 2000). The training set is free from redundant and duplicate records, encompassing files in both .txt and .arff formats. Specifically, the KDDTrain+.txt, sized at 18.2 MB, and KDDTest+, with a size of 3.28 MB, were employed in our experiments. The raw traffic data, initially captured, was transformed into a .txt format, featuring 40 features provided in Table 10 (Tavallaee et al. 2009). It was processed where the data points were in binary and an additional label was added categorizing the type of attacks. The traffic contains multiple attack types. This was again updated by changing attack labels to their respective attack class while experimenting. These NSL-KDD data sets were used for Model 1 and Model 2. Table 10 provides the list of features (Tavallaee et al. 2009).

4.2 Primary Datasets

Subsequently, primary datasets were generated specifically for this Proposed research. The details including their sizes, sources, and the methods used for their generation, are explained. A network diagram is provided to illustrate the data generation process, and further validation is explained based on the feature with respect to one of the secondary datasets.

Table 12: 40 features from NSL-KDD dataset (Tavallaee et al. 2009)



```
#   Column
--- ------
0   duration
1   protocol_type
2   service
3   flag
4   src_bytes
5   dst_bytes
6   land
7   wrong_fragment
8   urgent
9   hot
10  num_failed_logins
11  logged_in
12  num_compromised
13  root_shell
14  su_attempted
15  num_root
16  num_file_creations
17  num_shells
18  num_access_files
19  num_outbound_cmds
20  is_host_login
21  is_guest_login
22  count
23  srv_count
24  serror_rate
25  srv_serror_rate
26  rerror_rate
27  srv_rerror_rate
28  same_srv_rate
29  diff_srv_rate
30  srv_diff_host_rate
31  dst_host_count
32  dst_host_srv_count
33  dst_host_same_srv_rate
34  dst_host_diff_srv_rate
35  dst_host_same_src_port_rate
36  dst_host_srv_diff_host_rate
37  dst_host_serror_rate
38  dst_host_srv_serror_rate
39  dst_host_rerror_rate
40  dst_host_srv_rerror_rate
41  label
```

4.2.1 Collection of Primary Dataset 1

Generation of Primary Dataset-1 was produced on a single machine equipped with an Intel i7 11th generation 2.30 GHz processor, 64 GB RAM, and a 2 TB hard disk. Within a VMware environment, four virtual machines were set up, comprising one Linux machine (LVPC), two Windows machines (WVPC-1 and WVPC-2), and one Windows server (WVS). These virtual machines were interconnected through VMware network adapters on a single NIC card of the host machine with IP version 6 addresses. To generate traffic in the instance of an attack, an ICMPv6 DDoS attack was launched using a Scapy script. The attack is launched from two virtual machines (LVPC and WVPC-1), targeting the Windows server (WVS). The ensuing network traffic, encompassing both normal activity and the ICMPv6 attack, was captured on the Windows server (WVS) using Wireshark, resulting in a dataset size of 18.3 MB

(60,000 bytes/sec). Subsequently, the captured traffic was transformed into an Excel sheet as scdtsets.csv of size 58.2 MB. The proposed Model was employed on this dataset.

4.2.2 Collection of Primary Dataset 2

Dataset-2 was generated through the implementation of a straightforward network design, featuring a Cisco 2901 router, a Cisco 3560 switch, and four Windows systems. Within the network, three Linux operating systems (LPC-0, LPC-1, LPC-2) were installed using VMware, alongside a Windows server system (WVS). Physical connectivity was established through the com 4 port, with individual network adapters in VMware tailored to respective individual systems that have individual NIC cards, configured within the University of Staffordshire Lab environment using IP version 6 addresses. Configuration of the router and switch was carried out using PuTTY, ensuring seamless network traffic among all devices. To assess network behaviour, both under normal conditions and during an ICMPv6 attack, Wireshark was employed on the WVS system to capture the traffic running the Scapy script from LPC-1 and LPC-2. In the beginning, normal traffic was captured for a while and later attack traffic for a duration of 4 hrs 45 min approximately accumulating to 5.12 GB was captured (500,000 bytes/sec). This traffic was subsequently transformed into an Excel sheet with a size of 183 MB as a sample dataset with the file name Labdataset.csv. The proposed Model was employed on this dataset.

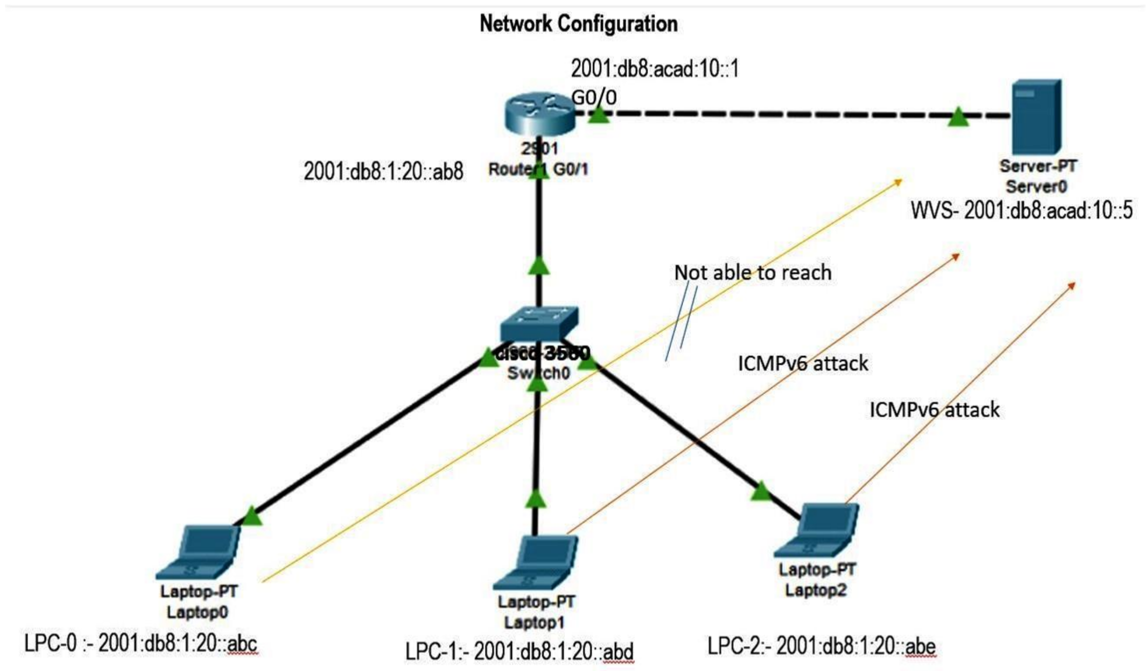


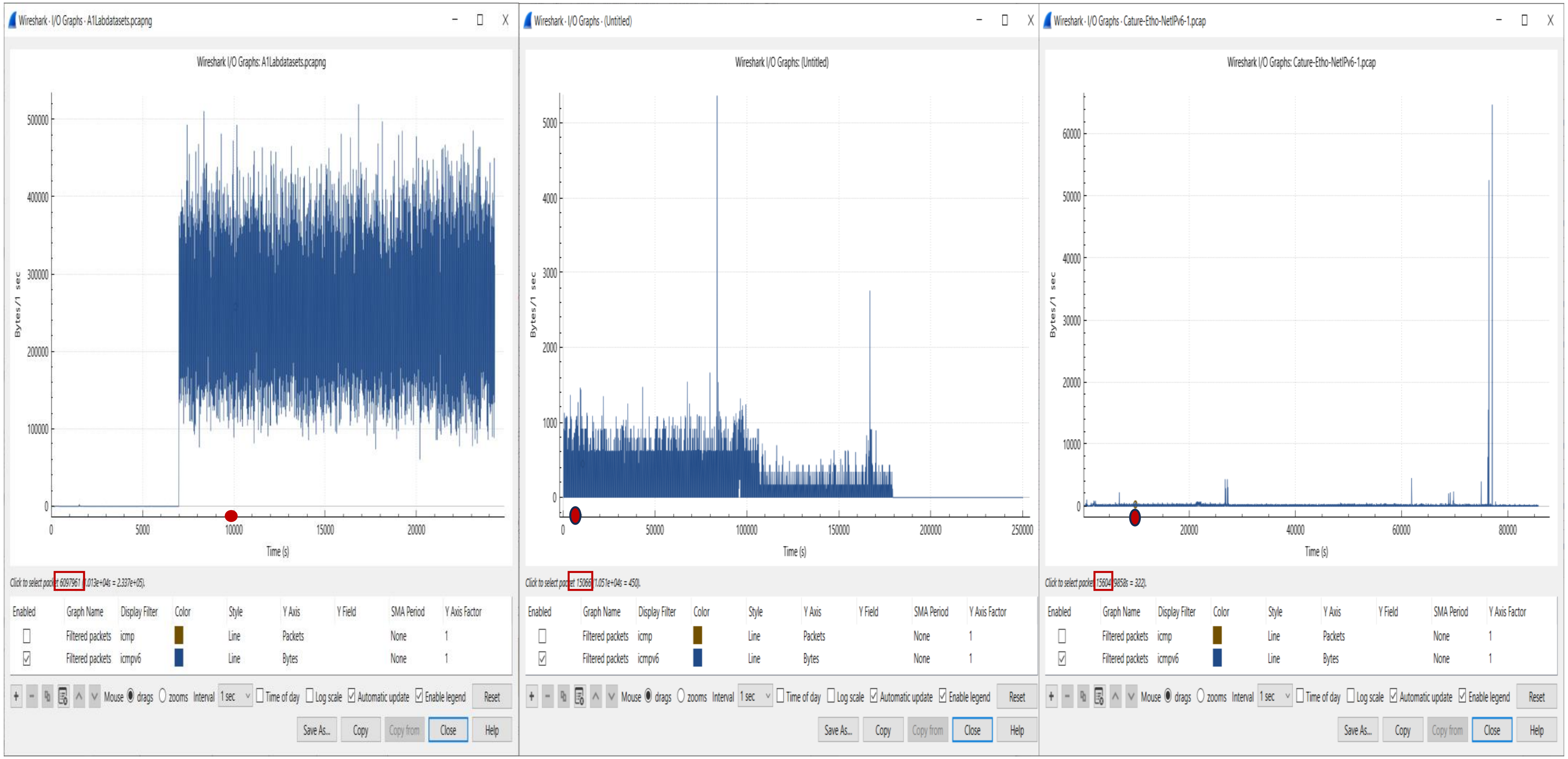
Figure 18: DDoS attack Scenario in University Lab.

Figure 18 depicts the network architecture utilized to simulate a scenario for launching a DDoS attack and capturing the resulting traffic to generate datasets. The router is configured with the IPv6 address 2001:db8:acad:10::1 on interface Gigabit 0/0 (G0/0), which connects to a Windows Server assigned the address 2001:db8:acad:10::5. Similarly, the other interface of the router, G0/1, is assigned the address 2001:db8:1:20::db8 and is linked to a Switch via Fast Ethernet0/0 (Fe0), with additional connections to LPC-0, LPC-1, and LPC-2 on ports Fe1, Fe2, and Fe3 respectively, each assigned an IPv6 address. All devices including the Router, Switch, Server, and nodes (LPC-0, LPC-1, LPC-2), are verified to be connected and communicating with each other using the ping command and their respective assigned IP addresses. Wireshark was installed on the Windows Server to capture both normal and attack traffic packets. The DDoS attack is initiated using a Scapy script from LPC-1 and LPC-2, targeting the Windows Server with a high volume of Echo requests and Echo replies to packets. Periodically, the Windows Server is tested by pinging from LPC-0 to ascertain its availability. If the server is determined to be down due to the attack, evidenced by response timeouts when pinged from LPC-0, the traffic capturing process is halted.

4.2.3 Primary Dataset Validations

Figure 19 illustrates the packet rate validation in both the Network and ICMPv6 attack scenarios, highlighting the traffic flow. In the Lab dataset, the packet flow speed was 6,097,961 packets per 1.013e^{+04}s (10,130 seconds). In the Sain Malaysian dataset, the packet flow speed was 15,066 packets per 1.051e^{+04}s (10,510 seconds). In the LT Dataset 2, the packet flow speed was 15,604 packets per 0.98e^{+58}s (0.98×1058 seconds). It is evident from the graph that all three datasets exhibit a high number of packets, which serves as a key parameter for validating the datasets based on their features. The graph dataset of Lab Dataset 1 consists of predominantly high attack traffic and less normal traffic. In contrast, LT Dataset 2 has a higher volume of normal traffic and less attack traffic. This intentional variation between the datasets aims to evaluate the proposed model's performance under different traffic conditions. These distinct scenarios help determine the model's strength and efficiency in accuracy in identifying DDoS attacks. The ability of the model to maintain consistent detection performance across these varying conditions demonstrates its efficacy as a reliable solution for mitigating DDoS attacks. It was ensured that the traffic content was consistent in terms of traffic volume, speed, features, and values during an attack across both datasets.

To ensure the collected features are sufficient for our model training, we studied feature sets from various sources, including the Malaysian dataset, and selected an appropriate feature set. Figure 20 illustrates the features captured from packet traffic using Wireshark. These features were transformed into an Excel sheet, detailed in section 5.3 of Chapter 5, and underwent necessary processing procedures before being input into the proposed model for deployment. The cleaning and pre-processing of datasets are also explained in the same section.


Lab Data set 1 -Traffic capturing speed
Wireshark · I/O Graphs · A1Labdatasets.pcapng
Wireshark I/O Graphs: A1Labdatasets.pcapng
Bytes/1 sec
Time (s)
Sain Malaysian Data sets capturing speed
Wireshark · I/O Graphs · (Untitled)
Wireshark I/O Graphs: (Untitled)
Bytes/1 sec
Time (s)
LT Data set 2 -Traffic capturing speed
Wireshark · I/O Graphs · Cature-Etho-NetIPv6-1.pcap
Wireshark I/O Graphs: Cature-Etho-NetIPv6-1.pcap
Bytes/1 sec
Time (s)
Save As... Copy Copy from Close Help Reset

Lab Data set 1 -Traffic capturing speed

Click to select packet 6097961 (1.013e+04s = 2.337e+05).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☐	Filtered packets	icmp	🟫	Line	Packets		None	1
☑	Filtered packets	icmpv6	🟦	Line	Bytes		None	1

Mouse [✓ drags] zooms Interval 1 sec ☐ Time of day ☐ Log scale ☑ Automatic update ☑ Enable legend

Sain Malaysian Data sets capturing speed

Click to select packet 15066 (1.051e+04s = 450).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☐	Filtered packets	icmp	🟫	Line	Packets		None	1
☑	Filtered packets	icmpv6	🟦	Line	Bytes		None	1

Mouse [✓ drags] zooms Interval 1 sec ☐ Time of day ☐ Log scale ☑ Automatic update ☑ Enable legend

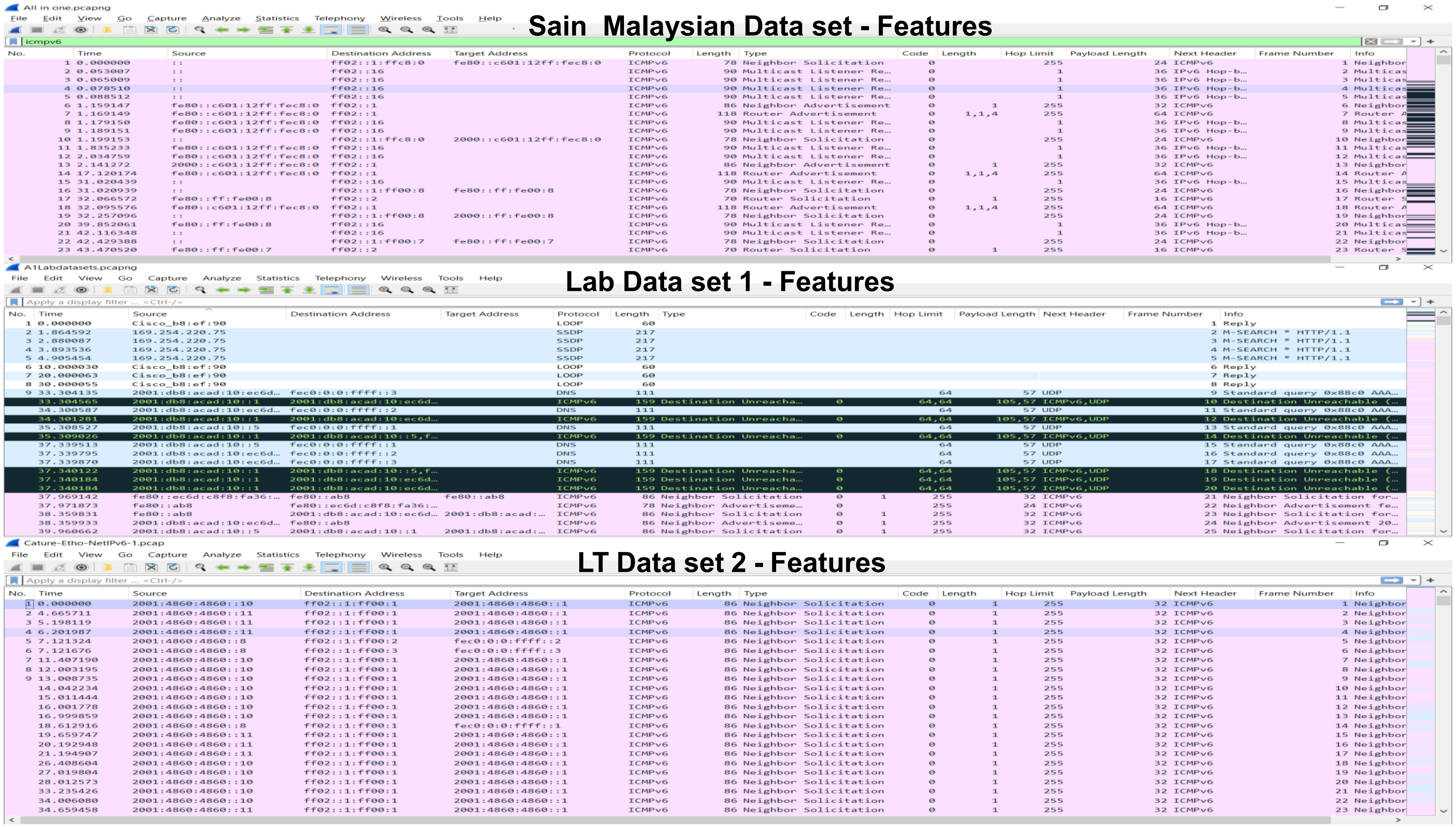
LT Data set 2 -Traffic capturing speed

Click to select packet 15604 (9858s = 322).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☐	Filtered packets	icmp	🟫	Line	Packets		None	1
☑	Filtered packets	icmpv6	🟦	Line	Bytes		None	1

Mouse [✓ drags] zooms Interval 1 sec ☐ Time of day ☐ Log scale ☑ Automatic update ☑ Enable legend

Figure 19: Primary Datasets validation based on packet flow and in the form of a waveform



All in one.pcapng

Sain Malaysian Data set - Features

icmpv6

No.	Time	Source	Destination Address	Target Address	Protocol	Length	Type	Code	Length	Hop Limit	Payload Length	Next Header	Frame Number	Info
1	0.000000	::	ff02::1:ffc8:0	fe80::c601:12ff:fec8:0	ICMPv6	78	Neighbor Solicitation	0		255	24	ICMPv6	1	Neighbor
2	0.053007	::	ff02::16		ICMPv6	90	Multicast Listener Re…	0		1	36	IPv6 Hop-b…	2	Multicas
3	0.065009	::	ff02::16		ICMPv6	90	Multicast Listener Re…	0		1	36	IPv6 Hop-b…	3	Multicas
4	0.078510	::	ff02::16		ICMPv6	90	Multicast Listener Re…	0		1	36	IPv6 Hop-b…	4	Multicas
5	0.088512	::	ff02::16		ICMPv6	90	Multicast Listener Re…	0		1	36	IPv6 Hop-b…	5	Multicas
6	1.159147	fe80::c601:12ff:fec8:0	ff02::1		ICMPv6	86	Neighbor Advertisement	0	1	255	32	ICMPv6	6	Neighbor
7	1.169149	fe80::c601:12ff:fec8:0	ff02::1		ICMPv6	118	Router Advertisement	0	1,1,4	255	64	ICMPv6	7	Router A
8	1.179150	fe80::c601:12ff:fec8:0	ff02::16		ICMPv6	90	Multicast Listener Re…	0		1	36	IPv6 Hop-b…	8	Multicas
9	1.189151	fe80::c601:12ff:fec8:0	ff02::16		ICMPv6	90	Multicast Listener Re…	0		1	36	IPv6 Hop-b…	9	Multicas
10	1.199153	::	ff02::1:ffc8:0	2000::c601:12ff:fec8:0	ICMPv6	78	Neighbor Solicitation	0		255	24	ICMPv6	10	Neighbor
11	1.835233	fe80::c601:12ff:fec8:0	ff02::16		ICMPv6	90	Multicast Listener Re…	0		1	36	IPv6 Hop-b…	11	Multicas
12	2.034759	fe80::c601:12ff:fec8:0	ff02::16		ICMPv6	90	Multicast Listener Re…	0		1	36	IPv6 Hop-b…	12	Multicas
13	2.141272	2000::c601:12ff:fec8:0	ff02::1		ICMPv6	86	Neighbor Advertisement	0	1	255	32	ICMPv6	13	Neighbor
14	17.120174	fe80::c601:12ff:fec8:0	ff02::1		ICMPv6	118	Router Advertisement	0	1,1,4	255	64	ICMPv6	14	Router A
15	31.020439	::	ff02::16		ICMPv6	90	Multicast Listener Re…	0		1	36	IPv6 Hop-b…	15	Multicas
16	31.020939	::	ff02::1:ff00:8	fe80::ff:fe00:8	ICMPv6	78	Neighbor Solicitation	0		255	24	ICMPv6	16	Neighbor
17	32.066572	fe80::ff:fe00:8	ff02::2		ICMPv6	70	Router Solicitation	0	1	255	16	ICMPv6	17	Router S
18	32.095576	fe80::c601:12ff:fec8:0	ff02::1		ICMPv6	118	Router Advertisement	0	1,1,4	255	64	ICMPv6	18	Router A
19	32.257096	::	ff02::1:ff00:8	2000::ff:fe00:8	ICMPv6	78	Neighbor Solicitation	0		255	24	ICMPv6	19	Neighbor
20	39.852061	fe80::ff:fe00:8	ff02::16		ICMPv6	90	Multicast Listener Re…	0		1	36	IPv6 Hop-b…	20	Multicas
21	42.116348	::	ff02::16		ICMPv6	90	Multicast Listener Re…	0		1	36	IPv6 Hop-b…	21	Multicas
22	42.429388	::	ff02::1:ff00:7	fe80::ff:fe00:7	ICMPv6	78	Neighbor Solicitation	0		255	24	ICMPv6	22	Neighbor
23	43.470520	fe80::ff:fe00:7	ff02::2		ICMPv6	70	Router Solicitation	0	1	255	16	ICMPv6	23	Router S

A1Labdatasets.pcapng

Lab Data set 1 - Features

Apply a display filter … <Ctrl-/>

No.	Time	Source	Destination Address	Target Address	Protocol	Length	Type	Code	Length	Hop Limit	Payload Length	Next Header	Frame Number	Info
1	0.000000	Cisco_b8:ef:90			LOOP	60							1	Reply
2	1.864592	169.254.220.75			SSDP	217							2	M-SEARCH * HTTP/1.1
3	2.880087	169.254.220.75			SSDP	217							3	M-SEARCH * HTTP/1.1
4	3.893536	169.254.220.75			SSDP	217							4	M-SEARCH * HTTP/1.1
5	4.905454	169.254.220.75			SSDP	217							5	M-SEARCH * HTTP/1.1
6	10.000030	Cisco_b8:ef:90			LOOP	60							6	Reply
7	20.000063	Cisco_b8:ef:90			LOOP	60							7	Reply
8	30.000055	Cisco_b8:ef:90			LOOP	60							8	Reply
9	33.304135	2001:db8:acad:10:ec6d…	fec0:0:0:ffff::3		DNS	111				64	57	UDP	9	Standard query 0x88c0 AAA…
	33.304565	2001:db8:acad:10::1	2001:db8:acad:10:ec6d…		ICMPv6	159	Destination Unreacha…	0		64,64	105,57	ICMPv6,UDP	10	Destination Unreachable (…
	34.300587	2001:db8:acad:10:ec6d…	fec0:0:0:ffff::2		DNS	111				64	57	UDP	11	Standard query 0x88c0 AAA…
	34.301281	2001:db8:acad:10::1	2001:db8:acad:10:ec6d…		ICMPv6	159	Destination Unreacha…	0		64,64	105,57	ICMPv6,UDP	12	Destination Unreachable (…
	35.308527	2001:db8:acad:10::5	fec0:0:0:ffff::1		DNS	111				64	57	UDP	13	Standard query 0x88c0 AAA…
	35.309026	2001:db8:acad:10::1	2001:db8:acad:10::5,f…		ICMPv6	159	Destination Unreacha…	0		64,64	105,57	ICMPv6,UDP	14	Destination Unreachable (…
	37.339513	2001:db8:acad:10::5	fec0:0:0:ffff::1		DNS	111				64	57	UDP	15	Standard query 0x88c0 AAA…
	37.339795	2001:db8:acad:10:ec6d…	fec0:0:0:ffff::2		DNS	111				64	57	UDP	16	Standard query 0x88c0 AAA…
	37.339870	2001:db8:acad:10:ec6d…	fec0:0:0:ffff::3		DNS	111				64	57	UDP	17	Standard query 0x88c0 AAA…
	37.340122	2001:db8:acad:10::1	2001:db8:acad:10::5,f…		ICMPv6	159	Destination Unreacha…	0		64,64	105,57	ICMPv6,UDP	18	Destination Unreachable (…
	37.340184	2001:db8:acad:10::1	2001:db8:acad:10:ec6d…		ICMPv6	159	Destination Unreacha…	0		64,64	105,57	ICMPv6,UDP	19	Destination Unreachable (…
	37.340184	2001:db8:acad:10::1	2001:db8:acad:10:ec6d…		ICMPv6	159	Destination Unreacha…	0		64,64	105,57	ICMPv6,UDP	20	Destination Unreachable (…
	37.969142	fe80::ec6d:c8f8:fa36:…	fe80::ab8	fe80::ab8	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	21	Neighbor Solicitation for…
	37.971873	fe80::ab8	fe80::ec6d:c8f8:fa36:…		ICMPv6	78	Neighbor Advertiseme…	0		255	24	ICMPv6	22	Neighbor Advertisement fe…
	38.359831	fe80::ab8	2001:db8:acad:10:ec6d…	2001:db8:acad:…	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	23	Neighbor Solicitation for…
	38.359933	2001:db8:acad:10:ec6d…	fe80::ab8		ICMPv6	86	Neighbor Advertiseme…	0	1	255	32	ICMPv6	24	Neighbor Advertisement 20…
	39.960662	2001:db8:acad:10::5	2001:db8:acad:10::1	2001:db8:acad:…	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	25	Neighbor Solicitation for…

Cature-Etho-NetIPv6-1.pcap

LT Data set 2 - Features

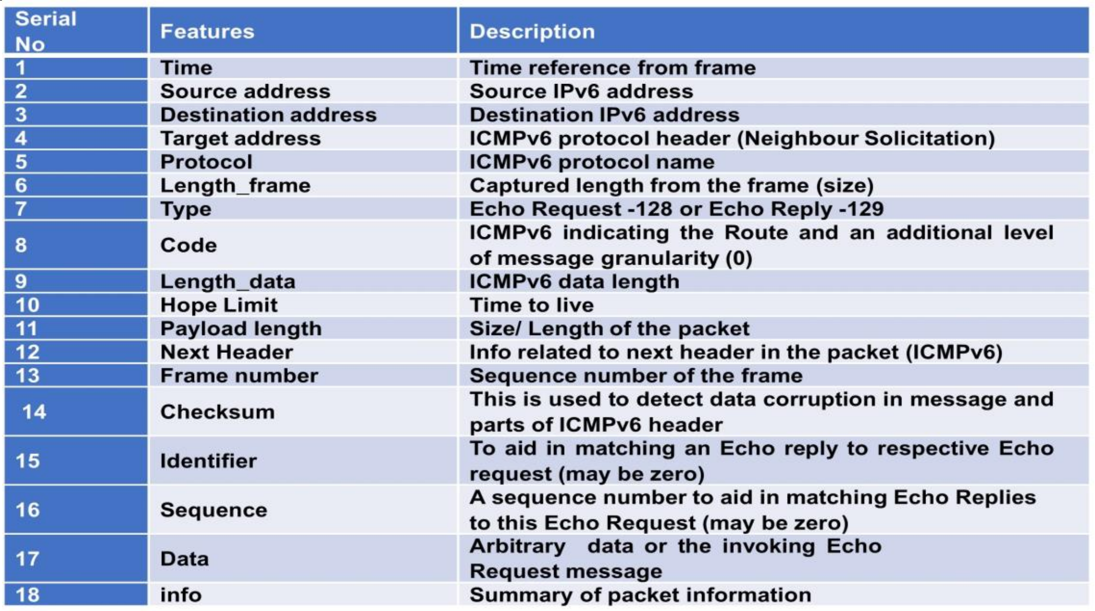
Apply a display filter … <Ctrl-/>

No.	Time	Source	Destination Address	Target Address	Protocol	Length	Type	Code	Length	Hop Limit	Payload Length	Next Header	Frame Number	Info
1	0.000000	2001:4860:4860::10	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	1	Neighbor
2	4.665711	2001:4860:4860::11	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	2	Neighbor
3	5.198119	2001:4860:4860::11	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	3	Neighbor
4	6.201987	2001:4860:4860::11	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	4	Neighbor
5	7.121324	2001:4860:4860::8	ff02::1:ff00:2	fec0:0:0:ffff::2	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	5	Neighbor
6	7.121676	2001:4860:4860::8	ff02::1:ff00:3	fec0:0:0:ffff::3	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	6	Neighbor
7	11.407190	2001:4860:4860::10	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	7	Neighbor
8	12.003195	2001:4860:4860::10	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	8	Neighbor
9	13.008735	2001:4860:4860::10	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	9	Neighbor
	14.042234	2001:4860:4860::10	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	10	Neighbor
	15.011444	2001:4860:4860::10	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	11	Neighbor
	16.001778	2001:4860:4860::10	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	12	Neighbor
	16.999859	2001:4860:4860::10	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	13	Neighbor
	18.612916	2001:4860:4860::8	ff02::1:ff00:1	fec0:0:0:ffff::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	14	Neighbor
	19.659747	2001:4860:4860::11	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	15	Neighbor
	20.192948	2001:4860:4860::11	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	16	Neighbor
	21.194907	2001:4860:4860::11	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	17	Neighbor
	26.408604	2001:4860:4860::10	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	18	Neighbor
	27.019804	2001:4860:4860::10	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	19	Neighbor
	28.012573	2001:4860:4860::10	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	20	Neighbor
	33.235426	2001:4860:4860::10	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	21	Neighbor
	34.006080	2001:4860:4860::10	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	22	Neighbor
	34.659458	2001:4860:4860::11	ff02::1:ff00:1	2001:4860:4860::1	ICMPv6	86	Neighbor Solicitation	0	1	255	32	ICMPv6	23	Neighbor

Figure 20: Primary Datasets validation-based features and values

On keen observation, the data set features are the same and their values are also mostly appropriate meaningful values. However, some features may not have any values when scrolled down that would be taken care of at the time of the preprocessing step.

Table 13: 18 Features Primary Datasets



Serial No	Features	Description
1	Time	Time reference from frame
2	Source address	Source IPv6 address
3	Destination address	Destination IPv6 address
4	Target address	ICMPv6 protocol header (Neighbour Solicitation)
5	Protocol	ICMPv6 protocol name
6	Length_frame	Captured length from the frame (size)
7	Type	Echo Request -128 or Echo Reply -129
8	Code	ICMPv6 indicating the Route and an additional level of message granularity (0)
9	Length_data	ICMPv6 data length
10	Hope Limit	Time to live
11	Payload length	Size/ Length of the packet
12	Next Header	Info related to next header in the packet (ICMPv6)
13	Frame number	Sequence number of the frame
14	Checksum	This is used to detect data corruption in message and parts of ICMPv6 header
15	Identifier	To aid in matching an Echo reply to respective Echo request (may be zero)
16	Sequence	A sequence number to aid in matching Echo Replies to this Echo Request (may be zero)
17	Data	Arbitrary data or the invoking Echo Request message
18	info	Summary of packet information

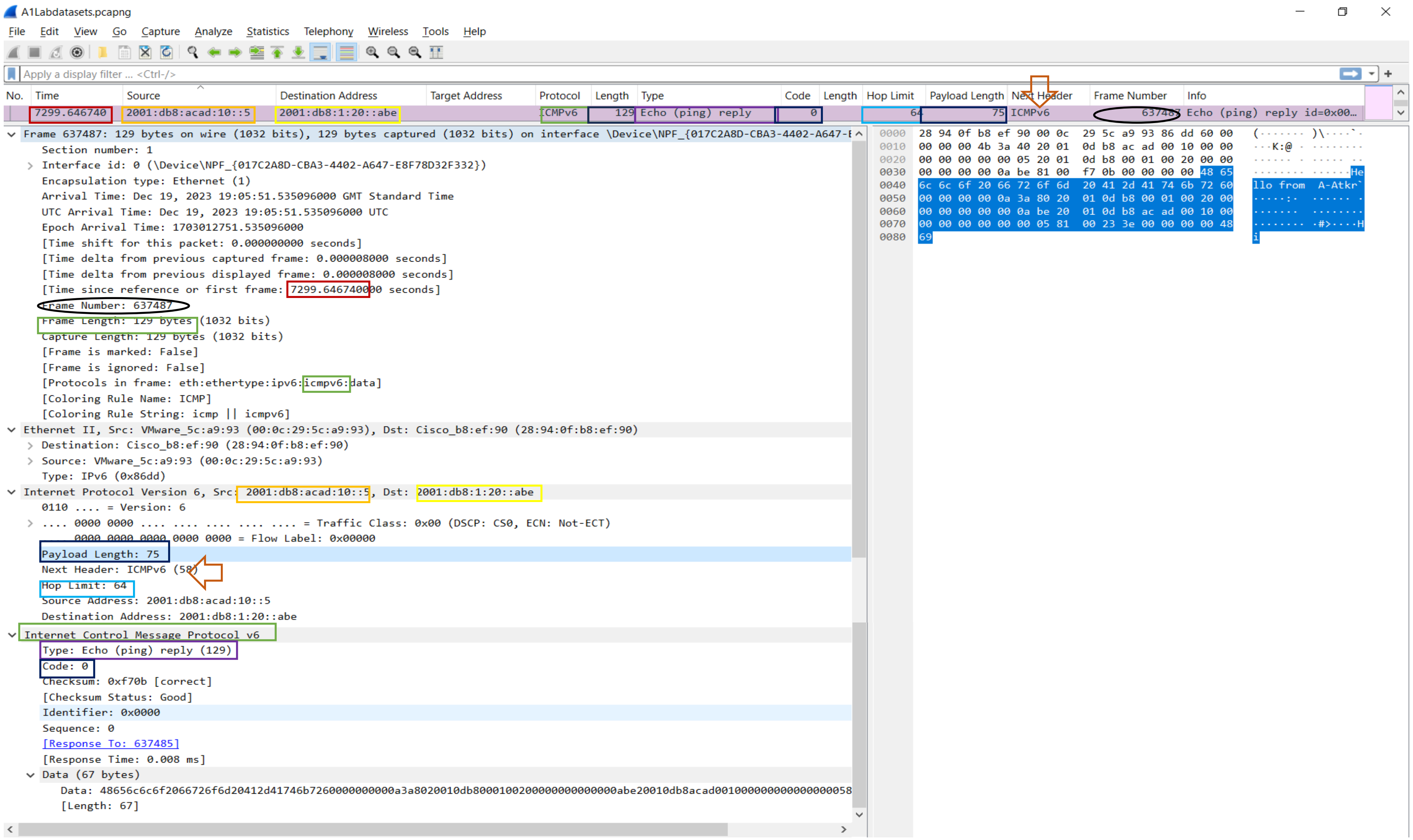
As mentioned in the First Chapter section 1.1.1 the Features listed in Table 11 are from the first 3 layers of the OSI Model Physical layer, Data link layer and Network layer. However, the author Omar Elejla from Sain Malaysian University has used only 11 features confined to traffic (packet) flow. The validation of the datasets is performed based on the features listed in Table 12, highlighting the commonality in the description between the two datasets.

Table 14: Common Features

Common Features of Sain Malaysian Dataset and Primary Datasets 1 and 2				
Serial No.	**Sain Malaysian dataset Features**	**Description**	**Primary Dataset Features**	**Description**

1	**ICMPv6 Type**	**Type of sent packet**	**Type**	**Type code of Echo request and Echo reply**
2	Packet Number	Number of sent packets within the flow	Frame Number	Sequence number of the frame/packet
3	Transferred Bytes	Number of bytes sent from the source to destination	Sequence	Number to aid in matching Echo replies to Echo request
4	Duration	Time length of flow	Time	Time reference from frame/packet
5	Ratio	Ratio of bytes transferring during the flow	Identifier	To aid in matching echo reply to respective Echo request
6	Length	The variation in the lengths of flow's packets	Length frame/packet	Capture length from the frame/packet size
7	Flow label	The variation in the Flow Label of flow's packets	Data	Arbitrary data
8	Hope Limit	The variation in the Hope Limit	Hope Limit	Time to live
9	Traffic Class	The variation in the traffic class of flow's	Checksum	To detect data corruption in messages and parts of the header
10	Next Header	The variation in the next header	Next Header	Info related to next header in packet
11	Payload Length	The variation in the next payload length	Payload Length	Size/length of the packet.

In this research experiment, generated data set features are taken for the ICMPv6 header fields and followed the same aspect of distinguishing the packets as Normal and Attack under Class Column (Label). The “info” feature is the summary of the packet. Figure 21 depicts a sample of an ICMPv6 packet, correlating with the features listed in Table 11. It shows that the highlighted parameters are mostly consistent across all three datasets, with appropriate values. Some of the header fields, like ICMPv6 header fields, including source and destination addresses, payload length, protocol type, and embedded message, etc., are explained below:


A1Labdatasets.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time         Source              Destination Address  Target Address  Protocol  Length  Type                 Code  Length  Hop Limit  Payload Length  Next Header  Frame Number  Info
     7299.646740  2001:db8:acad:10::5  2001:db8:1:20::abe                   ICMPv6       129  Echo (ping) reply       0                64              75  ICMPv6              637487  Echo (ping) reply id=0x00…
```

```
∨ Frame 637487: 129 bytes on wire (1032 bits), 129 bytes captured (1032 bits) on interface \Device\NPF_{017C2A8D-CBA3-4402-A647-E
    Section number: 1
  > Interface id: 0 (\Device\NPF_{017C2A8D-CBA3-4402-A647-E8F78D32F332})
    Encapsulation type: Ethernet (1)
    Arrival Time: Dec 19, 2023 19:05:51.535096000 GMT Standard Time
    UTC Arrival Time: Dec 19, 2023 19:05:51.535096000 UTC
    Epoch Arrival Time: 1703012751.535096000
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.000008000 seconds]
    [Time delta from previous displayed frame: 0.000008000 seconds]
    [Time since reference or first frame: 7299.646740000 seconds]
    Frame Number: 637487
    Frame Length: 129 bytes (1032 bits)
    Capture Length: 129 bytes (1032 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ipv6:icmpv6:data]
    [Coloring Rule Name: ICMP]
    [Coloring Rule String: icmp || icmpv6]
∨ Ethernet II, Src: VMware_5c:a9:93 (00:0c:29:5c:a9:93), Dst: Cisco_b8:ef:90 (28:94:0f:b8:ef:90)
  > Destination: Cisco_b8:ef:90 (28:94:0f:b8:ef:90)
  > Source: VMware_5c:a9:93 (00:0c:29:5c:a9:93)
    Type: IPv6 (0x86dd)
∨ Internet Protocol Version 6, Src: 2001:db8:acad:10::5, Dst: 2001:db8:1:20::abe
    0110 .... = Version: 6
  > .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
    Payload Length: 75
    Next Header: ICMPv6 (58)
    Hop Limit: 64
    Source Address: 2001:db8:acad:10::5
    Destination Address: 2001:db8:1:20::abe
∨ Internet Control Message Protocol v6
    Type: Echo (ping) reply (129)
    Code: 0
    Checksum: 0xf70b [correct]
    [Checksum Status: Good]
    Identifier: 0x0000
    Sequence: 0
    [Response To: 637485]
    [Response Time: 0.008 ms]
  ∨ Data (67 bytes)
      Data: 48656c6c6f2066726f6d20412d41746b7260000000000a3a8020010db8000100200000000000000abe20010db8acad001000000000000000058
      [Length: 67]
```

```
0000  28 94 0f b8 ef 90 00 0c  29 5c a9 93 86 dd 60 00   (·······)\····`·
0010  00 00 00 4b 3a 40 20 01  0d b8 ac ad 00 10 00 00   ···K:@ ·········
0020  00 00 00 00 00 05 20 01  0d b8 00 01 00 20 00 00   ······ ······ ··
0030  00 00 00 00 0a be 81 00  f7 0b 00 00 00 00 48 65   ··············He
0040  6c 6c 6f 20 66 72 6f 6d  20 41 2d 41 74 6b 72 60   llo from  A-Atkr`
0050  00 00 00 00 0a 3a 80 20  01 0d b8 00 01 00 20 00   ······:·  ······ ·
0060  00 00 00 00 00 0a be 20  01 0d b8 ac ad 00 10 00   ·······  ········
0070  00 00 00 00 00 00 05 81  00 23 3e 00 00 00 00 48   ········ ·#>····H
0080  69                                                 i
```

Figure 21: ICMPv6 packet details correlating the features listed in Table 11

1) **Frame Number:**

- **Value**: 637487
- **Explanation**: This is the sequential number of the captured packet in the pcap file.

2) **Timestamp:**

- **Value**: 7299.646740
- **Explanation**: Time in seconds since the first captured packet, helping determine when the packet was seen.

3) **Frame Length**

- **Value**: 129 bytes
- **Explanation**: Total length of the packet captured on the wire, including headers and payload.

4) **Protocol**

- **Value**: ICMPv6
- **Explanation**: Identifies the protocol in use, which is **Internet Control Message Protocol for IPv6**.

5) **IPv6 Addresses**

- **Source**: 2001:db8:acad:10::5
- **Destination**: 2001:db8:1:20::abe
- **Explanation**: These are the source and destination IPv6 addresses of the devices involved in communication.

6) **Payload Length**

- **Value**: 75
- **Explanation**: Indicates the length of the payload in the IPv6 packet (i.e., data excluding the IPv6 header).

7) **Next Header**

- **Value**: ICMPv6 (58)
- **Explanation**: Specifies the type of header immediately following the IPv6 header (here, ICMPv6).

8) **Hop Limit**

- **Value**: 64
- **Explanation**: Works like the TTL (Time to Live) in IPv4, determining how many hops the packet can take before being discarded.

9) ICMPv6 Details

- **Type**: 129 (Echo Reply)
- **Code**: 0
- **Checksum**: Correct
- **ID**: 0x00f0
- **Sequence Number**: 2
- **Explanation**: This is an Echo Reply message in ICMPv6, often used in ping operations to test connectivity.

10) Data Content

- **Payload**: ASCII-encoded string visible on the right: "Hello from A-Atkr"
- **Explanation**: The actual message data is sent back in the Echo Reply packet.

Feature Engineering:

From the aim in Chapter 1, the feature engineering from the first two layers are processed and mainly included along with the above-mentioned features in phase 4 experiments that are given in the following:

Table 15: Feature Engineering

Serial No	Feature	Significance in DDoS Detection
1	Error Type Count	Detects malformed packets, indicative of protocol abuse/flooding
2	Inter-Frame Delay	Identifies high-rate/automated traffic
3	Packet Rate	Measures traffic intensity; high rates suggest DDoS
4	Total Frame Size	Flags volume-based attacks through payload size patterns
5	LLC Indicator	Detects protocol misuse or spoofing

Chapter 4 provides a comprehensive overview of the datasets used in this research. It explains how each dataset was generated, detailing its structure, size, key features, and relevance to the study. The chapter also illustrates the nature of the captured traffic and highlights the significance of specific feature values in representing real-world network behaviour.

5 EXPERIMENTS

5.1 Introduction:

This chapter delves into the experiments conducted and the results obtained. Building on the discussion of the DDoS attack mechanism in Section 4.2.2 of the previous chapter. It outlines the metrics employed to measure the results, comparing them based on the datasets used. The chapter presents the promising experimental results in graphical form and validates them in tabular form, concluding with a summary.

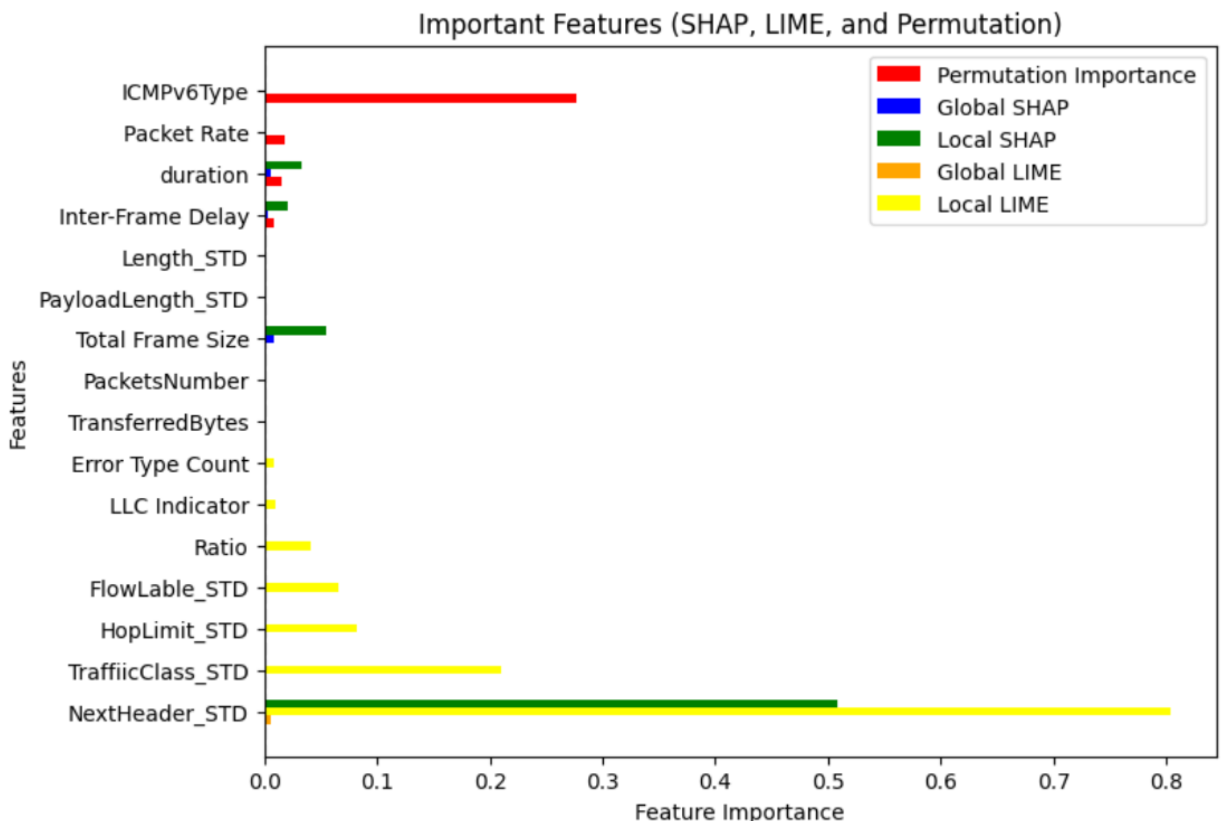
Before discussing the experiments, let's try to summarise the research gap in the 1.2 section, that despite the development of intelligent Intrusion Detection Systems (IDS), the increasing intensity and sophistication of ICMPv6-based DDoS attacks exploiting protocol vulnerabilities, such as fragment manipulation and header evasion, continue to elude detection. Existing systems often fail to identify these attacks due to their lack of identifiable signatures and the complexity of ICMPv6 traffic. With the widespread adoption of IPv6 and the growing attack surface across smart technologies, there is a critical research gap in effectively detecting such covert threats. This study addresses this gap by exploring advanced AI techniques, particularly deep neural networks, to enhance the detection and prediction of ICMPv6 DDoS attacks with the aim of achieving near-perfect accuracy.

5.2. A summary of the dataset's background and the context scenarios:

In this section, the basic information of each dataset and the feature contribution towards the model are provided. Further, the model performance is briefed with respect to Table 16, focusing on the metrics used.

Sain Malaysian datasets: This is an ICMPv6 network traffic dataset using a predefined network topology over a 2-hour capture period, resulting in a 15.8 MB dataset. A 2.8 MB subset, formatted as .csv files, containing 11 features. The data was pre-processed to include a "Class" column labelling each entry as either "NORMAL" or "ATTACK,". Further detailed information about the dataset is given in Chapter 4, section 4.1.1. Further, it has IPv6-based headers. Feature engineering from the first two layers (physical and datalink layers), windowing, ADASYN, SHAP, and LIME are applied.

The figure 22 provides brief information about the contribution of features that impact the model's performance



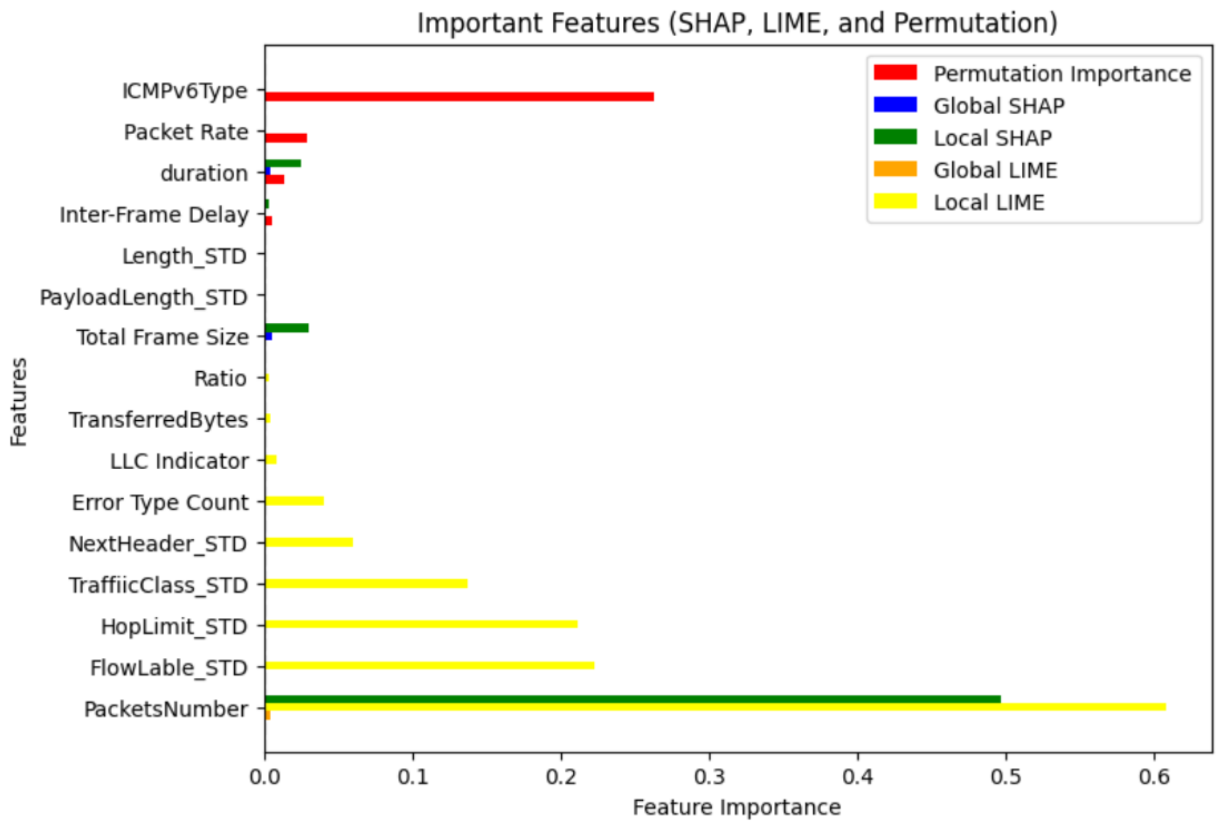


Figure 22 Sain Malaysian Feature contribution.

Top Plot Highlights CNN_LSTM:

- NextHeader_STD is highly important in both Local SHAP and Local LIME.
- TrafficClass_STD, HopLimit_STD, and FlowLabel_STD show moderate importance in LIME evaluations.
- ICMPv6Type has the highest Permutation Importance, indicating strong global relevance.
- Packet Rate and duration show minimal but non-zero importance in

Bottom Plot Highlights RNN_GRU:

- ICMPv6Type stands out in Permutation Importance.
- PacketsNumber, FlowLable_STD, and HopLimit_STD are dominant in Local SHAP and Local LIME.
- TrafficClass_STD and NextHeader_STD also show consistent importance in LIME evaluations.

Significant features from both models:

- ICMPv6Type is a consistently important feature across both models, particularly for Permutation Importance.
- NextHeader_STD, PacketsNumber, and TrafficClass_STD are significant in SHAP and LIME, especially for local interpretations.
- FlowLable_STD and HopLimit_STD also contribute substantially, especially in LIME-based importance.

These features are likely crucial for the model's predictive performance, particularly in distinguishing specific patterns (possibly anomalies or attack types) in the dataset.

Mendeley datasets: This dataset is generated on SDN environment, including. These attacks were simulated using the Mininet emulator with

Scapy and TCP Reply, generating traffic stored in a 34.7 MB .pcap file. The RYU controller recorded the traffic in a 37.9 MB .csv file, containing 25 features based on IPv4. The data was pre-processed to include a "label" column labeling each entry as either "NORMAL_ICMP" or "DDOS_ICMP". Further detailed information about the dataset is given in Chapter 4, section 4.1.2. Feature engineering from the first two layers (physical and datalink layers), windowing, ADASYN, SHAP, and LIME are applied.

The figure 23 provides brief information about the contribution of features that impact the model's performance

Top Plot Highlights CNN_LSTM:

- Local SHAP identifies ttl, src_port, and csum_icmp as significant.
- Permutation Importance highlights ttl, src_port, and dst_port as important.
- LIME (Local) gives the highest importance to features such as Normal, version, header_length, and tos.

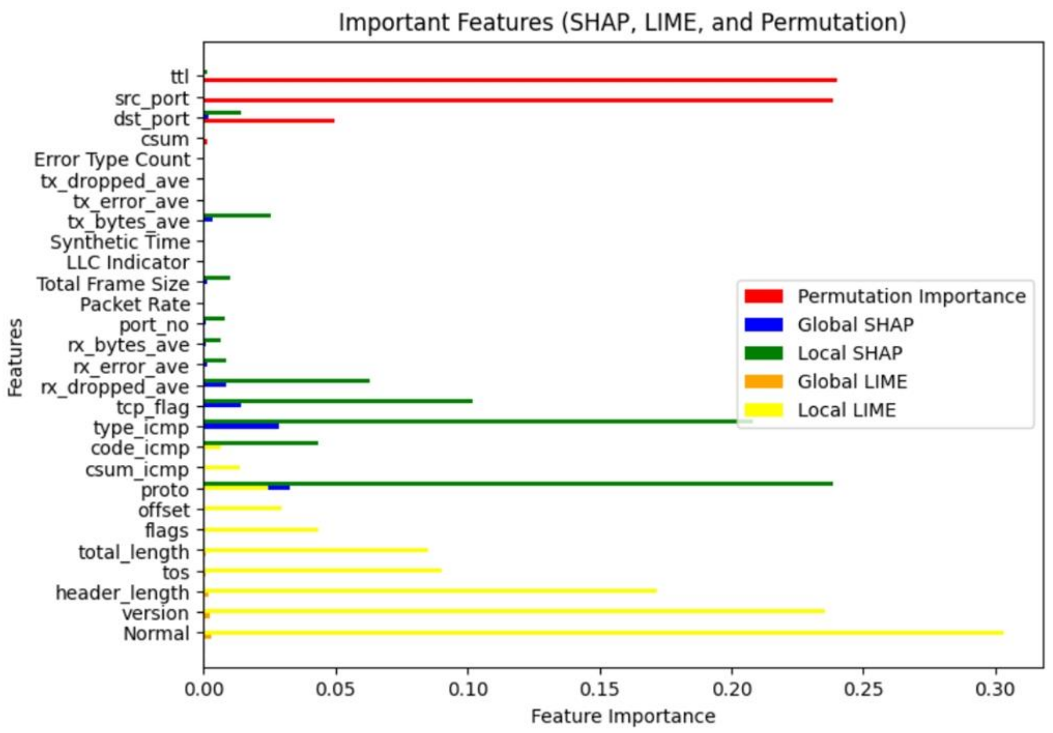
Bottom Plot Highlights RNN_GRU:

- Local LIME again emphasizes Normal, version, header_length, and tos as top contributors.
- Local SHAP highlights rx_bytes_ave, type_icmp, and csum_icmp.
- Global SHAP assigns modest importance to code_icmp and proto.

Significant features from both models:

- Normal, version, header_length, and tos are the most dominant in Local LIME, showing very high importance (> 0.4).
- ttl, src_port, type_icmp, and csum_icmp appear significant in Local SHAP and/or Permutation Importance.
- rx_bytes_ave shows some importance in both SHAP and LIME.

These features are likely critical for the model's decision-making and potentially useful for further feature selection or interpretation in intrusion detection or anomaly classification tasks.



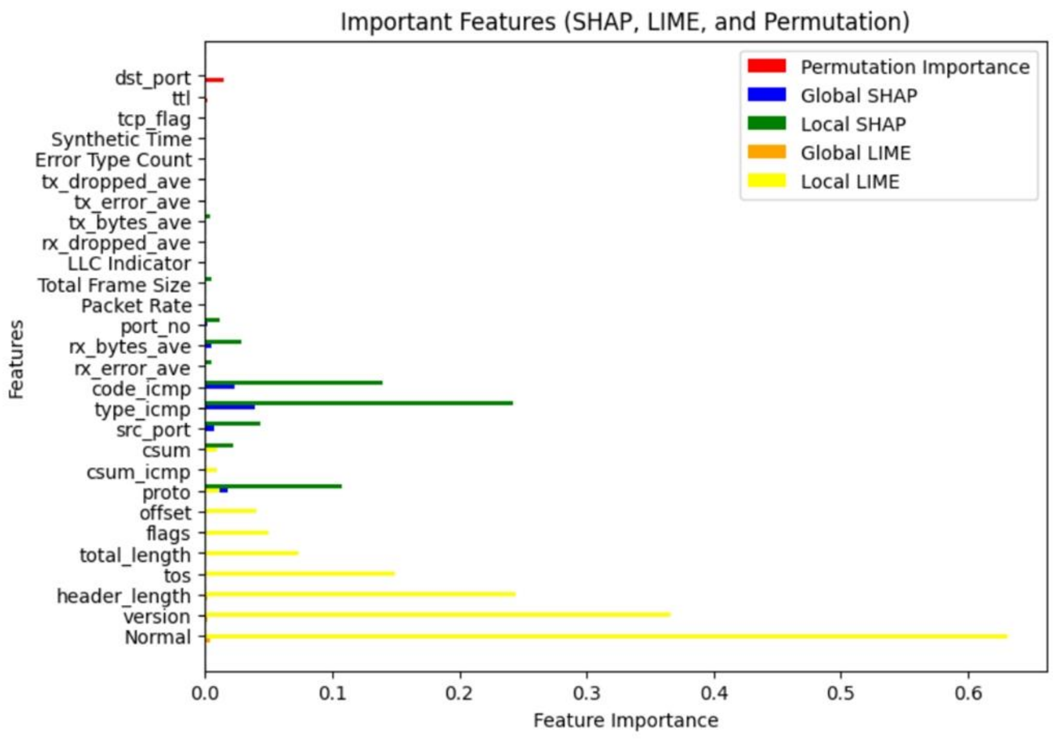


Figure 23: Mendeley Feature contribution.

NSL_KDD datasets: The KDD dataset, derived from the DARPA'98 IDS evaluation program, includes millions of connection records labelled as normal or attack, with 40 features based on IPv4. Its improved version, the NSL-KDD dataset, removes duplicates and redundancies and is available in .txt and .arff formats. In this study, the KDDTrain+ (18.2 MB) and

KDDTest+ (3.28 MB) files were generated. The data was pre-processed to include a "label" column labelling each entry as “Attack” and “Normal”. Further detailed information about the dataset is given in Chapter 4, section 4.1.3. Windowing, ADASYN, SHAP, and LIME are applied, however, feature engineering is not applied. Figure 24 provides brief information about the contribution of features that impact the model's performance.

Top Plot Highlights CNN_LSTM:

- **High contribution across all methods**: flag_S0, count, service_http, logged_in, dst_host_same_srv_rate, srv_count, service_private
- **Consistent LIME and SHAP importance**: dst_host_srv_count, dst_host_count, error_rate, srv_error_rate

Bottom Plot Highlights RNN_GRU:

- **Dominant Features**: count, dst_host_srv_count, srv_count, dst_host_count, service_http, logged_in, src_bytes, flag_S0 Identified prominently by SHAP, LIME, and Permutation)

Significant features from both models: The most significant features contributing to model performance across interpretability methods are:

- count
- dst_host_srv_count
- srv_count
- service_http
- logged_in
- flag_S0
- src_bytes

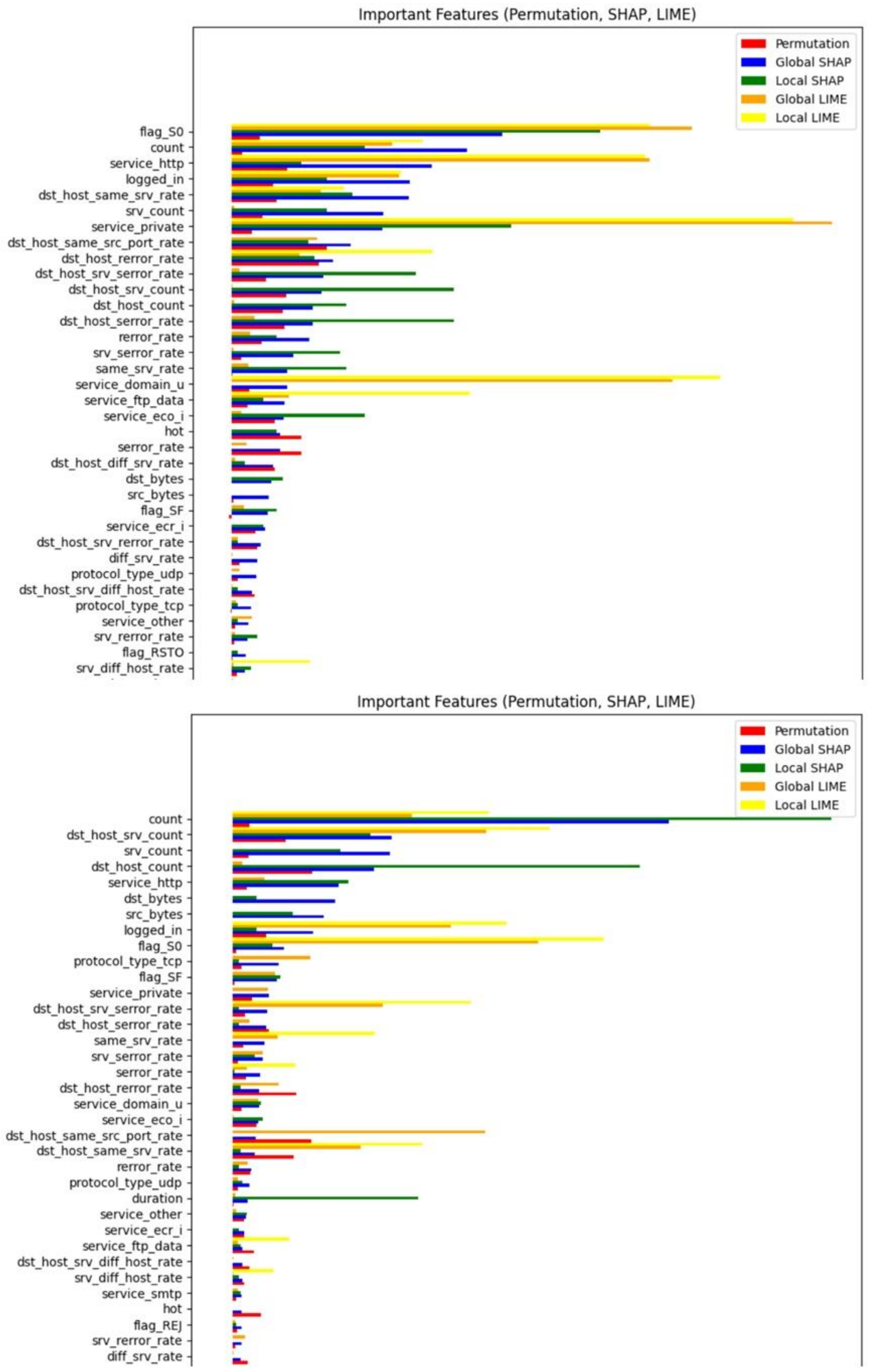


Figure 24 NSL-KDD Feature contribution.

These features play a critical role in intrusion detection and are consistent across both levels for interpretability techniques.

LTVM datasets: This dataset was generated in a VMware environment using IPv6, with ICMPv6 DDoS attacks launched from two virtual machines targeting a Windows server. Network traffic was captured using Wireshark and converted into an Excel file of size 58.2 MB, containing 14 features. The data was pre-processed to include a "Class" column labelling each entry as “Attack” and “Normal”. Further detailed information about the dataset is given in Chapter 4, section 4.2.1. Feature engineering from the first two layers (physical and datalink layers), windowing, ADASYN, SHAP, and LIME are applied.

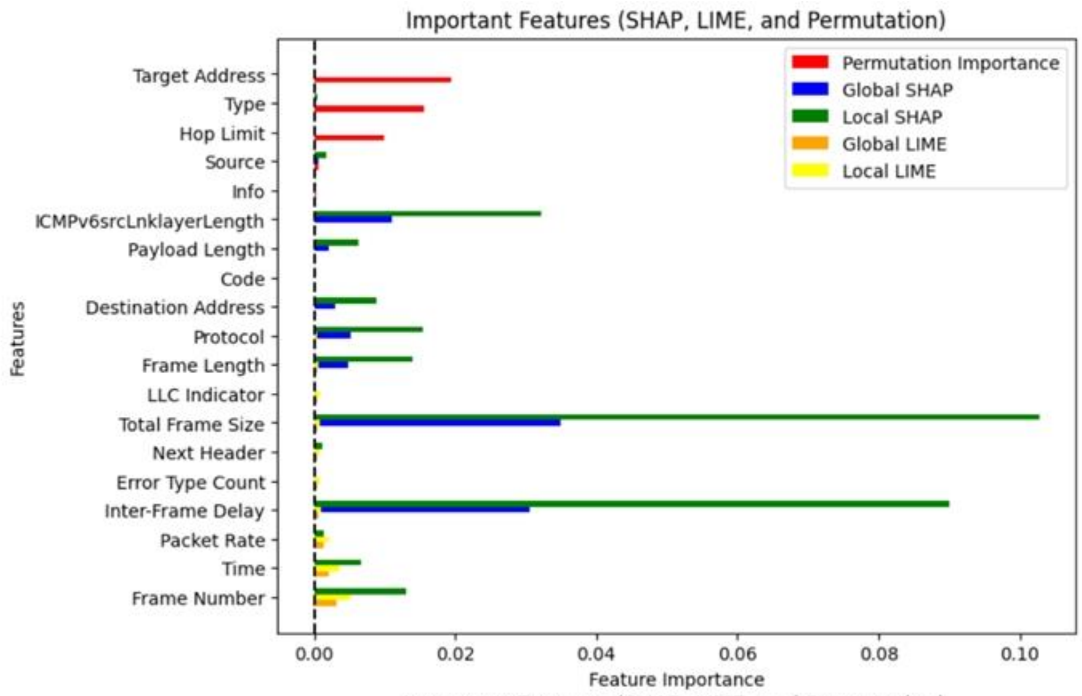
Figure 25 provides brief information about the contribution of features that impact the model's performance.

Top Plot Highlights CNN_LSTM:

- Total Frame Size — very high local SHAP **i**mportance
- Next Header — strong SHAP contribution
- ICMPv6srcLnklayerLength, Protocol, Inter-Frame Delay — moderate importance across SHAP/LIME

Bottom Plot Highlights RNN_GRU:

- Source, Total Frame Size, Protocol, Code — high local SHAP contribution
- Inter-Frame Delay, Frame Number, Next Header — key for global SHAP



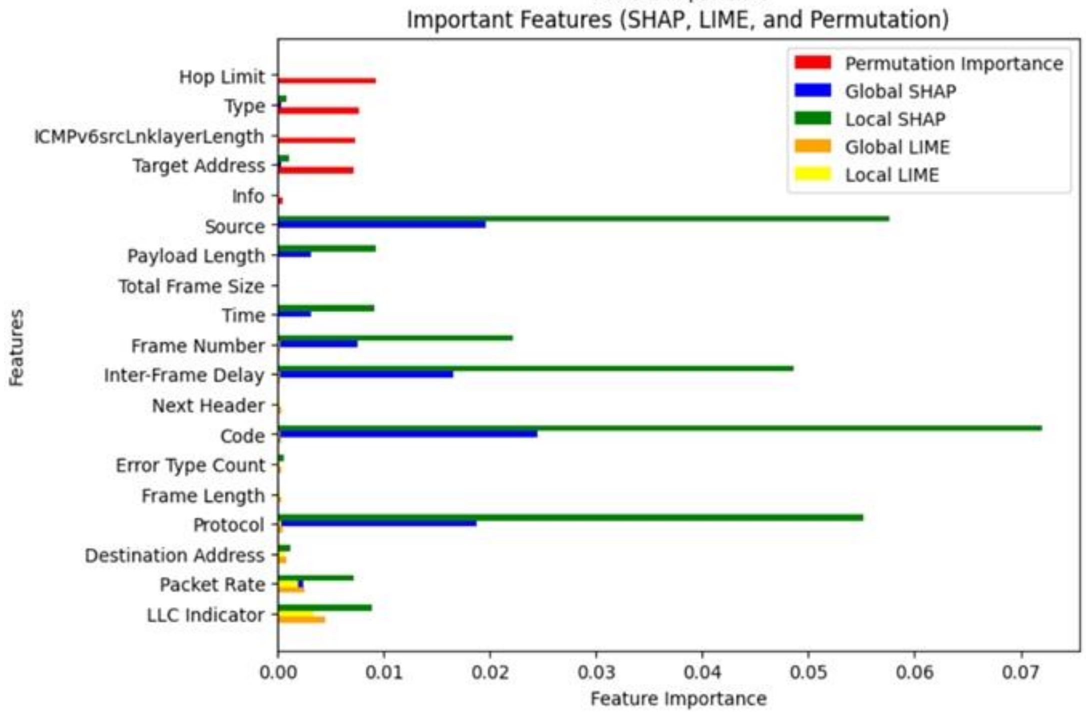


Figure 25 LTVM Feature contribution

Significant features from both models:

The most impactful features identified across SHAP, LIME, and permutation importance are:

1. Total Frame Size
2. Source

3. Protocol
4. Code
5. Next Header
6. Inter-Frame Delay
7. Time, LLC Indicator
8. Frame Number and Time

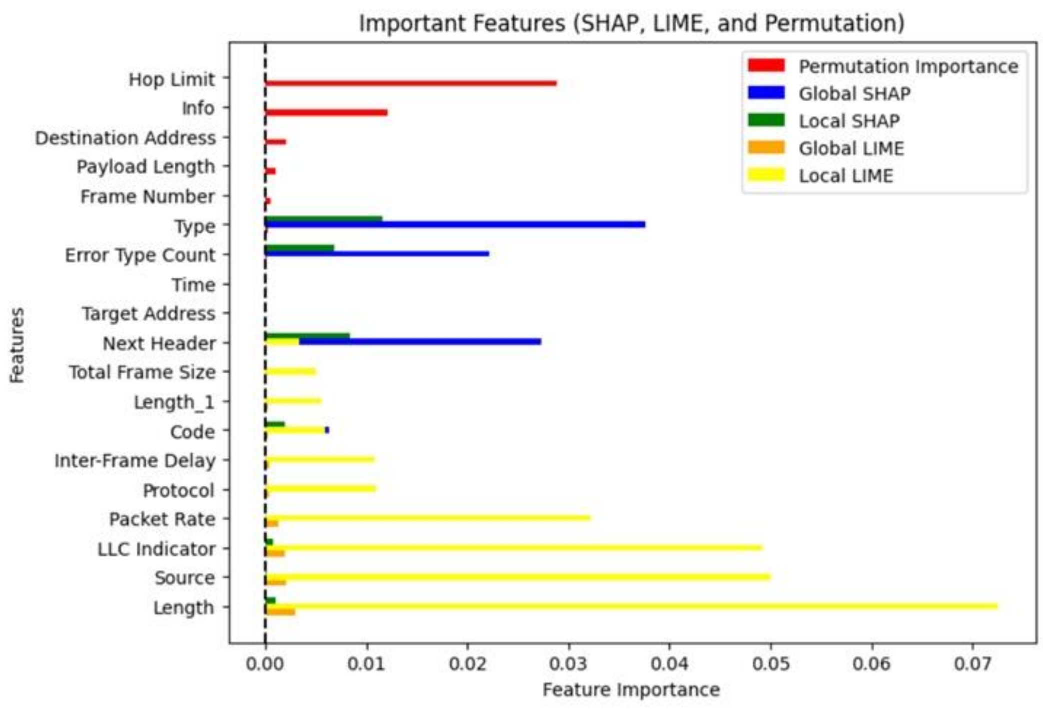
These features play a dominant role in model decision-making for the LTVM dataset, especially with SHAP providing consistent global and local importance.

Lab datasets: This dataset was created in a real lab setup using Cisco hardware and multiple virtual machines configured with IPv6. ICMPv6 attack traffic was generated from two Linux systems targeting a Windows server, with both normal and attack traffic captured over 4 hours and 45 minutes using Wireshark. The resulting 5.12 GB of traffic was converted into a sample Excel file of size 183 MB, containing 14 features. The data was pre-processed to include a "Class" column labelling each entry as “Attack” and “Normal”. Further detailed information about the dataset is given in Chapter 4, section 4.2.2. Feature engineering from the first two layers (physical and datalink layers), windowing, Time Series Split, ADASYN, SHAP, and LIME are applied.

Figure 26 provides brief information about the contribution of features that impact the model's performance.

Top Plot Highlights CNN_LSTM:

- Type and Next Header — high global SHAP
- LLC Indicator, Source, Length — dominant in local LIME
- Error Type Count and Time — visible across SHAP and permutation
- Payload Length, Frame Number — lower but consistent presence



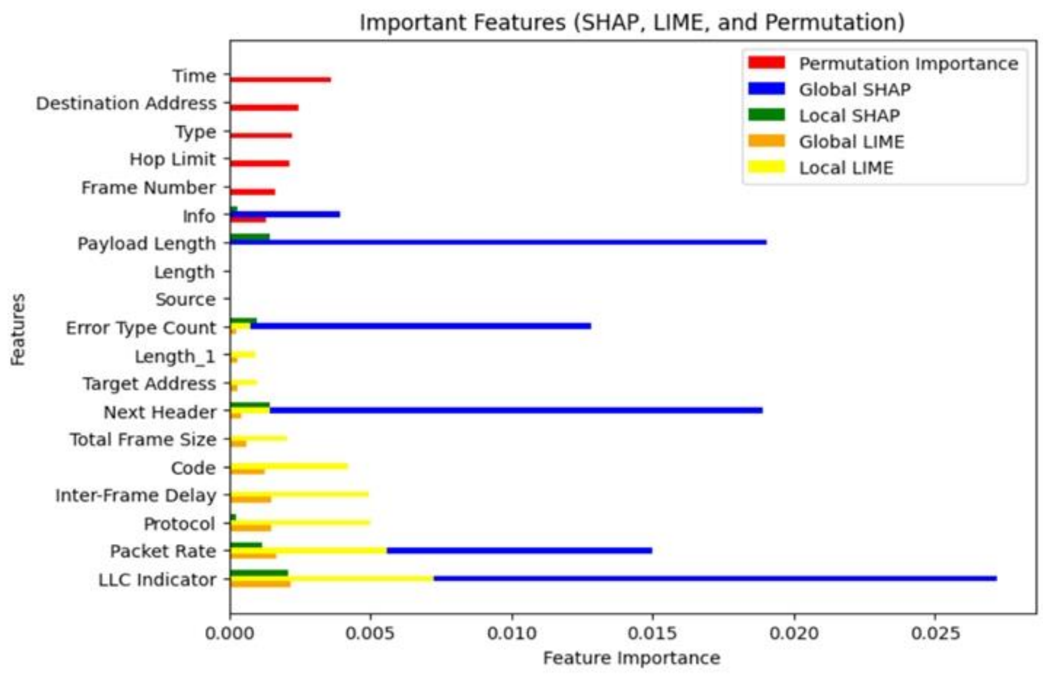


Figure 26 Lab Feature contribution

Bottom Plot Highlights RNN_GRU:

- Payload Length, Total Frame Size, Next Header, Error Type Count standout in global SHAP
- Packet Rate, Protocol, LLC Indicator important in local and global LIME
- Info, Hop Limit, Destination Address minor contribution via permutation

Significant features from both models:

1. **Next Header** – strong SHAP relevance
2. **Payload Length** – high SHAP importance
3. **Packet Rate** – key LIME-based contribution
4. **LLC Indicator** – impactful in local LIME
5. **Protocol** – consistently important in both LIME and SHAP
6. **Error Type Count** – relevant across multiple methods

These features are the most influential in driving model performance for the Lab dataset based on both global and local interpretability methods.

5.3 Comparative Evaluation with Baseline Models:

Table 16: Comparison with state-of-the-art ML and the proposed model.

Serial No	Author	Model / Algorithm	Detection of DDoS/ Dos	Datasets	Split ratio	Metric: Accuracy
1	Ojugo and Eboka 2020	Hidden Markov	DDoS	CIDDS-2017	Train-test split	80%
2.	Zewdie and Girma 2022	DT, KNN, RF	DDoS & DoS	CIC-IDS2017	Train-test split	92.19% to 99.66%
3.	Manjula and Mangla 2023	KNN, RF, Naïve Bayes	ICMP, TCP, and UDP flood attacks	Primary (self-generated datasets)	Train-test split	96.75%
4	Dasari and Kalari 2024	XGboost, LGBM, CatBoost, Random Forest (RF), and Decision Tree (DT)	DDoS	CIC-IDS2017	Train-test split	99.77%

5.	Liang and Znati 2019	SVM, RBF-SVM,KNN, Kmeans, NB, NN	DDoS	CAIDA and DARPA	Train-test split	77.03%
Comparison of the proposed model with the results						
6.	Proposed model 1 combination	CNN_LSTM	DDoS	Primary Dataset	Train-test split	99.36%
7.	Proposed model 2 combination	RNN-GRU	DDOS	Primary dataset	Train-test split	94.48%

Table 16 presents a comparative overview of various research efforts focused on detecting Distributed Denial of Service (DDoS) attacks using different models and algorithms. The primary metric for comparison in this table is Accuracy.

5.3.1 Comparative Summary of Model Performance (Based on Accuracy):

- **High-Performing Models:** Several studies report very high accuracy in detecting DoS/DDoS attacks.
    - **Dasari and Kalari (2024)**, using an ensemble of XGBoost, LGBM, CatBoost, Random Forest (RF), and Decision Tree (DT) on the CIC-IDS2017 dataset, achieved a remarkable 99.77% accuracy. This suggests that ensemble methods leveraging multiple tree-based algorithms can be highly effective for this task on this specific dataset.
    - **Proposed model** demonstrates strong performance on a Primary Dataset using both deep learning models: CNN_LSTM (99.36%) and RNN_GRU (94.48%). This highlights the potential of Recurrent Neural Networks (RNNs), particularly with Convolutional layers for feature extraction, in detecting these types of attacks.
    - **Manjula and Mangla (2023)** achieved 96.75% accuracy using KNN, RF, and Naive Bayes on their self-generated datasets for detecting ICMP, TCP, and UDP flood attacks. This indicates that even traditional machine learning algorithms can yield high accuracy depending on the dataset characteristics and the specific attack types targeted.

  - **Zewdie and Girma (2022)** reported a wide accuracy range of 92.19% to 99.66% using DT, KNN, and RF on the CIC-IDS2017 dataset. This broad range likely reflects variations in the specific algorithms or configurations used within their study.
- **Lower-Performing Models (Comparatively):** While still demonstrating reasonable detection capabilities, some models reported lower accuracy compared to the top performers.
  - **Ojujo and Eboka (2020)** obtained 80% accuracy using a Hidden Markov Model on the CIDDS-2017 dataset. While HMMs can be useful for sequential data analysis, their performance might be less competitive compared to more advanced machine learning and deep learning techniques for this type of detection task.
  - **Liang and Znati (2019)** achieved 77.03% accuracy using SVM, RBF-SVM, KNN, K-means, NB, and NN on the CAIDA and DARPA datasets. The lower accuracy here might be attributed to the complexity of these specific datasets or the suitability of the algorithms employed.

General Observations:

- **Dataset Influence:** The performance of the models is significantly influenced by the dataset used for training and testing. CIC-IDS2017 is commonly used benchmark, with some models achieving very high accuracy on it. The self-generated and CAIDA/DARPA datasets presented different levels of challenges.
- **Model Variety:** A wide range of models, from traditional machine learning algorithms (like Naive Bayes, KNN, Random Forest, Decision Trees, SVM) to deep learning architectures (CNN_LSTM, RNN_GRU, Neural Networks) and statistical models (Hidden Markov Model), have been applied to DoS/DDoS detection.
- **Ensemble Methods:** The highest accuracy reported in this table was achieved by an ensemble method, suggesting that combining the strengths of multiple models can lead to improved detection capabilities.

- **Deep Learning Potential:** The strong performance of CNN_LSTM and RNN_GRU indicates the potential of deep learning models to learn complex patterns in network traffic for effective DoS/DDoS detection.

In conclusion, the best performing models in this comparison, based solely on accuracy, are the ensemble approach by Dasari and Kalari, and the deep learning models (CNN_LSTM and RNN_GRU) from proposed model combinations on their respective datasets.

5.3.2 Base Line Model 1, model 2, and Stacked Model Results Evaluation:

Table 17: Results of 3 IPv6 datasets using Feature Engineering:

Serial No.	Datasets	IPv6 or IPv4	Number of Features	Train Test Series	Model 1 and Model 2	Metric: Accuracy
1	Sain Malaysian	Ipv6	Original 11 + Feature Engineering 5 = Total 16	Yes	CNN_LSTM	81.56%
					RNN_GRU	82.58%
2	Primary dataset 1 (LTVM)	Ipv6	Original 13 + Feature Engineering 5 = Total 18	Yes	CNN_LSTM	99.91%
					RNN_GRU	94.45%
3.	Primary dataset 2 (Lab)	Ipv6	Original 13 + Feature Engineering 5 = Total 18	Yes	CNN_LSTM	99.98%
					RNN_GRU	98.69%

By applying feature engineering from the Data Link Layer (Layer 2) and the Physical Layer (Layer 1), 5 additional features were extracted. Initial model evaluation, employing a standard 80:20 train-test split, yielded promising accuracy results ranging from 81.56% to 99.98%. Notably, the close performance scores of the two primary models (CNN_LSTM and RNN_GRU, as seen in the above table) suggested the potential for improvement through Ensemble Stacking.

To further investigate the proposed model's behaviour, the experiments were extended in their scope to include two additional datasets and the implementation of Ensemble Stacking (ES). In this extended evaluation, the data splitting strategy was adapted to a Time Series Split, Average Attack Detection Alarm is applied and the ES technique is used to get standard results that can also provide prediction accuracy.

Explanation of each metric with respect to the 5 datasets used :

Table 16 presents the performance of two models (Model 1 and Model 2), specifically CNN_LSTM and RNN_GRU, on five different network intrusion detection datasets (Sain Malaysian, Primary dataset (LTVM), Primary dataset (Lab), NSL_KDD, and Mendeley).

The proposed model was evaluated on the above datasets, with varying numbers of features and using a time series split validation strategy. The key performance metrics reported are Accuracy, Precision, Recall, and F1-Score, along with the ADASYN application and Ensemble Stacking results.

The evaluation metrics, Accuracy, Precision, Recall, and F1-Score, are critical in assessing the performance of DDoS detection models. Across most datasets, particularly the Mendeley (IPv4) and Primary (LTVM and Lab, IPv6), both CNN_LSTM and RNN_GRU models demonstrate high accuracy (often exceeding 95%, and reaching 100% in some cases), indicating strong overall performance. However, accuracy alone can be misleading, especially in imbalanced datasets where a model may perform well by favouring the majority class.

To complement this, precision measures how often the model's positive predictions (i.e., attack detections) are correct. High precision (often above 90%) across most datasets confirms that false positives are minimal, especially in Mendeley and Ensemble Stacking scenarios.

Table 18: 5 datasets extended to Ensemble stacking

Serial No.	Datasets	IPv6 or IPv4	Number of Features	Time Series Split	ADASYN: Yes /No	Model 1 and Model 2	Metric:	AAD Alarm Time (ms)	Ensemble Stacking results
1	Sain Malaysian	IPv6	Original 11 + Feature Engineering 5 = Total 16	Yes	Yes	CNN_LSTM	Recall: 91.80% Precision:18.46 % F1 Measure: 30.74% Accuracy: 61.55%	0:00:00.000222	Recall:93.54% Precision:20.16 % F1 Measure:33.18% Accuracy:64.97 %
						RNN_GRU	Recall: 91.94% Precision:18.39 % F1 Measure: 30.66% Accuracy: 61.34%	0:00:00.000527	
2		IPv6	Original 13 + Feature	Yes	Yes	CNN_LSTM	Recall: 99.48%	0:00:00.000447	Recall: 99.69%

	Primary dataset 1 (LTVM)		Engineering 5 = Total 18				Precision:96.32 % F1 Measure: 97.87% Accuracy: 96.30%		Precision:95.50 % F1 Measure: 97.55 % Accuracy: 95.72%
						RNN_GRU	Recall: 100.0% Precision:85.63 % F1 Measure:92.26 % Accuracy: 85.63%	0:00:00.000197	
3.	Primary dataset 2 (Lab)	IPv6	Original 13 + Feature Engineering 5 = Total 18	Yes	Yes	CNN_LSTM	Recall: 90.90% Precision:100.0 % F1 Measure: 95.23% Accuracy: 99.99%	0:00:00.000195	Recall: 100.0 % Precision:100.0 % F1 Measure: 1.0% Accuracy: 1.0%

						RNN_GRU	Recall: 90.92% Precision:100.0 % F1 Measure: 92.32% Accuracy: 99.98%	0:00:00.000459	
4	NSL_KDD	IPv4	Original 44 + Feature Engineering 0 = Total 44 (No Feature Engineering)	Yes	Yes	CNN_LSTM	Recall: 68: 73% Precision: 98.77% F1 Measure: 81.05 % Accuracy: 78.27%	0:00:00.000261	Recall: 85.65% Precision:99.77 % F1 Measure: 92.17% Accuracy:90.16 %
						RNN_GRU	Recall: 73.72% Precision:99.21 % F1 Measure: 84.59 % Accuracy: 81.83%	0:00:00.000297	

5	Mendeley	IPv4	Original 22 + Feature Engineering 4 = Total 26	Yes	Yes	CNN_LSTM	Recall: 100.0% Precision:1.0% F1 Measure: 100.0% Accuracy: 100.0%	0:00:00.000397	Recall: 100.0% Precision:1.0% F1 Measure: 100.0% Accuracy: 100.0%
						RNN_GRU	Recall: 100.0% Precision:1.0% F1 Measure: 100.0% Accuracy: 100.0%	0:00:00.000220	

However, lower precision, such as RNN_GRU’s 30% on the Sain Malaysian dataset, reveals cases where the model frequently misclassifies benign traffic as attacks.

Recall, on the other hand, evaluates the model’s ability to detect all actual attack instances. Although recall is high in several cases, significant drops (as low as 6% for both models on the Sain Malaysian dataset) highlight scenarios where many attacks go undetected, severely limiting the model's practical security effectiveness.

The F1-Score serves as a balanced indicator by harmonizing precision and recall. Consistently high F1-Scores (close to or at 1.0) in the Mendeley and Primary datasets underscore effective and reliable detection. In contrast, very low F1-Scores on challenging datasets like Sain Malaysian (as low as 8–11%) illustrate an imbalanced or inconsistent detection capability.

5.3.3 Conclusion on Best Model Performance:

- **Dataset Dependency:** The best-performing model seems to be highly dependent on the specific dataset. The Mendeley dataset (IPv4) consistently shows perfect or near-perfect performance for both CNN_LSTM and RNN_GRU across all metrics. This suggests that this dataset might be relatively easier to classify or that the features are highly discriminative for the types of attacks present.
- **Model Consistency:** While both CNN_LSTM and RNN_GRU show strong performance on some datasets, neither model consistently outperforms the other across all scenarios. In some cases, CNN_LSTM shows slightly better results (e.g., on NSL_KDD), while in others, RNN_GRU performs comparably well (e.g., on the Primary datasets).
- **Challenge of Sain Malaysian (IPv6):** Both models struggle significantly with the Sain Malaysian dataset (IPv6), exhibiting very low recall and consequently poor F1-Scores. This suggests that the features or the nature of attacks in this dataset might be more complex or require different modelling approaches.

- **Potential of Ensemble Stacking:** The Ensemble Stacking results often show improved performance, particularly in terms of recall and F1-Score, suggesting that combining the predictions of CNN_LSTM and RNN_GRU can lead to a more robust and accurate intrusion detection system.
- **NSL_KDD Difficulty:** The NSL_KDD dataset (IPv4) appears to be more challenging than the Primary datasets and Mendeley for both individual models.

5.3.4 Summary:

Based on the provided results, it's difficult to definitively declare one model as universally "best" i.e. **CNN_LSTM or RNN_GRU.**

- For the **Mendeley dataset (IPv4)**, both **CNN_LSTM and RNN_GRU** demonstrate excellent performance with perfect or near-perfect scores across all metrics.
- For the **Primary datasets (LTVM and Lab) with IPv6**, both models also perform very well with high accuracy, precision, recall, and F1-scores.
- The **Sain Malaysian dataset (IPv6)** poses a significant challenge for both models.
- The **NSL_KDD dataset (IPv4)** shows moderate performance, with CNN_LSTM generally exhibiting slightly better metrics than RNN_GRU on this specific dataset.
- **Ensemble Stacking** appears to be a promising approach for improving overall performance by leveraging the strengths of both individual models.

The robustness of the proposed model is discussed in detail in the following section, mostly **focused on the Primary datasets (Lab Dataset)** as it is generated in the context of **DDoS attack scenario**.

5.4 Evaluation of the Model:

The common methods that are applied on all the datasets:

5.4.1 Feature Engineering:

5 features were extracted based on the following formulae:

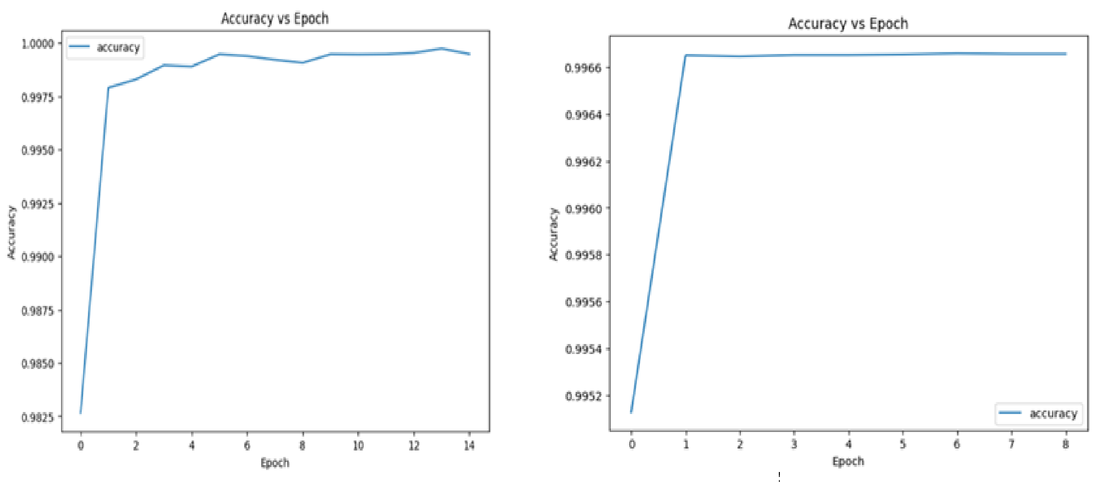
- **Error Type Count:** Count the occurrences of specific error types within a defined time interval (Δt). Let Ci(t) be the cumulative count of error type i up to time t.
  The error type count for error type i in the interval [t1,t2] is: ErrorCounti=Ci(t2)−Ci(t1) .Where Δt=t2−t1.
- **Inter-Frame Delay (IFD):** Measure the time difference between the arrival of consecutive frames at a network interface. Let Ti be the arrival timestamp of the i-th frame. The inter-frame delay between the (i−1)-th and i-th frame is: IFDi=Ti−Ti−1.Aggregation over a time window (Δt) and Average IFD: $\overline{\mathrm{IFD}} = \frac{1}{N}\sum_{i=1}^{N}\mathrm{IFD}_i$ .where N is the number of inter-frame delays calculated within Δt
- **Packet Rate:** Measure the number of frames (packets at Layer 2) observed on a network interface within a specific time interval. Let N(Δt) be the number of frames observed during the time interval Δt=t2−t1. The packet rate is: $\frac{N(\Delta t)}{\Delta t}$.
- **Total Frame Size:** Measure the size of each Data Link Layer frame and aggregate these sizes over a time interval. Let Si be the size (in bytes) of the i-th frame. Average Frame Size over Δt (with N frames): $\overline{S} = \frac{1}{N}\sum_{i=1}^{N} S_i$ and Total Bytes Transferred over Δt: $\sum_{i=1}^{N} S_i$. Rate of Bytes Transferred (Throughput)= $\frac{\mathrm{TotalBytes}}{\Delta t}$
- **LLC Indicator:** Identify if a Data Link Layer frame uses Logical Link Control (LLC) encapsulation (primarily relevant for older Ethernet standards like 802.3). For Ethernet II frames, the EtherType field in the header directly indicates the next-level protocol. If a frame uses 802.3/LLC, the EtherType field is typically 0x0000.Header Presence, If the EtherType is 0x0000, the following bytes constitute the LLC header (Destination SAP, Source SAP, Control field). Binary Indicator: $\mathrm{LLC\_Present} = \begin{cases} 1 & \text{if EtherType} = 0x0000 \\ 0 & \text{otherwise} \end{cases}$

Feature engineering is not applied to the NSL_KDD datasets and the rest 4 datasets were applied. Time Series Split, Rolling windows, and ADASYN are applied to all 5 datasets. The NSL dataset can be treated as a different scenario, and the results would be highlighted.

5.4.2 Time Series Split: The time-series strategy is particularly important when working with time-dependent data, such as network traffic, sensor readings, or financial data. Its key significance lies in preserving the temporal order of data, with standard cross-validation methods. The datasets that were used for experiments were related to network traffic, and the primary dataset is generated in the Networking lab, created in a scenario of Volumetric DDoS attack. Further, the dataset is steadily composed in the initial which is treated as "Normal" instances, and after sufficient time, there is a sudden dramatic surge that significantly indicates an anomaly towards the end, categorizing them as "Attack" instances. This suggests a period of intense attack activity concentrated within a specific timeframe. The stark contrast between the prolonged periods of normal activity and the sudden attack burst is a key characteristic of this dataset.

5.4.3 Rolling window: This method uses a fixed-size window of recent data for both training and testing, which moves forward in time for each split. The choice between these methods depends on the characteristics of the time series data and the specific forecasting problem. The rolling window focuses on evaluating the model on more recent patterns.The Critical evaluation is mainly focused on the Primary dataset (Lab), considered as a very good sample, as this dataset is very close to the Volumetric DDoS attacks scenario when compared to the remaining datasets:

5.4.4 Accuracy & Training validations:



(CNN_LSTM) (RNN_GRU)

Figure 27 Accuracy vs Validation (CNN_LSTM & RNN_GRU)

Figure 27 illustrates CNN_LSTM the "Accuracy vs Epoch" graph showing a learning curve where the model's accuracy rapidly improves in the early stages of training and then quickly reaches a very high level, followed by a plateau with slight fluctuations. Whereas in RNN_ GRU it is constant. This suggests that the model learned the training data effectively within the first few epochs in both cases, and further training did not provide substantial gains. The near-perfect final accuracy in CNN-LSTM and overall high accuracy in RNN_GRU indicate a very strong fit to the training data in both combinations.

The CNN_LSTM diagnostic plot shows that both the training and validation loss are very low and decrease in the initial epochs. The validation loss remains consistently low and close to the training loss throughout the training period with reasonable fluctuations. This suggests that the model is learning reasonably and generalizing well to the validation data, with no significant signs of overfitting or underfitting.

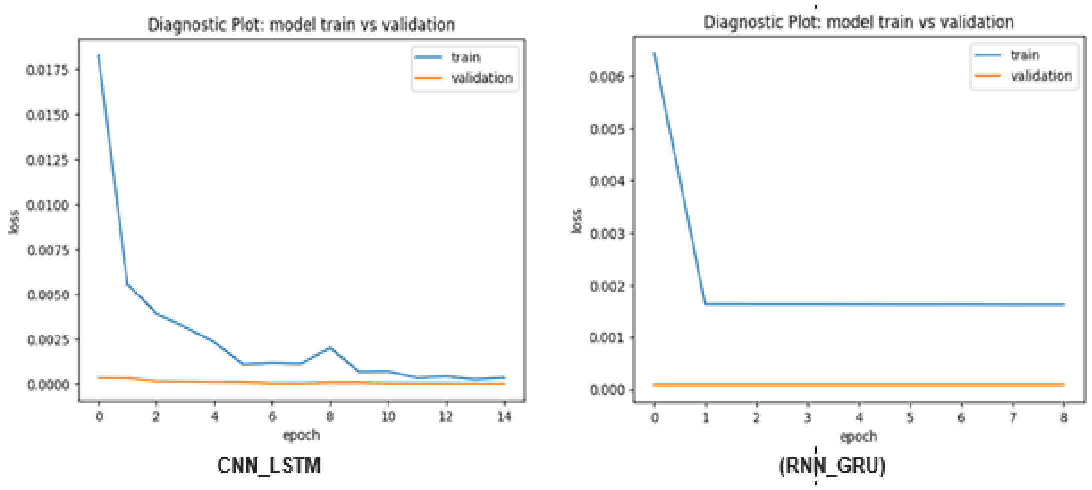


Figure 28 Train vs Validation (CNN_LSTM & RNN_GRU)

Similarly, Figure 28 show RNN-GRU diagnostic plot shows that both the training and validation loss are very low and decrease significantly in the initial epoch. The validation loss remains consistently low and close to the training loss throughout the training period. This suggests that the model is learning effectively and generalizing well to the validation data, with no significant signs of overfitting or underfitting. The low loss values on both sets indicate a good model fit.

5.4.5 ROC_AUC

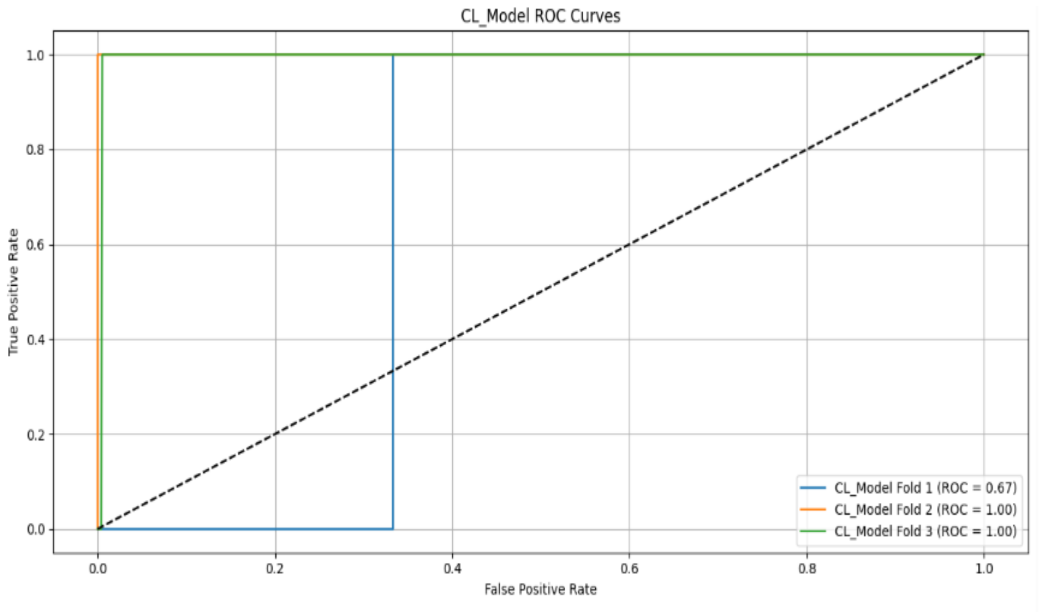


Figure 29 CL Model ROC curve

Figure 29 illustrates Receiver Operating Characteristic (ROC) curves for a "CL_Model" across three different folds from a cross-validation procedure. Each curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various classification thresholds. The diagonal dashed line represents a random classifier (AUC = 0.5). There are three distinct ROC curves, each representing the performance of the CL_Model on a different fold of the data. Fold 1 (ROC = 0.67) as shown in blue, Fold 2 (ROC = 1.00) as shown in orange, and the third Fold 3 (ROC = 1.00) as shown in green. The graph also provides the Area Under the Curve (AUC) for each fold. The AUC is a single scalar value that summarizes the classifier's overall performance across all possible thresholds. Fold 1 has an AUC of 0.67, Fold 2 has an AUC of 1.00 and Fold 3 has an AUC of 1.00.

- **CL Model Evaluation:** The model achieves perfect performance on two folds but only moderate performance on the other. This indicates variability in the model's ability to generalize across different subsets of the data. The overall assessment of the CL_Model provides the average or the distribution of these AUC scores across all folds.
- **Summary:** The image shows that the CL_Model performed perfectly on two out of three cross-validation folds (AUC = 1.00), but its performance was

considerably lower on the first fold (AUC = 0.67). This indicates that the model's effectiveness might depend on the specific data subset it is evaluated.

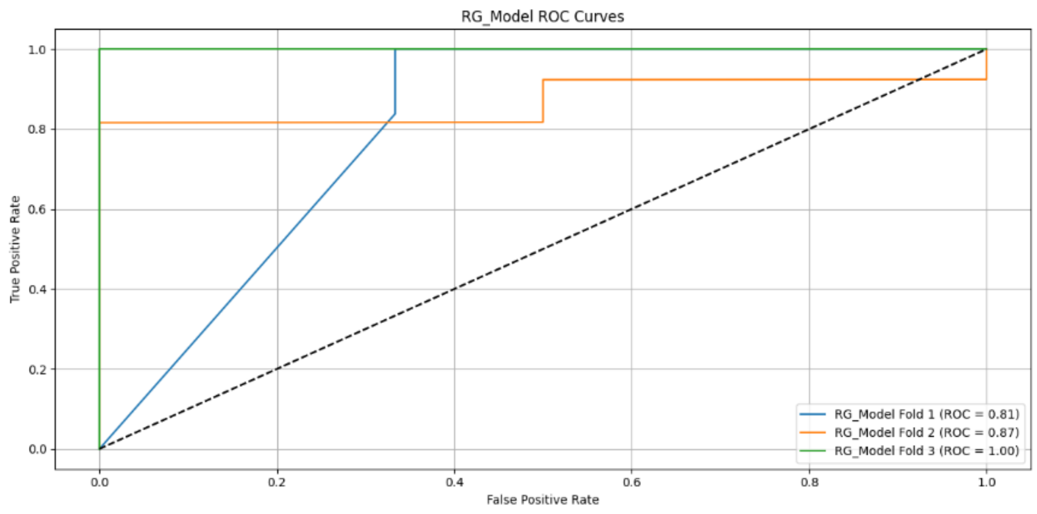


Figure 30 RG Model ROC curve

Figure 30 illustrates Receiver Operating Characteristic (ROC) curves for a "RG_Model" across three different folds from a cross-validation procedure. The plot shows three distinct ROC curves, each corresponding to a different fold of the RG_Model. Fold 1 (ROC = 0.81) is shown in blue, Fold 2 (ROC = 0.87) is shown in orange, and Fold 3 (ROC = 1.00) is shown in green. The graph also provides the AUC value for each fold, which summarizes the overall performance of the classifier on that specific fold, where Fold 1 has an AUC of 0.81, Fold 2 has an AUC of 0.87, and Fold 3 has a perfect AUC of 1.00.

- **Model Evaluation:** The RG_Model shows strong and consistent performance across the three folds, with AUC scores ranging from 0.81 to 1.00. This indicates that the model is generally effective at classification, with particularly excellent performance on the data subset corresponding to Fold 3. The variability in AUC scores indicates that the model's performance might be slightly influenced by the specific data it is trained and evaluated.
- **Summary:** The ROC curves indicate that the RG_Model is a good classifier, achieving high AUC scores across all three cross-validation folds, with one fold demonstrating perfect classification.

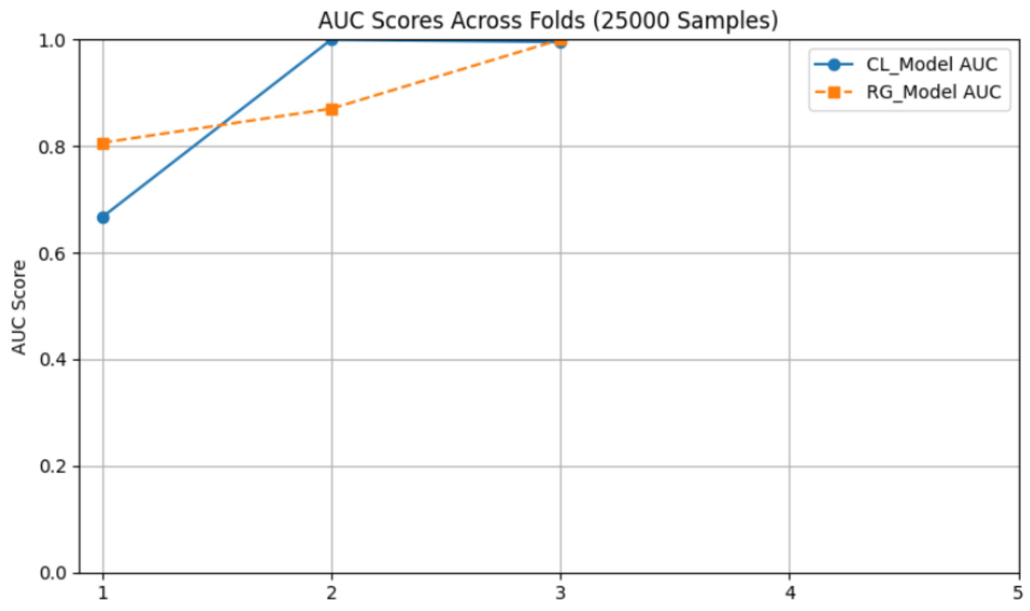


Figure 31 Comparison of CL and RG Models based on AUC

Figure 31 depicts the comparison of the Area Under the Curve (AUC) scores of two models, "CL_Model" and "RG_Model", across three folds of a dataset containing 25,000 samples. AUC of CL model is indicated by a blue line, and AUC of RG model is indicated by an orange line. Both models show an improvement in AUC scores as they move from Fold 1 to Fold 3. By Fold 3, both models achieve perfect classification performance on their respective data subsets.

- **Summary:** The graph indicates that both the CL_Model and the RG_Model show good to excellent performance across the three folds, with both achieving perfect AUC scores on the third fold. The RG_Model initially performs better on the first fold, but the CL_Model shows a more dramatic improvement between Fold 1 and Fold 2.

5.5. Sample Size Sensitivity:

```
[BASE] Sample Size 10000: F1 = 0.9979
[BASE] Sample Size 25000: F1 = 0.9993
[BASE] Sample Size 50000: F1 = 0.9998
[BASE] Sample Size 75000: F1 = 0.9997
[STACKED] Sample Size 15000: F1 = 0.9989
[STACKED] Sample Size 30000: F1 = 0.9994
[STACKED] Sample Size 60000: F1 = 0.9998
[STACKED] Sample Size 100000: F1 = 0.9998
```

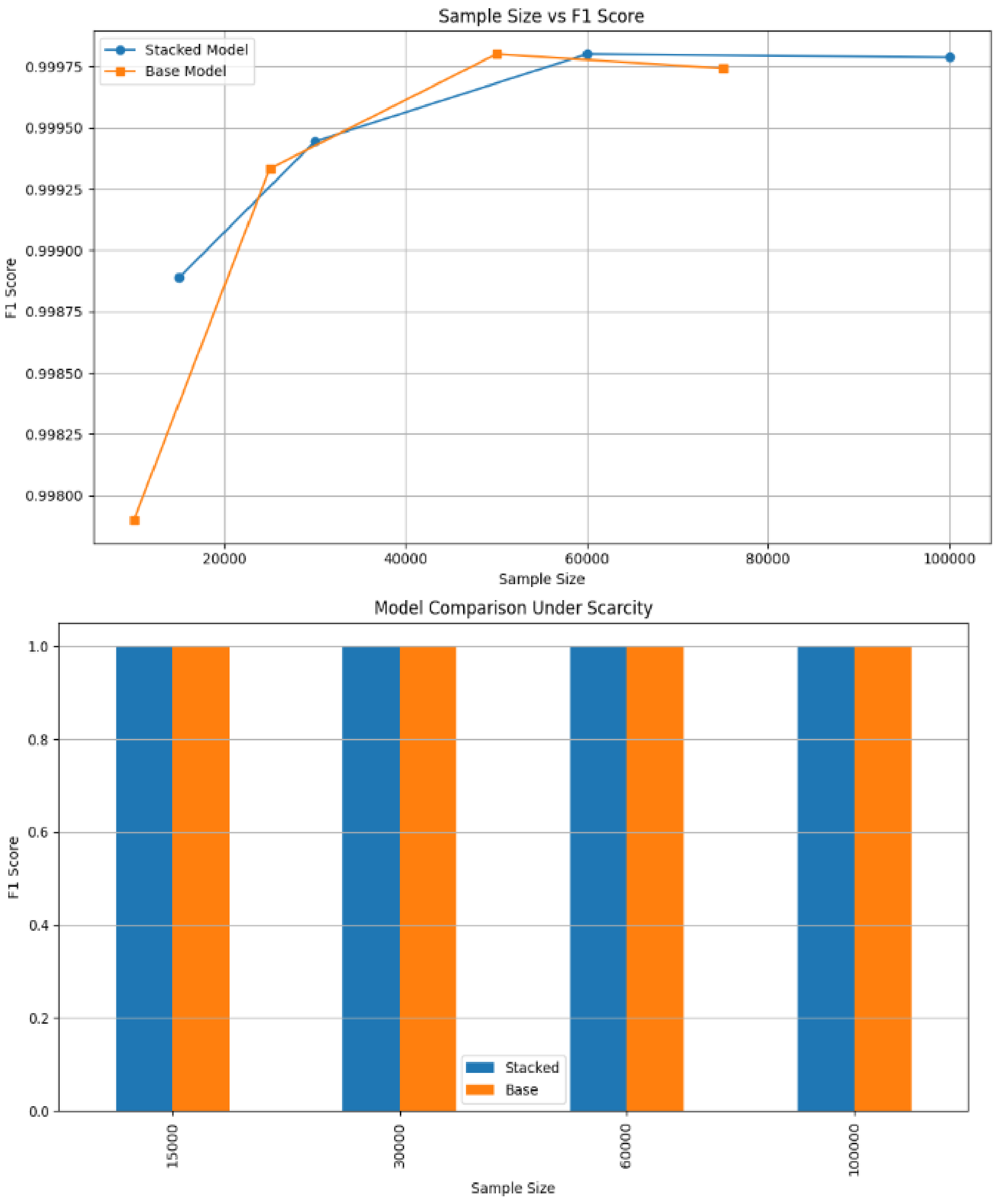


Figure 32: Sample Size Sensitivity

Figure 32 demonstrates that the base and stacked samples are used for size sensitivity. Relatively, line and scatter charts are generated, which explain that reducing the training dataset size leads to a decrease in the F1 score for both the Base and Stacked models. The Stacked Model is more robust to data scarcity, maintaining a higher F1 score than the Base Model when the sample size is small. As the sample size increases, the performance gap between the two models tends to narrow. The study uses F1 score as the primary performance metric and visually presents the relationship between sample size and F1 score using a line plot and a bar chart comparing the models under data scarcity.

5.6 Cross-Validation Folds

Table 19 summarizes the performance of a Base MLP model and a Stacked Model using 3-fold, 5-fold, and 10-fold cross-validation.

Table 19: Cross-Validation Folds for Base MLP and Stacked Model

Model CV Folds F1 Score (avg ± std) AUC Score (avg ± std)				
0	Base MLP	3	0.9986 ± 0.0000	0.9807 ± 0.0023
1	Stacked Mode	3	0.9986 ± 0.0000	0.9813 ± 0.0024
2	Base MLP	5	0.9986 ± 0.0000	0.9814 ± 0.0036
3	Stacked Model	5	0.9986 ± 0.0000	0.9813 ± 0.0025
4	Base MLP	10	0.9986 ± 0.0000	0.9816 ± 0.0060
5	Stacked Model	10	0.9986 ± 0.0000	0.9817 ± 0.0061

For each number of folds, the average and standard deviation of the F1 score and AUC score are reported.

- Cross-Validation Runs: The experiment involved running both the Base MLP and the Stacked Model with 3-fold, 5-fold, and 10-fold cross-validation.
- Mean and Standard Deviation: The table explicitly tracks and reports the mean (average) and standard deviation (std) of both the F1 score and the AUC score for each model and each cross-validation configuration.
- Performance: The data given in the above table describes the best statistical performance of the Model related to number of CV Folds, average and standard deviations for both F1 and AUC Scores.

- Impact of More Folds on Reliability: Observing the standard deviation of the scores, there isn't a consistent or significant decrease in the standard deviation as the number of folds increases from 3 to 10 for either model and for both metrics (F1 and AUC). The statistics from the given table reveal that the standard deviation of the AUC score tends to increase with more folds in the case of Base MLP(Classifier).

  Generally, a higher number of folds is expected to provide a more robust estimate of the model's generalization performance and potentially lower the variance of the estimates. However, in this specific case, the performance of both models appears very stable across different folds, resulting in very low standard deviations that don't show a clear trend of decreasing with more folds. Therefore, based on this data, it's not definitively evident that more folds significantly improve the reliability (reduce variance) of the performance estimates for these particular models and dataset. The already high and consistent performance might be contributing to this lack of substantial change in standard deviation.

5.7 ADASYN

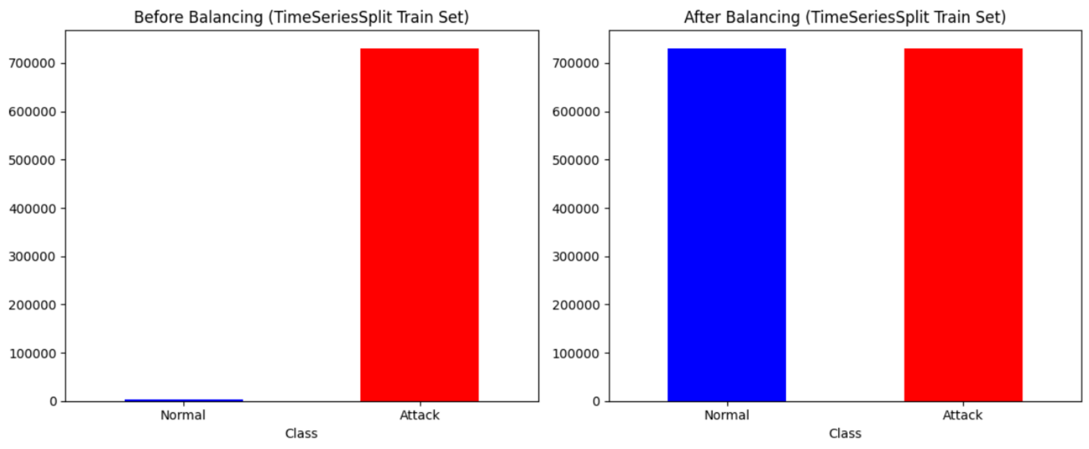
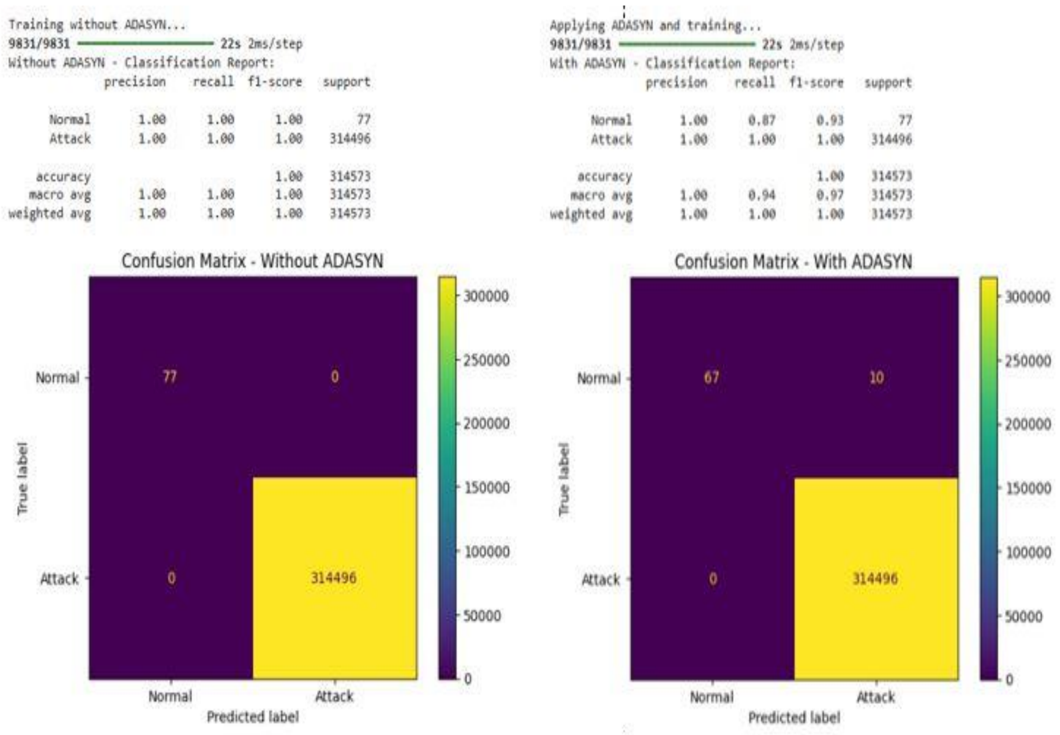


Figure 33 ADASYN - Balancing Normal and Attack

Figure 33 illustrates the effect of a balancing technique on a time series split training dataset. Initially, the dataset was severely imbalanced, with a minuscule representation of the "Normal" class compared to the overwhelming majority of "Attack" instances. After balancing, the number of "Normal" instances has been significantly increased to achieve a more even distribution between the two classes. This balancing step is crucial in training machine learning models to prevent bias towards the majority class

("Attack") and to improve the model's ability to correctly identify instances of the minority class ("Normal").

5.7.1 Confusion Matrix:



```
Training without ADASYN...
9831/9831 ━━━━━━━━━━━━━━━━━━━━ 22s 2ms/step
Without ADASYN - Classification Report:
              precision    recall  f1-score   support

      Normal       1.00      1.00      1.00        77
      Attack       1.00      1.00      1.00    314496

    accuracy                           1.00    314573
   macro avg       1.00      1.00      1.00    314573
weighted avg       1.00      1.00      1.00    314573
```

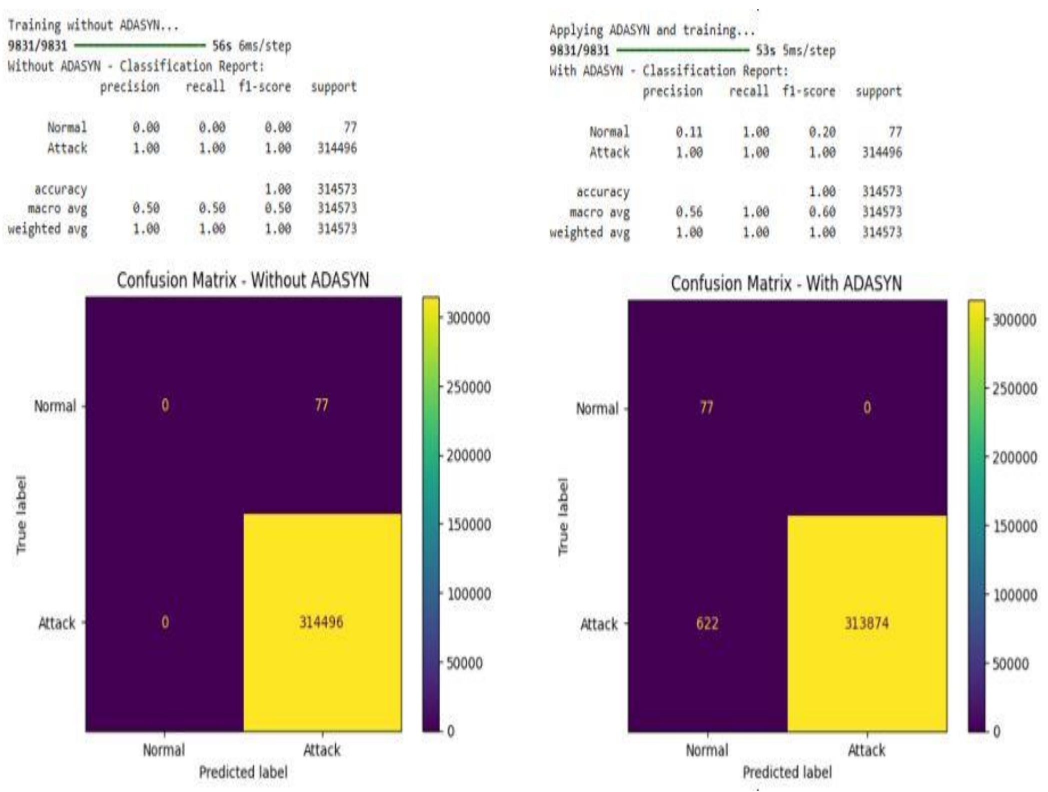
```
Applying ADASYN and training...
9831/9831 ━━━━━━━━━━━━━━━━━━━━ 22s 2ms/step
With ADASYN - Classification Report:
              precision    recall  f1-score   support

      Normal       1.00      0.87      0.93        77
      Attack       1.00      1.00      1.00    314496

    accuracy                           1.00    314573
   macro avg       1.00      0.94      0.97    314573
weighted avg       1.00      1.00      1.00    314573
```

Figure 34: Confusion Matrix before ADASYN and AFTER ADASYN CNN_ LSTM

Figure 34 depicts the Confusion Matrix (CM) of CNN_LSTM, after applying ADASYN, while the "Attack" class performance remains perfect, the "Normal" class experiences a slight decrease in precision, leading to 10 false positives. The overall accuracy remains at 1.00, but the introduction of false positives for the "Normal" class suggests that ADASYN, which aims to balance the class distribution by synthesizing new minority class samples, might have introduced some complexity that led to misclassification of some "Normal" instances as "Attack" in this specific case.

5.7.2 Confusion Matrix after ADASYN

Similarly, Figure 35 depicts the Confusion Matrix (CM) of RNN_GRU, without ADASYN, the model completely fails to correctly classify any "Normal" instances, classifying them all as "Attack". This results in perfect recall for both classes but zero precision and F1-score for "Normal".



```
Training without ADASYN...
9831/9831 ━━━━━━━━━━━━━━━━━━━━ 56s 6ms/step
Without ADASYN - Classification Report:
              precision    recall  f1-score   support

      Normal       0.00      0.00      0.00        77
      Attack       1.00      1.00      1.00    314496

    accuracy                           1.00    314573
   macro avg       0.50      0.50      0.50    314573
weighted avg       1.00      1.00      1.00    314573
```

```
Applying ADASYN and training...
9831/9831 ━━━━━━━━━━━━━━━━━━━━ 53s 5ms/step
With ADASYN - Classification Report:
              precision    recall  f1-score   support

      Normal       0.11      1.00      0.20        77
      Attack       1.00      1.00      1.00    314496

    accuracy                           1.00    314573
   macro avg       0.56      1.00      0.60    314573
weighted avg       1.00      1.00      1.00    314573
```

Figure 35 Confusion Matrix before ADASYN and AFTER ADASYN RNN_ GRU

After applying ADASYN, the model's ability to identify "Normal" instances dramatically improves, achieving perfect recall for this class. However, this comes at the cost of introducing a small number of false negatives for the "Attack" class. The overall accuracy remains at 1.00 in both scenarios, but the model with ADASYN demonstrates a much better ability to correctly identify both classes, even though it introduces a minor error in classifying "Attack" instances. ADASYN appears to have effectively addressed the issue of the model being unable to predict the minority "Normal" class, likely by balancing the class distribution during training.

5.8 Extended Analysis:

5.8.1 Rolling Window Analysis:

The information regarding "Window Size" and "Stride" is provided at the bottom of the image as part of the "Final Evaluation" section.

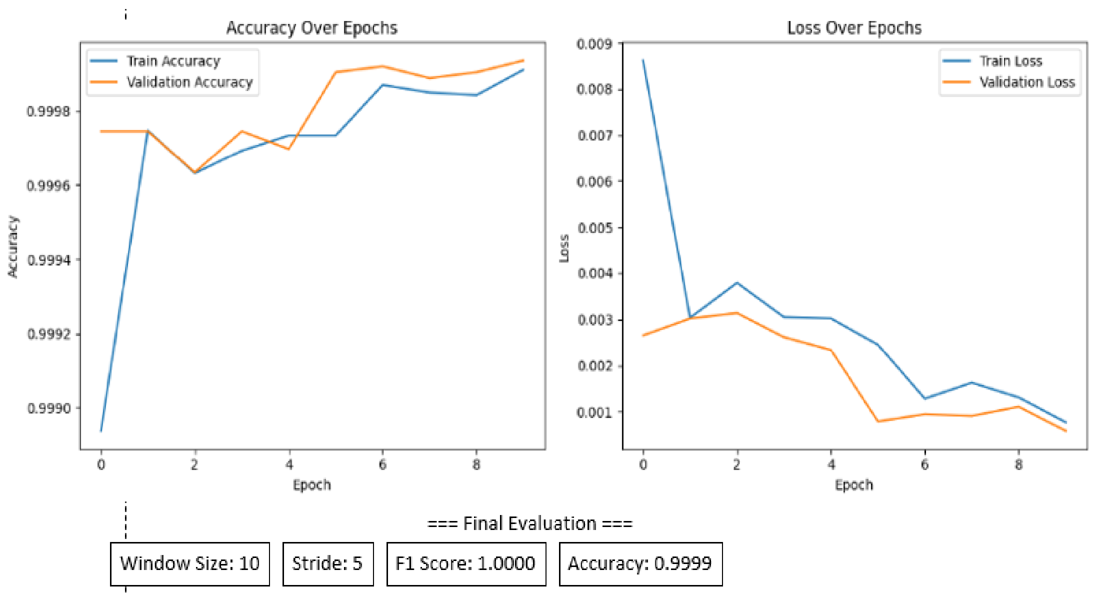


Figure 36 Rolling Windows Analysis

Figure 36: Explanation of Window Size, Stride, and Reasoning:

- Window Size: 10: This refers to the length of the temporal window used as input to the time-series model. It means that the model processes sequences of 10 consecutive data points at a time. The reasoning behind choosing a window size of 10 would depend on the underlying temporal dependencies in the data. A window too small might not capture relevant patterns, while a window too large could include irrelevant information or increase computational cost. In this context, a window size of 10 is used throughout the experiments.
- Stride: 5: The stride defines the step size by which the sliding window moves across the time series. A stride of 5 means that after processing a window of 10 data points, the next window starts 5 data points after the beginning of the previous window. Using a stride smaller than the window size (in this case, 5 < 10) results in overlapping windows. The reasons for using a stride of 5 are as follows:
  - Increase the number of training samples: Overlapping windows generate more data points for training the model.
  - Capture finer-grained temporal patterns: By looking at overlapping segments, the model might be better able to learn subtle changes and dependencies in the time series.
- Performance Changes with Temporal Slices:

The two graphs in the image illustrate how the model's performance (Accuracy and Loss) changes over training epochs. Each epoch represents a complete pass through the training data, which is implicitly structured into temporal slices due to the windowing and striding process.

- Accuracy Over Epochs (Left Graph):
    - The blue line shows the training accuracy, which generally increases over the epochs, indicating that the model is learning to classify the training data correctly.
    - The orange line shows the validation accuracy, which also increases initially but seems to plateau and even slightly decrease towards the later epochs. This suggests that the model might be starting to overfit the training data after a certain point, as its performance on unseen validation data is no longer improving significantly. The validation accuracy reaches a high level, indicating good generalization.
- Loss Over Epochs (Right Graph):
    - The blue line shows the training loss, which decreases over the epochs, as expected during the training process, where the model adjusts its weights to minimize the error on the training data.
    - The orange line shows the validation loss, which initially decreases but then starts to increase after a few epochs. This is another indicator of potential overfitting. While the model continues to reduce loss on the training data, its ability to generalize (as measured by the validation loss) deteriorates.
- Graphs to Illustrate Time-Series Model Behaviour:

The two line plots effectively illustrate the training dynamics of the time-series model over epochs. While the x-axis represents epochs (iterations of training), each epoch involves processing the time-series data in temporal slices defined by the window size and stride.

- Increasing Training Accuracy and Decreasing Training Loss: These trends suggest that the model is successfully learning the patterns within the temporal slices of the training data.
- Plateauing/Decreasing Validation Accuracy and Increasing Validation Loss: These trends are crucial for understanding the model's generalization ability

on unseen temporal slices (the validation set). The divergence between training and validation performance indicates that the model might be memorizing the training data rather than learning generalizable features from the temporal sequences.

Summary:

- F1 Score: 1.0000: This indicates perfect precision and recall on the test set, meaning the model correctly identified all positive instances and did not have any false positives or false negatives.
- Accuracy: 0.9999: This shows a very high overall accuracy on the test set, meaning the model correctly classified almost all instances.

The high final evaluation metrics suggest that despite the potential signs of slight overfitting observed in the validation curves towards the end of training, the model ultimately achieved excellent performance on unseen temporal sequences.

5.8.2 Inclusion of Physical Layer Features:

Figure 37 illustrates the comparison of features with and without those extracted from the first two layers i.e. Data link layer and Physical layer. These were discussed in the Feature engineering section. It shows model evaluation using TimeSeriesSplit across 3 folds, focusing solely on the accuracy metric. It includes a table and a corresponding bar chart visualizing the accuracy scores of different model configurations. The accuracy scores for the two main models are based on “Ext. Feat” which means adding the features from the Physical layer and Data link layer.

Similarly, “No Ext Feat.” No such extra features are added to the dataset. The scores with and without features are as follows:

- CL_Model (Ext. Feat.): CL_Model using External Features. It achieved an accuracy of 1.0.
- CL_Model (No Ext Feat.): CL_Model without using External Features. It also achieved an accuracy of 1.0.
- RG_Model (Ext. Feat.): RG_Model using External Features. It achieved an accuracy of 1.0.

```
Model Evaluation with TimeSeriesSplit (3 Folds):
                     Model  Accuracy
0    CL_Model (Ext. Feat.)       1.0
1   CL_Model (No Ext Feat)       1.0
2     RG_Model (Ext. Feat)       1.0
3  RG_Model (No Ext Feat.)       1.0
```

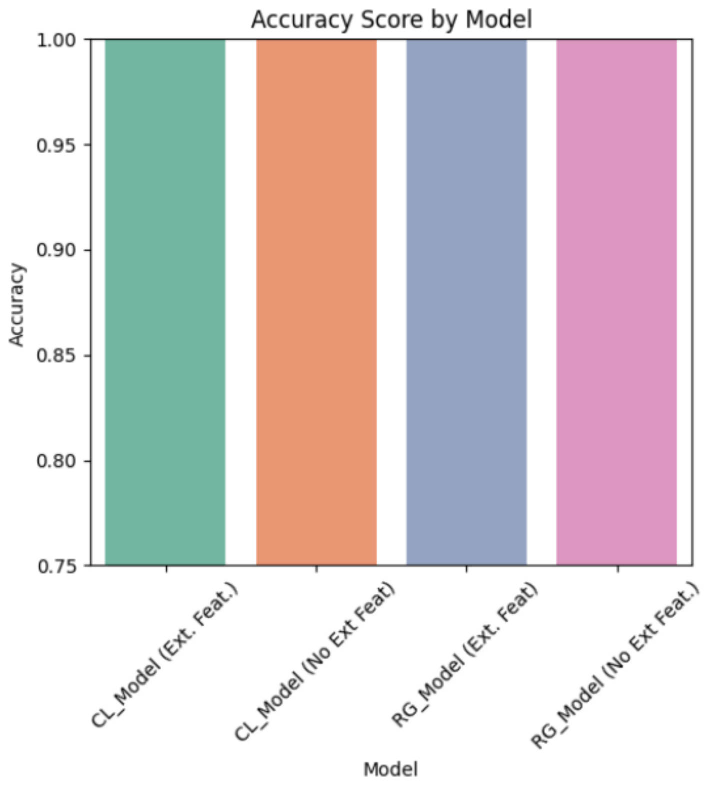


Figure 37 Comparison of features with and without physical layer Features

- RG_Model (No Ext Feat.): RG_Model without using External Features. It also achieved an accuracy of 1.0.

Figure 37 also indicates that all four model configurations achieved a perfect accuracy score of 1.0 across the 3 time series split folds. The bar chart visually confirms the results from the table, showing that all four model variations achieved a perfect accuracy score.

Summary: The evaluation results, based on accuracy using time series split across 3 folds, show that both the CL_Model and the RG_Model, regardless of whether they use external features or not, achieved a perfect accuracy score of 1.0. This suggests that all these model configurations performed exceptionally well on the evaluated data, correctly classifying all instances. From this, it is clear that the features generated through feature engineering do not impact the model performance. However, further we shall deeply analyse features that contribute to the model performance based on LIME and SHAP.

5.9 Model Explainability and Interpretation

5.9.1 LIME: Figure 38 displays Local Interpretable Model-agnostic Explanations (LIME) for five different samples (Sample 0, Sample 1, Sample 2, Sample 3, and Sample 4). For each sample, there's a horizontal bar chart showing the contribution of different features to the model's prediction for that specific instance.

- LIME Explanation: Each chart is labelled "LIME Explanation for Sample X (0 to 4)".
- Prediction and Ground Truth: Above each chart, it indicates the model's "Prediction" and the "Ground Truth" for that sample. In all five cases shown, the Prediction is 0 and the Ground Truth is also 0, meaning the model correctly classified these samples.
    - Feature Contributions: The horizontal bars represent the contribution of each feature to the prediction. Red bars indicate that the feature's value pushes the prediction towards the predicted class (in this case, class 0 (Normal)). The length of the bar signifies the magnitude of the contribution. Green bars indicate that the feature's value pushes the prediction away from the predicted class (towards the other class, presumably class 1 (Attack)).
- Feature Values and Conditions: Each bar is labeled with the feature name and the condition (e.g., "Error Type Count <= 0.91", "Length <= 0.15", "0.39 < Frame Number <= 0.63"). This indicates the value range of that feature for the specific sample being explained.

Observations of Samples:

- Error Type Count: This feature consistently has a strong negative contribution (red bar) across all five samples, suggesting that when "Error Type Count" is low (less than or equal to 0.91), it strongly supports the prediction of class 0.

- Length: The "Length" feature (<= 0.15) also shows a negative contribution (red bar) in most samples, indicating that shorter lengths tend to favor the prediction of class 0.
- Length_1: Similarly, "Length_1" (<= 0.14) also generally contributes negatively (red bar) towards the prediction of class 0.
- Frame Number: The condition "0.39 < Frame Number <= 0.63" often shows a positive contribution (green bar), suggesting that frame numbers within this range might push the prediction towards the other class (class 1).

Other Features: Features like "Payload Length", "Hop Limit", and "Inter-Frame Delay" appear in some explanations with relatively smaller contributions (both positive and negative)

Summary: The LIME explanations for these five correctly classified samples (as class 0) highlight that low values for "Error Type Count", "Length", and "Length_1" are strong indicators supporting this prediction. A "Frame Number" within a specific mid-range tends to have the opposite effect, pushing the prediction towards the other class. The other features shown have varying and generally smaller influences on the individual predictions. This provides insight into which feature values were most influential in the model's decision for these specific instances.

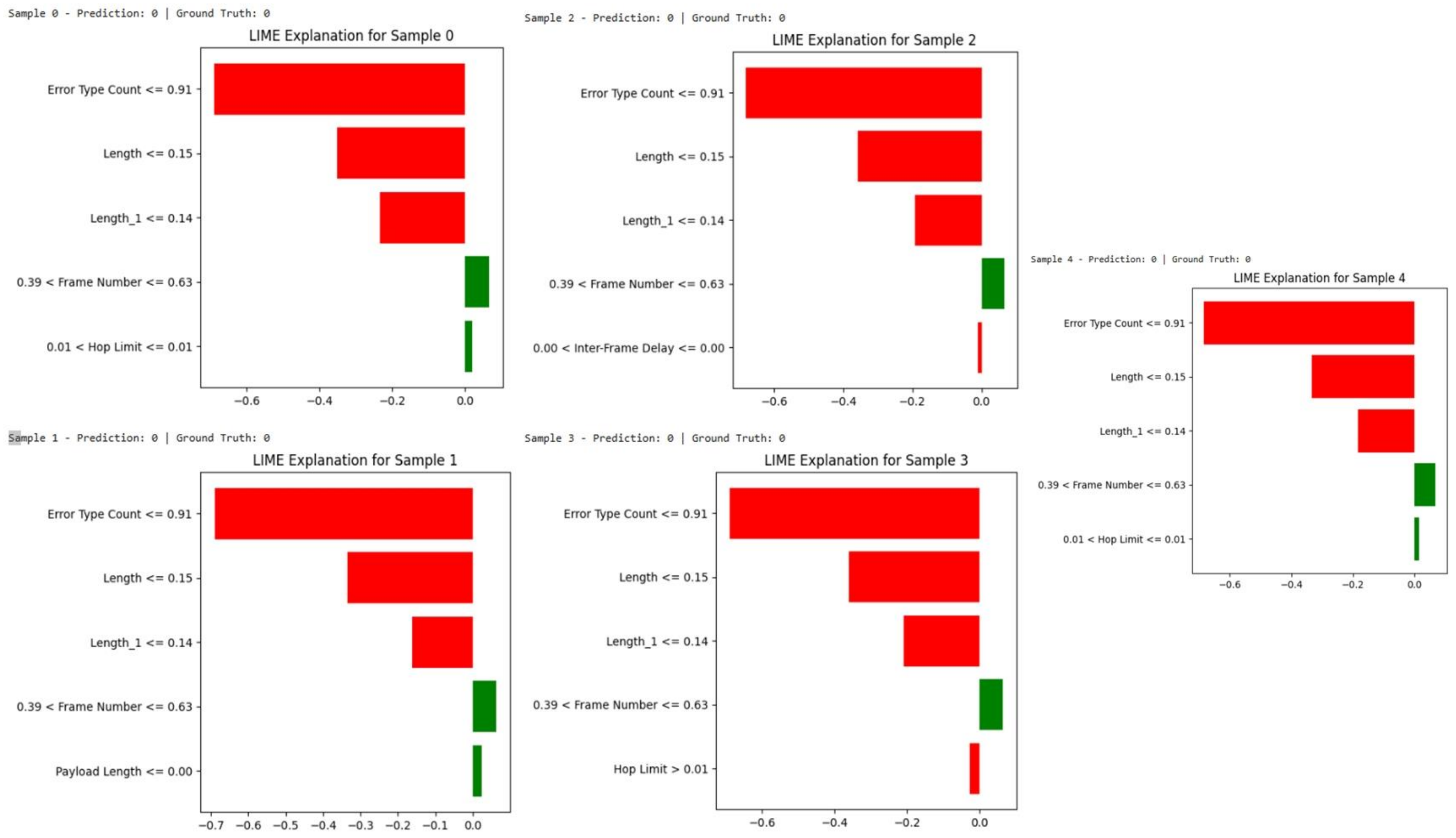


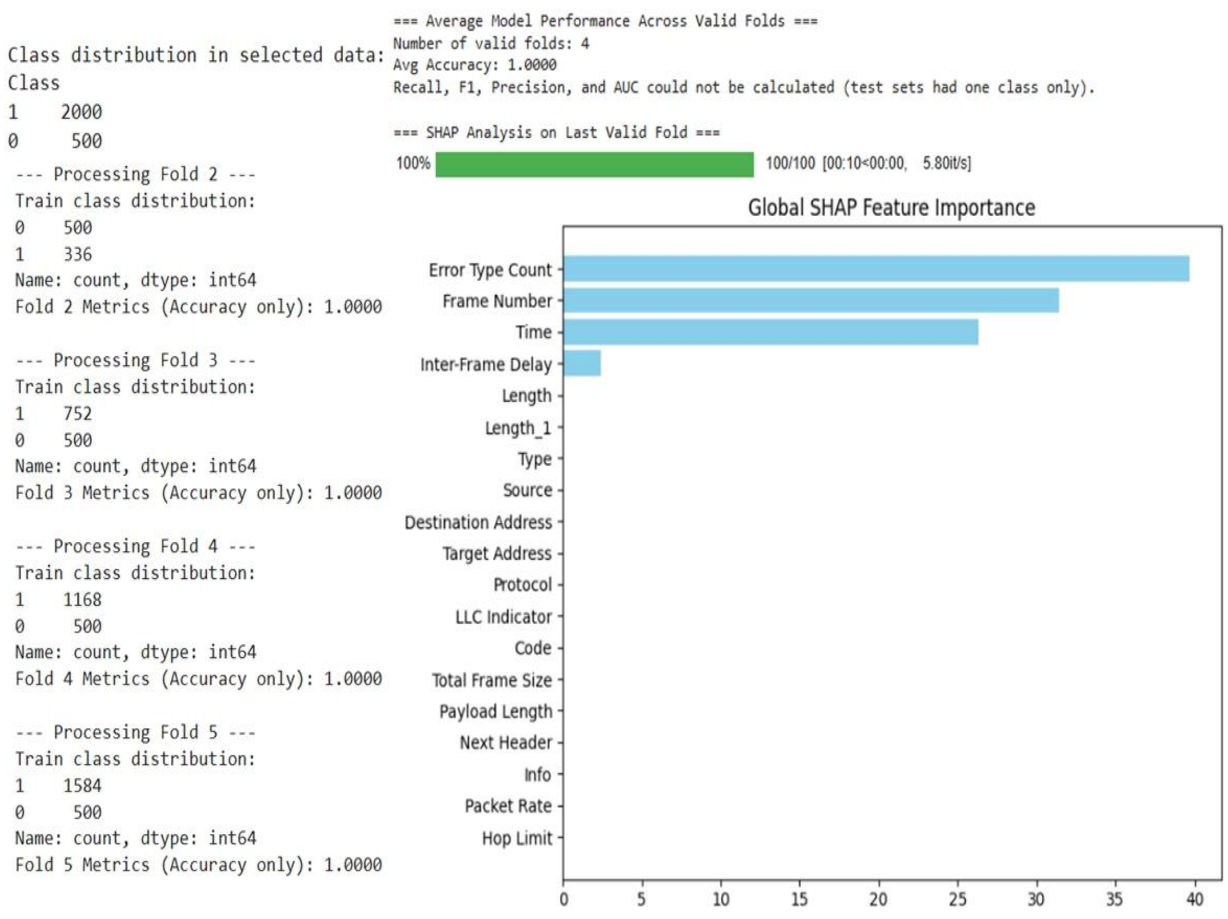
Figure 38 Lime

5.9.2 SHAP:

The figure 39 presents an analysis of a classification model's performance and feature importance using SHAP (Shapley Additive exPlanations).

The Left Side of the graph is Cross-Validation and Class Distribution

- Average Model Performance Across Valid Folds: The model was trained and evaluated using 4 valid folds (from a cross-validation procedure). The average accuracy across these folds is reported as 1.0000, indicating perfect classification performance on the validation sets. It also notes that Recall, Precision, and AUC could not be calculated, likely because the test sets in some or all folds contained only one class.
- Class Distribution in Selected Data: The initial class distribution in the selected data shows 2000 instances of class 1 and 500 instances of class 0. This indicates an imbalanced dataset with significantly more instances of class 1.
- Processing Fold 2 to Fold 5: This section details the class distribution in the training data for folds 2 through 5. In each of these folds, class 1 respectively), while class 0 consistently has 500 instances. The accuracy on each of these folds is reported as 1.0000.
- Right Side: SHAP Analysis on Last Valid Fold - Global Feature Importance
- SHAP Analysis on Last Valid Fold: This indicates that the SHAP analysis was performed on the predictions from the last (fourth) valid fold.
- Global SHAP Feature Importance: The horizontal bar chart displays the global feature importance as determined by SHAP values. The length of each bar represents the average absolute SHAP value for that feature across all instances in the last validation fold. This indicates the overall impact of each feature on the model's predictions.



```
Class distribution in selected data:
Class
1    2000
0     500
 --- Processing Fold 2 ---
 Train class distribution:
 0    500
 1    336
 Name: count, dtype: int64
 Fold 2 Metrics (Accuracy only): 1.0000

 --- Processing Fold 3 ---
 Train class distribution:
 1    752
 0    500
 Name: count, dtype: int64
 Fold 3 Metrics (Accuracy only): 1.0000

 --- Processing Fold 4 ---
 Train class distribution:
 1    1168
 0     500
 Name: count, dtype: int64
 Fold 4 Metrics (Accuracy only): 1.0000

 --- Processing Fold 5 ---
 Train class distribution:
 1    1584
 0     500
 Name: count, dtype: int64
 Fold 5 Metrics (Accuracy only): 1.0000
```

```
=== Average Model Performance Across Valid Folds ===
Number of valid folds: 4
Avg Accuracy: 1.0000
Recall, F1, Precision, and AUC could not be calculated (test sets had one class only).

=== SHAP Analysis on Last Valid Fold ===
100%|██████████| 100/100 [00:10<00:00,  5.80it/s]
```

Figure 39 SHAP

- Key Feature Importance:
  - Error Type Count is by far the most important feature, with a significantly higher SHAP value compared to others.
  - Frame Number is the second most important feature.
  - Time and Inter-Frame Delay also show relatively high importance.
  - Length and Length_1 have moderate importance.
  - The remaining features (Type, Source, Destination Address, Target Address, Protocol, LLC Indicator, Code, Total Frame Size, Payload Length, Next Header, Info, Packet Rate, Hop Limit) have considerably lower global importance according to the SHAP analysis.

Summary:

The model achieved perfect accuracy (1.0000) across all validation folds, although standard classification metrics like Recall, Precision, and AUC could not be computed due to the nature of the test sets. The SHAP analysis on the last validation fold reveals that Error Type Count is the most crucial feature influencing the model's predictions globally, followed by Frame Number, Time, and Inter-Frame Delay. Other features have a less significant overall impact on the model's output. The initial dataset shows an imbalance in class distribution.

5.9.3 Efficiency Metrics:

Figure 40 depicts bar chart comparing the inference times of three different machine learning models and AAD.

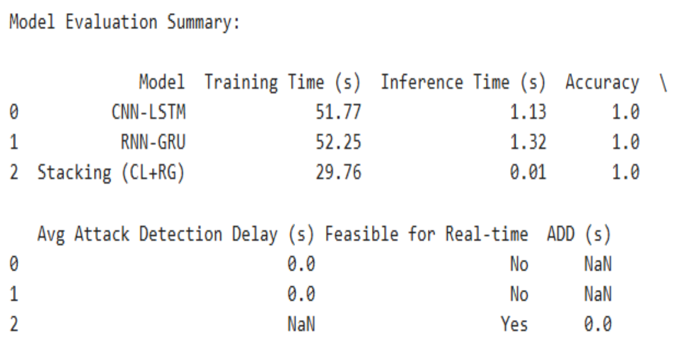
The table provides evaluation metrics for three models: CNN-LSTM, RNN-GRU, and Stacking (CL+RG). The metrics reported are:

- Training Time (s): The time taken to train each model. CNN-LSTM took 51.77 seconds, RNN-GRU took 52.25 seconds, and Stacking (CL+RG) took 29.76 seconds.
- Inference Time (s): The time taken for each model to make a prediction. CNN-LSTM had an inference time of 1.13 seconds, RNN-GRU had 1.32 seconds, and Stacking (CL+RG) had 0.01 seconds.
    - Accuracy: The classification accuracy of each model. All three models achieved a perfect accuracy of 1.0.

The figure also provides Avg Attack Detection Delay and AAD:

- This relates to the delay in detecting attacks and whether it's feasible for real-time application.
- For CNN-LSTM (row 0), the average attack detection delay is 0.0 seconds, and it's marked as "No" for feasibility.
- For RNN-GRU (row 1), the average attack detection delay is 0.0 seconds, and it's marked as "No" for feasibility.

- For Stacking (CL+RG) (row 2), the average attack detection delay is NaN (Not a Number), and it's marked as "Yes" for feasibility with an AAD of 0.0 seconds.



```
Model Evaluation Summary:

              Model  Training Time (s)  Inference Time (s)  Accuracy  \
0          CNN-LSTM              51.77                1.13       1.0
1           RNN-GRU              52.25                1.32       1.0
2  Stacking (CL+RG)              29.76                0.01       1.0

   Avg Attack Detection Delay (s) Feasible for Real-time  ADD (s)
0                             0.0                     No      NaN
1                             0.0                     No      NaN
2                             NaN                    Yes      0.0
```

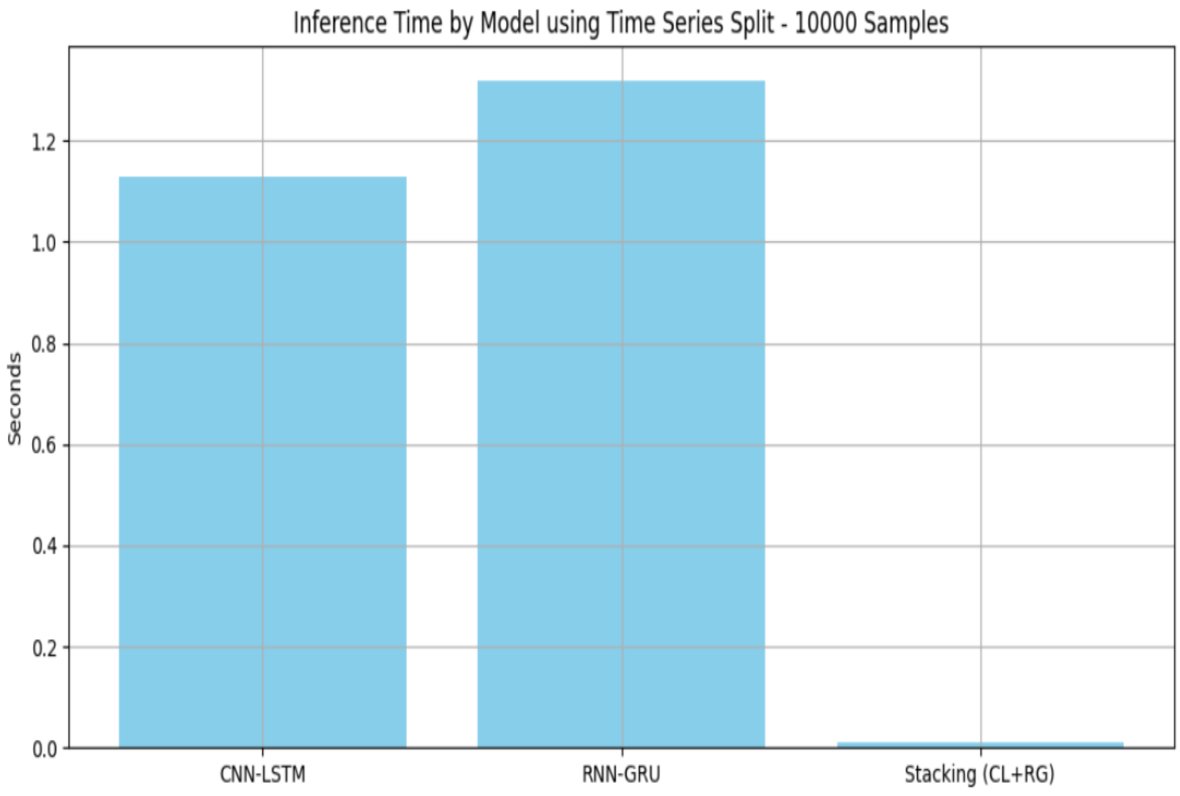


Figure 40 AAD

Inference Time by Model using Time Series Split - 10000 Samples (Bar Chart). . The bar chart visually compares the inference times of the three models, evaluated on a dataset of 10,000 samples using a time series split strategy.

- CNN-LSTM: Has an inference time of approximately 1.13 seconds (matching the table).
- RNN-GRU: Has an inference time of approximately 1.32 seconds (matching the table).
- Stacking (CL+RG): Has a very low inference time, close to 0 seconds (matching the table at 0.01 seconds, which is barely visible on this scale).

Summary: All three models (CNN-LSTM, RNN-GRU, and Stacking (CL+RG)) achieved perfect accuracy on the evaluated task. However, they differ significantly in their training and inference times. The Stacking (CL+RG) model has the lowest training time and a remarkably low inference time, making it potentially more suitable for real-time applications despite having a NaN value for average attack detection delay in the summary table. Both CNN-LSTM and RNN-GRU have considerably higher inference times. The feasibility of their attack detection delay for real-time applications is marked as "No," while Stacking (CL+RG) is marked as "Yes" despite the missing delay value. The bar chart clearly highlights the significant speed advantage of the Stacking (CL+RG) model during inference.

5.10 Summary of Experiments:

Table 16: The tables presented in the above table provide a comprehensive evaluation of various machine learning and deep learning techniques for the detection of DoS and DDoS attacks, highlighting the critical factors that influence their performance. Table 17 sets the stage by comparing the accuracy of different models from existing research, alongside the proposed CNN_LSTM and RNN-GRU models. This comparison underscores the substantial impact of the dataset on model performance. For instance, the CIC-IDS2017 dataset appears to be relatively "easier" to classify, with several models achieving high accuracy, while the CAIDA and DARPA datasets pose a greater challenge. Ensemble methods, as demonstrated by Dasari and Kalari (2024), achieve the highest accuracy in this table, indicating the power of combining multiple models. The proposed deep learning models also show strong potential.

Table 18 delves into the impact of feature engineering and the ADASYN technique on the proposed models' performance with IPv6 datasets. Feature engineering, which involves extracting additional relevant features from network traffic data, generally improves the accuracy of both CNN_LSTM and RNN-GRU. ADASYN is employed to address class imbalance, a common issue in network security datasets where normal traffic instances far outnumber attack instances. The comparable performance of CNN_LSTM and RNN-GRU in this table suggests the potential for further improvement through ensemble stacking. Table 18 expands the evaluation to five datasets and incorporates time series split validation and ensemble stacking. This table provides a more granular view of performance by including precision, recall, and F1-score, in addition to accuracy. The results further emphasize the dataset dependency, with the Mendeley dataset standing out as particularly easy to classify.

5.10.1 Overall comparison.

The Sain Malaysian dataset, on the other hand, proves to be challenging for both CNN_LSTM and RNN-GRU, highlighting the need for robust models capable of generalizing across diverse network conditions. Ensemble stacking generally leads to performance gains, demonstrating the benefits of combining the strengths of different models. Time series split validation is used to evaluate the model's ability to detect attacks in a realistic time-evolving scenario.

Summary: The tabulated results collectively demonstrate that while deep learning models like CNN_LSTM and RNN-GRU are effective for DDoS detection, their performance is significantly influenced by the characteristics of the network traffic data. Techniques like feature engineering, ADASYN, and ensemble stacking can further enhance detection capabilities. The choice of evaluation metrics and validation strategies, such as time series split, is also crucial for obtaining a comprehensive and realistic assessment of the model's effectiveness.

Table 20: Summary of the experiments and comparison

	Table 16: Comparison with State-of-the-Art	**Table 17: IPv6 Datasets with ADASYN & Feature Eng.**	**Table 18: Extended Evaluation with Stacking**
Focus	Comparative accuracy of various models	Impact of feature engineering and ADASYN	Detailed performance with time series split & stacking
Key Metric(s)	Accuracy	Accuracy	Accuracy, Precision, Recall, F1-Score
Best Performing Models/Techniques	Ensemble (XGBoost, etc.), CNN_LSTM	CNN_LSTM, RNN-GRU (high accuracy)	Ensemble Stacking (often improves)
Challenging Datasets	NSL_KDD	Sain Malaysian	Sain Malaysian, NSL_KDD
Impact of the Dataset	Significant influence on performance	Varies across datasets	Performance varies significantly by dataset
Feature Engineering	N/A	Improves performance	Used in most datasets, the impact varies
ADASYN	N/A	Improves handling of imbalance	Used, impact on precision/recall
Ensemble Stacking	N/A	Potential for improvement	Often enhances performance
Time Series Split	Train-test split	Train-test split	Time Series Split - Used for evaluation

5.10.2 Trade-offs observed across setups:

Trade-offs Observed Across Setups:

The evaluation of DDoS detection models involves navigating several trade-offs that significantly impact their performance, generalization, and practical applicability. The above discussions in this chapter highlight these trade-offs across various experimental setups, primarily concerning dataset selection, model complexity, feature engineering, class imbalance handling, and validation strategies.

i. Dataset Dependency vs. Model Generalization:

- A prominent trade-off lies in the dataset's influence on model performance. Models often achieve high accuracy on specific datasets (e.g., NSL_KDD/Mendeley), suggesting that the dataset's characteristics (e.g., traffic patterns, attack types, data quality) play a crucial role in a model's success.
- However, this can lead to a trade-off between achieving high accuracy on a particular dataset and the model's ability to generalize to unseen or diverse network traffic. For example, models performing well on Primary Datasets may struggle with Secondary datasets (Sain Malaysian/NSL_KDD), indicating a trade-off between dataset-specific optimization and broader applicability.

ii. Model Complexity vs. Performance and Computational Cost:

- The choice of model complexity presents another trade-off. Ensemble methods (e.g., XGBoost, Random Forest) and deep learning models (e.g., CNN_LSTM, RNN-GRU) often demonstrate strong performance. Ensemble methods, in particular, achieve high accuracy by combining multiple models, but this comes at the cost of increased computational complexity and training time.
- Deep learning models, while capable of learning complex patterns, also require substantial computational resources and may be harder to interpret compared to simpler machine learning algorithms. On the other hand,

simpler models like Naïve Bayes or KNN might be computationally efficient but may not capture intricate attack patterns, leading to a trade-off between computational cost and detection capability.

iii. Feature Engineering vs. Data Requirements:

- Feature engineering, the process of extracting relevant features from raw data, is shown to improve model performance. However, this introduces a trade-off: feature engineering requires domain expertise and can increase the complexity of the data preprocessing pipeline.
- Furthermore, engineered features might be specific to certain network environments, limiting the model's portability. The decision to invest in feature engineering involves balancing the potential performance gains with the added complexity and potential reduction in generalization.

iv. Class Imbalance Handling vs. Model Bias:

- Network traffic datasets often suffer from class imbalance, where normal traffic instances are far more frequent than attack instances. This imbalance can bias models towards the majority class, leading to poor detection of rare attacks.
- Techniques like ADASYN are used to address this by generating synthetic samples of the minority class. While ADASYN can improve the detection of attacks (increased recall), it might also introduce noise or overlap, potentially decreasing precision (increased false positives). Thus, there's a trade-off between improving attack detection and maintaining the accuracy of normal traffic classification.

v. Validation Strategy vs. Real-World Applicability:

- The choice of validation strategy influences how well model performance estimates reflect real-world performance. Traditional train-test splits might overestimate performance if the test set doesn't represent the temporal dynamics of network traffic.
- Time series split validation is employed to address this, providing a more realistic evaluation by respecting the temporal order of data. However, time series split validation can be more computationally intensive and might

require larger datasets. This represents a trade-off between evaluation accuracy and computational feasibility.

In conclusion, the development of effective DDoS detection systems necessitates careful consideration of these trade-offs. The optimal setup depends on the specific requirements of the deployment environment, including the acceptable level of computational cost, the need for real-time detection, the characteristics of the network traffic, and the importance of accurately detecting rare attack types.

5.10.3 Impact of Added Physical Features :

In the section Feature Engineering and Inclusion of Physical Feature, the utilization of specific features in the experiments discussed the results of proposed models on IPv6 datasets, incorporating with and without feature engineering. This implies that additional features beyond the basic network layer information were included. Considering the context of network traffic analysis for attack detection, "physical features" potentially refer to characteristics derived from the physical layer or data link layer of the network communication. These include features like Inter Frame Delay and Packet rate at the physical layer; and Error Type Count & LLC Indicator information at the data link layer. The fact that the results in Table 15 show promising performance for the proposed models on IPv6 datasets after feature engineering suggests that the inclusion of these added physical features contributed positively to the accuracy of the DDoS detection. By providing the models with information beyond just the network layer headers and protocols, these features also contributed to the models to identify subtle anomalies or patterns indicative of malicious activity at a lower level of the network stack. For instance, a high error type count or unusual patterns in frame sizes or link layer control information are strong indicators of an attack.

5.10.4 Windowing:

In the Section Window Analysis, it was discussed about the optimum window size being 10 and its results, focusing on F1 score and Accuracy. Time series splitting inherently involves processing data in temporal windows. The use of time series split implies that the models were trained and tested on sequential segments (windows) of the network traffic data.

The application of a windowing technique is crucial for capturing the temporal dependencies inherent in network traffic flows and attack patterns. DDoS attacks often manifest as a sustained surge of malicious traffic over a period. By processing data in windows, the models can learn to identify these temporal patterns and correlations that might be missed by analysing individual packets in isolation. For example, the rate of packet arrival, the consistency of inter-packet timings, or the persistence of certain header flags over a window of time can be significant indicators of an attack. The successful performance (perfect accuracy) of the models, as mentioned in the summary, further suggests that this windowing approach was effective in enabling the models to learn and detect attack patterns within the temporal context of the network traffic.

5.10.5 Impact of Interpretability:

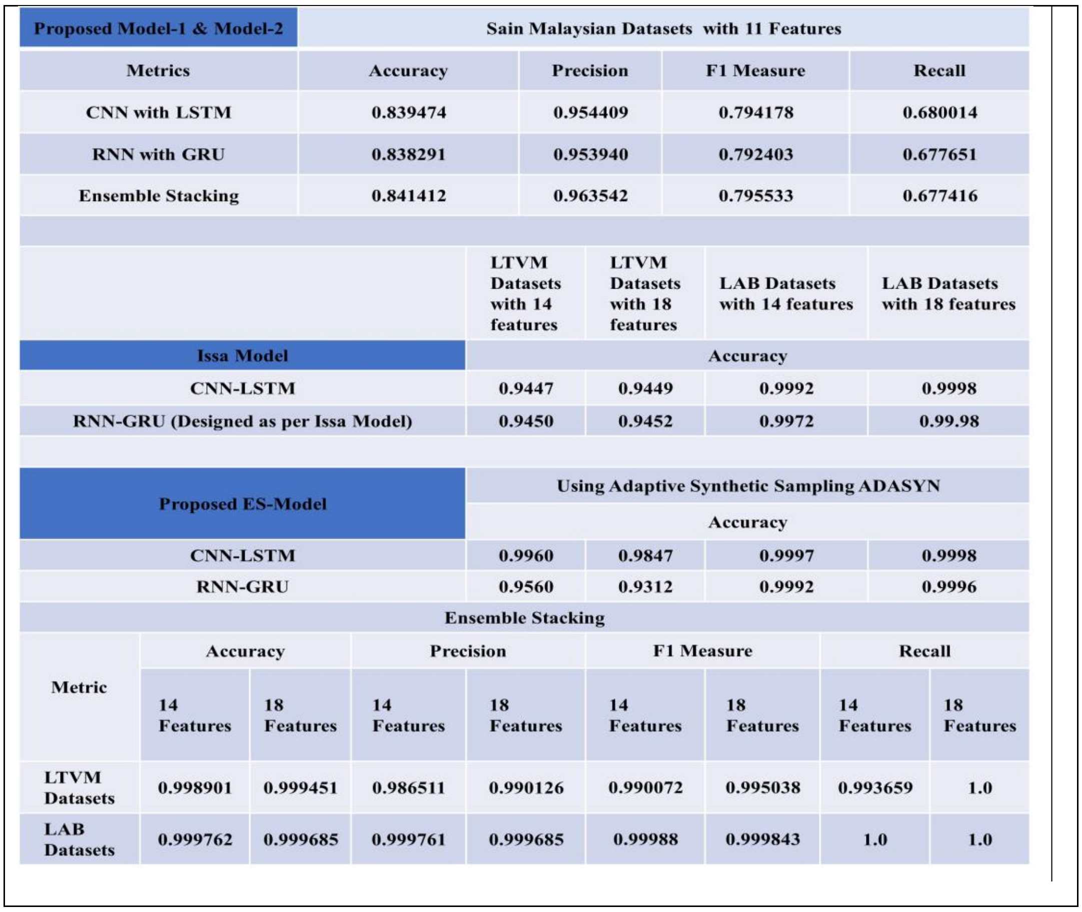
Section LIME explains that the five most contributory samples highlight values for "Error Type Count", "Length", and "Length_1" are strong indicators supporting the predictions. A "Frame Number" within a specific mid-range tends to have the opposite effect, pushing the prediction towards the other class. Similarly, the SHAP analysis on the last validation fold reveals that Error Type Count is the most crucial feature influencing the model's predictions globally, followed by Frame Number, Time, and Inter-Frame Delay. The inclusion of feature engineering, ADASYN to overcome imbalanced data and the use of time series split (applying windowing) have played a significant role in achieving the reported high accuracy in DDoS detection. In addition, the AAD and inference results clearly show that the base models are not suitable for real-time implementation. The proposed final Ensemble Stack Model is being recommended for real-time implementation.

5.11 Critical Evaluation

The above sections from 5.1 to 5.9 describe and discuss the model's robustness and its performance. Further experiments were done based on the train-test splits and applying Ahmet Issa‘s model architecture and the proposed model architecture using ADASYN and without ADSYN besides varying in features from 14 and 18.

Using the generated data sets, the “Ahmet Sardar Ahmed Issa” model and the proposed model are used with a primary emphasis on accuracy. The evaluation revealed that the proposed model outperformed the “Ahmet Sardar Ahmed Issa” model in terms of accuracy. Furthermore, additional metrics were assessed for the proposed model, showcasing notably high scores across various aspects, except for the Recall metric, which indicates room for improvement as it registered a lower value. A statistical tabular format for an easy understanding of the outstanding results obtained based on the data sets and the combination of Models can be viewed in Table 21.

Table 21: ES results of Issa and the Proposed model statistics using generated datasets with features and ADASYN.



Proposed Model-1 & Model-2	Sain Malaysian Datasets with 11 Features			
Metrics	Accuracy	Precision	F1 Measure	Recall
CNN with LSTM	0.839474	0.954409	0.794178	0.680014
RNN with GRU	0.838291	0.953940	0.792403	0.677651
Ensemble Stacking	0.841412	0.963542	0.795533	0.677416

	LTVM Datasets with 14 features	LTVM Datasets with 18 features	LAB Datasets with 14 features	LAB Datasets with 18 features
Issa Model	Accuracy			
CNN-LSTM	0.9447	0.9449	0.9992	0.9998
RNN-GRU (Designed as per Issa Model)	0.9450	0.9452	0.9972	0.99.98

Proposed ES-Model	Using Adaptive Synthetic Sampling ADASYN			
	Accuracy			
CNN-LSTM	0.9960	0.9847	0.9997	0.9998
RNN-GRU	0.9560	0.9312	0.9992	0.9996

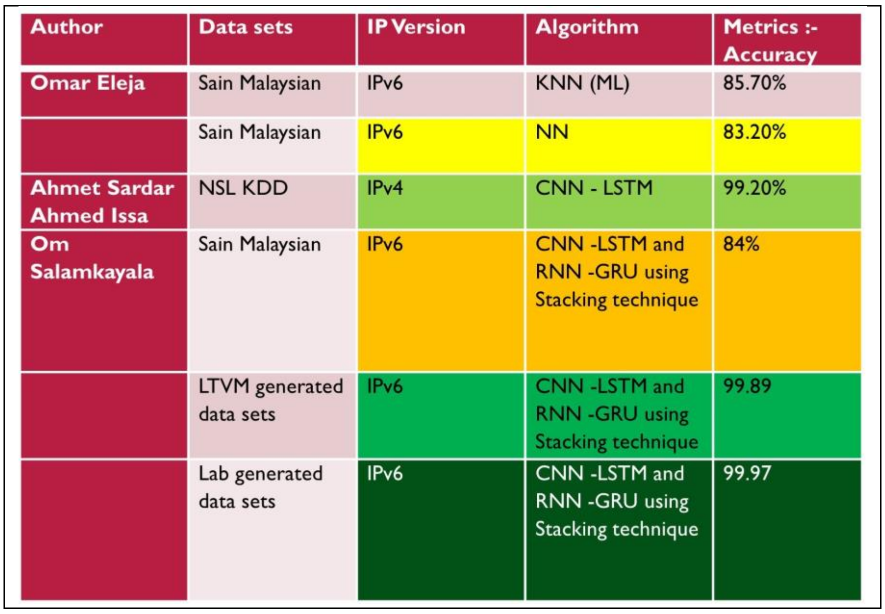
Ensemble Stacking								
Metric	Accuracy		Precision		F1 Measure		Recall	
	14 Features	18 Features	14 Features	18 Features	14 Features	18 Features	14 Features	18 Features
LTVM Datasets	0.998901	0.999451	0.986511	0.990126	0.990072	0.995038	0.993659	1.0
LAB Datasets	0.999762	0.999685	0.999761	0.999685	0.99988	0.999843	1.0	1.0

The statistics clearly show that experiments performed with different numbers of features and the ADASYN technique yield varied results. When using the Sain Malaysian datasets with the proposed model, the accuracy was 0.83%, and

Ensemble stacking did not enhance it significantly, resulting in an accuracy of 0.84%. However, with generated datasets, the results were significantly higher, achieving up to 0.94% with very little variation. This led to the decision to use Ensemble stacking, as both the Issa model with generated datasets and the proposed model with Sain Malaysian datasets showed similar differences in performance.

Table 22 illustrates the comparative results of experiments performed with combinations of the proposed model and the ”Ahmet Sardar Ahmed Issa” model using all the datasets.

Table 22: Results of Issa and Proposed model statistics using all 4 datasets

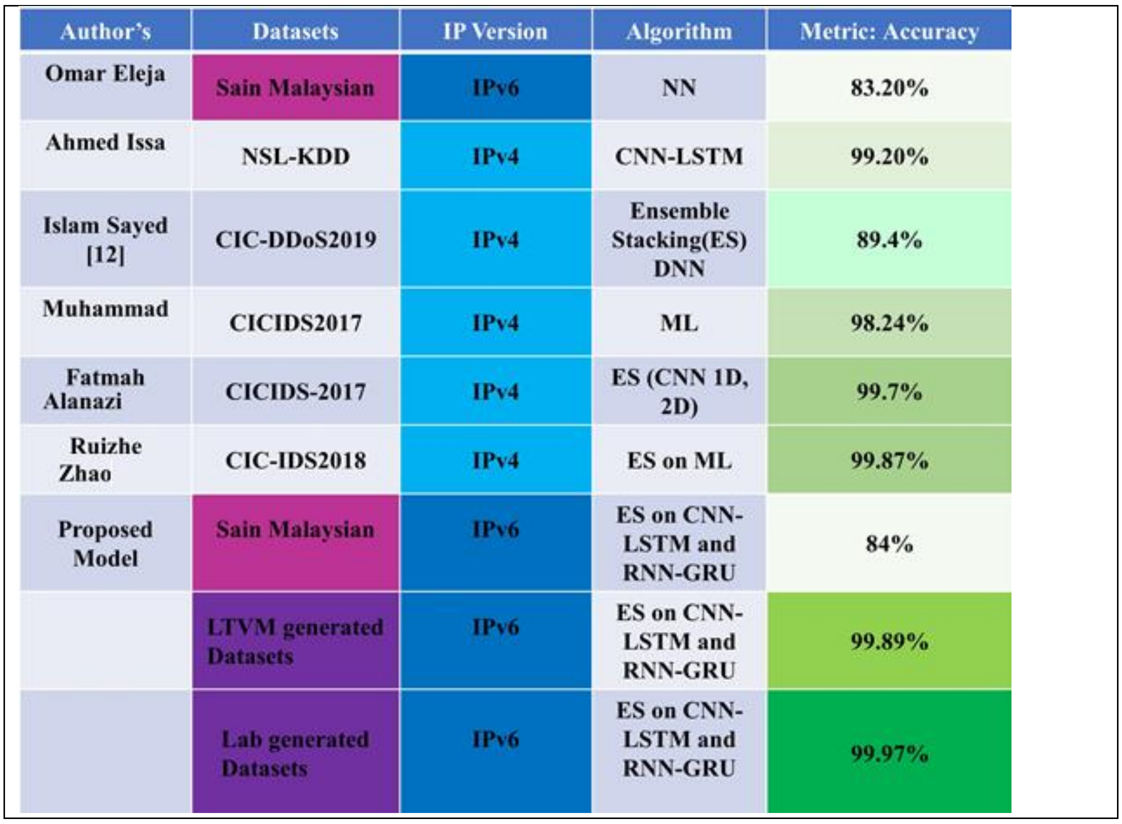


Author	Data sets	IP Version	Algorithm	Metrics :- Accuracy
Omar Eleja	Sain Malaysian	IPv6	KNN (ML)	85.70%
	Sain Malaysian	IPv6	NN	83.20%
Ahmet Sardar Ahmed Issa	NSL KDD	IPv4	CNN - LSTM	99.20%
Om Salamkayala	Sain Malaysian	IPv6	CNN -LSTM and RNN -GRU using Stacking technique	84%
	LTVM generated data sets	IPv6	CNN -LSTM and RNN -GRU using Stacking technique	99.89
	Lab generated data sets	IPv6	CNN -LSTM and RNN -GRU using Stacking technique	99.97

From Table 22 the fusion of Model 1, incorporating the CNN-LSTM architecture advocated by author Ahmet Sardar Ahmed Issa, has demonstrated its efficacy and robustness in both learning and detecting DDoS attacks. Similarly, the integration of Model 2 featuring RNN and GRU has yielded impressive outcomes. However, to emphasize the significance of these models and enhance performance metrics in DDoS attack detection and prediction, a Stacking ensemble technique has been employed, resulting in outstanding scores of up to 99.89% with LTVM datasets (primary dataset 1) and 99.97% with LAB datasets (primary dataset 2). This proves that the aim in Chapter 1 section 1.3 is successfully achieved by answering yes to the Hypothesis from Section 1.4.

The proposed model showcased its effectiveness in both learning and detecting ICMPv6 DDoS attacks by generating real-time datasets. According to the statistics presented in Table 18, the proposed model achieved an accuracy of 84.14%, surpassing Omar Eleja's result of 83.20% obtained using the Neural Network (NN) algorithm on his ICMPv6 datasets at Sain Malaysian University. The "Ahmet Sardar Ahmed Issa" Model, combining Convolutional Neural Network (CNN) with Long Short-Term Memory (LSTM), attained a 99.20% accuracy when trained on NSL-KDD IPv4 datasets. In comparison, the proposed model demonstrated even higher accuracies of 99.89% and 99.97% when trained on generated datasets, specifically ICMPv6. This answers the related questions from Chapter 1 section 1.5.

Table 23: Comparison of the proposed model with other researchers, with results



Author's	Datasets	IP Version	Algorithm	Metric: Accuracy
Omar Eleja	Sain Malaysian	IPv6	NN	83.20%
Ahmed Issa	NSL-KDD	IPv4	CNN-LSTM	99.20%
Islam Sayed [12]	CIC-DDoS2019	IPv4	Ensemble Stacking(ES) DNN	89.4%
Muhammad	CICIDS2017	IPv4	ML	98.24%
Fatmah Alanazi	CICIDS-2017	IPv4	ES (CNN 1D, 2D)	99.7%
Ruizhe Zhao	CIC-IDS2018	IPv4	ES on ML	99.87%
Proposed Model	Sain Malaysian	IPv6	ES on CNN-LSTM and RNN-GRU	84%
	LTVM generated Datasets	IPv6	ES on CNN-LSTM and RNN-GRU	99.89%
	Lab generated Datasets	IPv6	ES on CNN-LSTM and RNN-GRU	99.97%

This implementation specifically targeted ICMPv6 DDoS attack datasets. Furthermore, section 1.6 contributed a novel approach, leveraging a model that combines CNN with LSTM, RNN with GRU, and subsequently stacking both. The efficacy of this approach was assessed by comparing and evaluating the remarkable results against benchmark datasets such as NSL-KDD and Sain Malaysian, as well as against the model proposed by Ahmet Sardar Ahmed Issa. Additionally, an extra 2 contributions were made in section 1.6 by generating two ICMPv6 datasets in different environmental settings and publishing them on GitHub and Mendeley along with 3 papers in two different conferences. Hence, the outstanding results of the accuracy score proved that the proposed model is worthy of claiming that the Model is robust enough to detect and predict DDoS attacks.

Chapter 5 presents the experimental evaluation of the proposed model using the selected datasets. It discusses the results obtained, comparing key performance metrics to assess the model's effectiveness. The chapter includes a critical comparison with state-of-the-art models, highlighting strengths and limitations. It also explores various trade-offs and assesses the model's feasibility for real-world deployment based on requirements.

6 CONCLUSIONS

6.1 Introduction

This chapter provides the conclusion by summarizing the key reflections of the thesis, the results achieved, and how these fulfill the stated objectives, ultimately validating the contributions and further future research and some takeaways.

6.2 Summary of the thesis:

1. **Study Aim**: This study aims to detect ICMPv6 DDoS attacks using an innovative combination of techniques, fusing CNN with LSTM and RNN with GRU, and employing ensemble stacking to achieve superior accuracy, enabling early-stage mitigation (Chapter 1)
2. **ICMPv6 DDoS Attack Selection**: ICMPv6 DDoS echo-reply attacks were chosen due to their high vulnerability. The integrated combination approach is novel. A review of related academic research confirmed the scarcity of this approach in ICMPv6 DDoS attacks, with one author partially proposing a similar combination. This supports the validity of the approach for achieving superior accuracy (Chapter 2.
3. **Learning and Testing DDoS Attacks**: DDoS attacks were studied and learned through testing and analysing available attack methodologies (Chapter 1).
4. **Approach Analysis**: The study analysed traditional mechanisms, machine learning (ML), artificial intelligence (AI) methods, and the Onion Methodology to understand and counter DDoS attacks (Chapters 1 and 2).
5. **Dataset Analysis**: The required datasets, particularly those related to ICMPv6 DDoS attacks, were analysed due to their limitation of open/easy availability (Chapters 2 and 4).
6. **Dataset Creation and Validation**: ICMPv6 datasets were built in two distinct physical environments and validated to ensure suitability for experimentation (Chapter 4).
7. **Attack Scripting**: It was observed that scripts in Scapy could easily launch customized attacks regardless of infrastructure, demonstrating why Command and Control and script-related DDoS attacks are effective and often undetectable.

8. **Packet Crafting**: The study involved learning how to craft ICMPv6 packets, including modifying information to evade detection and launch attacks.
9. **Attack Testing**: Combined insights from 7 and 8 where successful ICMPv6 DDoS attacks were then crafted to achieve high speed without damaging the institute’s infrastructure (Chapter 4).
10. **Model Design and Development**: A model was designed and developed using Python on the Google Colab platform, and tested with existing secondary datasets of IPv4 and ICMPv6 (Chapter 5, sections 5.4 and 5.5).
11. **Initial Model Testing**: The model was initially tested using ICMPv6 datasets, yielding decent results. Voting classifiers were tried but did not achieve superior accuracy. To enhance performance, ensemble stacking was selected and employed on the proposed model, resulting in superior results (Chapter 5, sections 5.7)
12. **Data Splitting and Model Mechanism**: In all testing and implementation (Chapters 4 and 5), datasets were split into 80% for training and 20% for testing. The model mechanism is outlined below:
    a. Data points were categorized as "1" for attacks and "0" for normal.
    b. Outputs from base models were fed into a meta-model, which rigorously classified data points and repeated cross-validation three times. The first two times involved training, and the third time tested the model, producing the highest accuracy in classifying attacks and normal traffic.
    c. The model successfully captured and dropped attack packets while allowing normal packets into the system.
13. **Performance Metrics**: Accuracy was the main metric for measuring model performance, along with precision, F1 measure, and recall. All metrics showed high percentages, except for recall.

6.3 Contribution

1. From Chapter 1 section 1.7 Literature review was successfully accomplished and based on that, a model was proposed i.e., the combination of CNN with LSTM and RNN with GRU that determines the attack and predicts based on the Ensemble Stacking technique.

2. Due to the shortage of ICMPv6 datasets that are not openly available, two datasets were generated based on two distinct environments.
3. The performance of the designed model results was compared and validated with state-of-the-art to determine the best fit for the learning and detection of DDoS attacks.
4. 3 papers were published, Two at the CLMA conference in London. One at the ICMLC Conference in Japan via virtual presentation. Two primary datasets were published one in GitHub and the other in Mendeley.

6.4 Critical validation of the proposed Model based on comparison to state-of-the-art Researcher's

Chapter 5 presents experimental results demonstrating the robustness of the proposed model in detecting DDoS attacks, using techniques such as feature engineering, windowing, Time Series Split, and ADASYN (to address class imbalance between "Attack" and "Normal" instances). Five diverse datasets, each reflecting different traffic generation scenarios, were used. The model achieved performance ranging from 81.56% to 99.98%, with some cases reaching 100%.

The average attack detection results indicate that base classifiers alone are not suitable for real-time implementation. However, the ensemble stacking approach proves to be effective for real-time deployment. Table 19 **c**ompares dataset performance, highlighting that NSL-KDD and Sain Malaysian datasets posed greater challenges, suggesting reduced model performance on previously unseen or more complex data

Furthermore, compared to other researchers who suggested and implemented the Ensemble technique in their studies, this proposed combination of two learning models and deploying a meta-model on them has proven to be the best solution, particularly for ICMPv6 DDoS Echo request and Echo reply, attacks. This conclusion is based on the promising and significantly higher results, as illustrated in Table 20.

In section 1.4 of the first chapter, the hypothesis is explored, revealing that the integration of Multi DNN techniques has resulted in significantly higher scores, particularly emphasizing accuracy metrics, achieving superior scores. Moving on to section 1.5, the objectives were successfully accomplished, focusing on designing an efficient model and implementing the code in Python within Google Labs.

Additionally, an extra 2 contributions were made in section 1.6 by generating two ICMPv6 datasets in different environmental settings and publishing them on GitHub and Mendeley along with 3 papers in two different conferences. Hence, the outstanding results of the accuracy score proved that the proposed model is worthy to claim that the Model is robust enough to detect and predict DDoS attacks.

In Section 6.4, the contributions of the research are clearly explained, while Section 6.3 and the beginning of this section demonstrate that the objectives listed in Chapter 1, Section 1.6 have been met. This includes a critical evaluation of the proposed model against the state-of-the-art, achieving a superior score that is provided in Table 19. The Onion Methodology, incorporating both quantitative and qualitative techniques, was utilized in this research, as detailed in Chapter 1, Section 1.8.2, with the implementation method described in Section 1.9. This methodology proved to be quite successful and safe, allowing flexibility in choosing other techniques based on resource availability.

All studies and experiments were conducted in accordance with the university's policies and ethics guidelines. No personal data or related information was used or exploited in completing the research work. Utmost care was taken during the experiments to ensure the risk and safety of the infrastructure especially configuring and generating datasets.

6.5 Challenges

During the course of the program, two main challenges were encountered, particularly concerning dataset availability for ICMPv6, as discussed in Section 6.2. Initially, obtaining ICMPv6 datasets was not straightforward. We successfully acquired them through a request to Sain Malaysian University, which can be considered a qualitative method. To further test and validate our proposed model, we reached out to other existing research groups in various universities but did not receive fruitful responses. Consequently, we needed additional datasets, prompting us to generate our own primary datasets following a quantitative method.

Another challenge was with the model that initially combined CNN with LSTM and RNN with GRU. Both combinations showed similar performance with minor differences. To address this, we initially used a voting classifier to select between them based on a voting mechanism. However, the voting classifier consistently favoured one combination, leading to biased results. To overcome this, further research was conducted on enhancing techniques, and the voting classifier was subsequently replaced with the Stacking Ensemble technique, which yielded outstanding results without bias.

6.6 Learning Outcomes and Takeaways

This research demonstrated that by integrating a novel combination of deep neural networks and selected algorithms, ICMPv6 DDoS attacks can be mitigated at an early stage. The study explored how necessary datasets can be generated under various scenarios and environments. The research methods employed had a significant impact on the direction of the study, providing key turning points crucial for accomplishing the required tasks. This process has contributed to professional development by fostering the ability to generate innovative ideas that can be transformed into research projects. The skills developed through this research enable the acceptance of challenges and the implementation of similar projects in the future.

This program also provided an opportunity to learn multiple subjects in networking, AI, software development, management, and cybersecurity. Additionally, the experience honed the ability to submit papers to conferences, enhancing both communication and presentation skills. Teaching similar modules Like Digital Forensics, Mobile Forensics at level 7, CyberOps Security (MoD) at level 5, etc., further refined these skills and provided an opportunity to pursue related program like PgCHPE Fellow HEA to develop teaching capabilities that was funded by the Computer Science Department, University of Staffordshire and Managing Leadership Performance a funded module by EU social fund for enhancing leadership skills. The use of various tools for research, teaching, and presentations, such as Overleaf LaTeX, MS Office Word, PowerPoint, and Project, was also mastered during this research program.

6.7 Extension of Proposed Research

Current Performance Merits: The proposed ensemble stacking model demonstrates strong performance across diverse datasets, achieving detection accuracies ranging from 81.56% to 100%. Time Series Split validation and windowing (optimal size = 10) enhanced temporal learning, while feature engineering—especially with added physical and data link layer features—contributed to improved DDoS detection in IPv6 environments. SHAP and LIME interpretations further validated feature relevance.

Limitations: Despite high accuracy, models such as CNN_LSTM and RNN-GRU show reduced generalization on complex or unseen datasets like Sain Malaysian /NSL_KDD, indicating possible overfitting or dataset dependency. Recall scores remain relatively lower, and base models lack real-time feasibility due to high inference delay or model drift across distributions.

Future Research: In continuation, it should focus on improving generalization across unknown traffic patterns, enhancing recall, and exploring lightweight yet interpretable models suitable for real-time deployment. Investigating more adaptive feature engineering strategies and drift detection mechanisms will also support robust, scalable solutions.

Future it can also be extended to encompass other ICMPv6 attacks, such as DDoS attacks based on Neighbour Discovery, spoofing and Man-in-the-Middle attacks, alongside conventional IPv6 attack vectors. The existing primary datasets can be leveraged for exploring vulnerabilities in home network environments. The proposed model can be deployed on non-supervised learning particularly using the captured traffic in .pcap format of the DDoS attacks mostly may affect the DDoS attacks detection due to the learning aspect. Additionally, it could serve as a preliminary assessment for industrial domains like IoT DDoS attacks, Autonomous vehicular DDoS attacks, Drone DDoS attacks, etc. provided careful feature selection or similar key attributes aligning with their common values. Similarly, detection of DDoS attacks at a signal level using Amplitude shift keying, Frequency shift keying, and Time shift keying can be implied using the deployment of the proposed model where appropriate signal-related data sets are to be used for effective and efficient promising results. Further, the generated primary datasets are available on Mendeley and GitHub where similar kinds of research can be carried enabling easy access to the new researchers.

A similar mitigation proposal can be implemented at the edge router to enhance network security and protect the enterprise. By integrating the functionalities of an IDS or firewall, the solution becomes highly compatible and portable within the router. This implementation poses challenges, particularly in terms of physical resources, as it requires substantial memory, which can be addressed by using high-capacity DDR5 memory. Additionally, the computing cost may be high due to the training time of the deep neural networks, which necessitates the selection of very high-speed processors to mitigate this issue.

7 REFERENCES

Ahmad, Ijaz, Zhong Wan, and Ashfaq Ahmad (2023). “A big data analytics for DDOS attack detection using optimized ensemble framework in Internet of Things”. In: Internet of Things 23, p. 100825.

Ahmed, Md Rayhan et al. (2023). “Intrusion Detection System in Software-Defined Networks Using Machine Learning and Deep Learning Techniques–A Comprehensive Survey”. In: Authorea Preprints

Alghazzawi, Daniyal et al. (2021). “Efficient detection of DDoS attacks using a hybrid deep learning model with improved feature selection”. In: Applied Sciences 11.24, p. 11634.

Alghuraibawi, Adnan Hasan Bdair et al. (2021). “Detection of ICMPv6-based DDoS attacks using anomaly based intrusion detection system: A comprehensive review”. In: International Journal of Electrical and Computer Engineering 11.6, p. 5216.

Alguliyev, Rasim M, Ramiz M Aliguliyev, and Fargana J Abdullayeva (2019). “Deep learning method for prediction of DDoS attacks on social media”. In: Advances in Data Science and Adaptive Analysis 11.01n02, p. 1950002.

Alharbi, Afnan and Khalid Alsubhi (2021). “Botnet detection approach using graph-based machine learning”. In: IEEE Access 9, pp. 99166–99180.

Alharbi, Yasser et al. (2021). “Denial-of-Service Attack Detection over IPv6 Network Based on KNN Algorithm”. In: Wireless Communications and Mobile Computing 2021.1, p. 8000869.

Ali, Muhammad et al. (2023). “Effective network intrusion detection using stackingbased ensemble approach”. In: International Journal of Information Security 22.6, pp. 1781–1798.

Alshra’a, Abdullah Soliman, Ahmad Farhat, and Jochen Seitz (2021). “Deep Learning Algorithms for Detecting Denial of Service Attacks in SoftwareDefined Networks”. In: Procedia Computer Science 191, pp. 254–263.

Alturki, Ryan (2021). “Research onion for smart IoT-enabled mobile applications”. In: Scientific Programming 2021, pp. 1–9.

An, Yi et al. (2019). “Deep learning enabled superfast and accurate M 2 evaluation for fiber beams”. In: Optics Express 27.13, pp. 18683–18694.

Apuke, OD (2017). Quantitative Research Methods: A Synopsis Approach. Kuwait Chapter of Arabian Journal of Business and Management Review, 6 (11), 40–47.

Asad, Muhammad et al. (2020). “Deepdetect: detection of distributed denial of service attacks using deep learning”. In: The Computer Journal 63.7, pp. 983–994.

Assis, Marcos VO et al. (2021). “A GRU deep learning system against attacks in software defined networks”. In: Journal of Network and Computer Applications 177, p. 102942.

Aydın, Hakan, Zeynep Orman, and Muhammed Ali Aydın (2022). “A long short term memory (LSTM)-based distributed denial of service (DDoS) detection and defense system design in public cloud network environment”. In: Computers & Security 118, p. 102725.

Bahashwan, Abdullah Ahmed, Mohammed Anbar, and Sabri M Hanshi (2020). “Overview of IPv6 based DDoS and DoS attacks detection mechanisms”. In: Advances in Cyber Security: First International Conference, ACeS 2019, Penang, Malaysia, July 30–August 1, 2019, Revised Selected Papers 1. Springer, pp. 153–167.

Bdair, Adnan Hasan et al. (2020). “Brief of intrusion detection systems in detecting ICMPv6 attacks”. In: Computational Science and Technology: 6th ICCST 2019, Kota Kinabalu, Malaysia, 29-30 August 2019. Springer, pp. 199–213.

Bhaya, Wesam and Mehdi EbadyManaa (2017). “DDoS attack detection approach using an efficient cluster analysis in large data scale”. In: 2017 Annual Conference on New Trends in Information & Communications Technology Applications (NTICT). IEEE, pp. 168–173.

Biondi, Philippe (2008-2024). “Scapy Introduction”. In: Scapy community 8ed86476, p. 1. url: https://scapy.readthedocs.io/en/latest/introduction.html.

CAIDA (2022). “CAIDA 2018-2022 Program Plan”. In: Caida.org 2018-2022, p. 7. url: https://www.caida.org/about/progplan/progplan2018/.

Callegari, Christian, Stefano Giordano, and Michele Pagano (2024). “A Real Time Deep Learning based Approach for Detecting Network Attacks”. In: Big Data Research, p. 100446.

Chen, Jinyin et al. (2019). “DAD-MCNN: DDoS attack detection via multi-channel CNN”. In: Proceedings of the 2019 11th International Conference on Machine Learning and Computing, pp. 484–488.

Chen, Yen-Hung et al. (2020). “Detecting linking flooding attacks using deep convolution network”. In: Proceedings of the 2020 the 3rd International Conference on Computers in Management and Business, pp. 70–74.

Cheng, Jieren et al. (2019). “A novel DDoS attack detection method using optimized generalized multiple kernel learning”. In: arXiv preprint arXiv:1906.08204.

Chicco, Davide, Matthijs J Warrens, and Giuseppe Jurman (2021). “The Matthews correlation coefficient (MCC) is more informative than Cohen’s Kappa and Brier score in binary classification assessment”. In: IEEE Access 9, pp. 78368–78381.

CIC, Jeffrey J. Henderson (Chair) (2021). “Canadian Institute for Cybersecurity”. In: University of New Brunswick, Jeffrey J. Henderson (Chair), unb.ca

0, p. 4. url: https://www.unb.ca/cic/about/index.html.

Cil, Abdullah Emir, Kazim Yildiz, and Ali Buldu (2021). “Detection of DDoS attacks with feed forward based deep neural network model”. In: Expert Systems with Applications 169, p. 114520.

CISCO (2006). “IPv6 Extension Headers Review and Considerations.” In: CISCOWhite Paper, pp. 1–12.

Cloudflare (2024). “DDoS Attack Trends for 2024 Q1”. In: The Cloudflare Blog Report 3, p. 18. url: https://radar.cloudflare.com/reports/ddos-

2024-q1.

D’hooge, Laurens et al. (2020). “Inter-dataset generalization strength of supervised machine learning methods for intrusion detection”. In: Journal of Information Security and Applications 54, p. 102564.

Dahiya, Amrita and Brij B Gupta (2021). “A reputation score policy and Bayesian game theory based incentivized mechanism for DDoS attacks mitigation and cyber defense”. In: Future Generation Computer Systems 117, pp. 193–204.

DARPA, MIT Lincoln Laboratory (July 2000). “2000 DARPA INTRUSION DETECTION SCENARIO SPECIFIC DATASETS”. In: consensus from the Wisconsin Re-think meeting and the July 2000 Hawaii PI meeting. 0, p. 3. url: https://www.ll.mit.edu/r-d/datasets/2000-darpaintrusion-detection-scenario-specific-datasets.

Davies, Elwyn B. and J´anos Moh´acsi (May 2007). Recommendations for Filtering ICMPv6 Messages in Firewalls. RFC 4890. doi: 10 . 17487 / RFC4890. url: https://www.rfc-editor.org/info/rfc4890

Deering, Dr. Steve E. and Alex Conta (Dec. 1998). Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 2463. doi: 10.17487/RFC2463. url:

https://www.rfceditor.org/info/rfc2463%20;%20https://www.rfc-editor.org/rfc/pdfrfc/rfc2463.txt.pdf.

Deering, Dr. Steve E. and Bob Hinden (Feb. 2006). IP Version 6 Addressing Architecture. RFC 4291. doi: 10.17487/RFC4291. url: https://www.rfc-editor.org/info/rfc4291.

Dong, Shi and Mudar Sarem (2019). “DDoS attack detection method based on improved KNN with the degree of DDoS attack in software-defined networks”. In: IEEE Access 8, pp. 5039–5048.

Droms, Ralph (Aug. 2014). IPv6 Multicast Address Scopes. RFC 7346. doi: 10.17487/RFC7346. url: https://www.rfc-editor.org/info/rfc7346

Elejla, Omar E et al. (2019). “Comparison of classification algorithms on ICMPv6based DDoS attacks detection”. In: Computational Science and Technology: 5th ICCST 2018, Kota Kinabalu, Malaysia, 29-30 August 2018. Springer, pp. 347–357.

Elubeyd, Hani and Derya Yiltas-Kaplan (2023). “Hybrid Deep Learning Approach for Automatic DoS/DDoS Attacks Detection in Software-Defined Networks”. In: Applied Sciences 13.6, p. 3828.

Gaurav, Akshat, Brij B Gupta, and Prabin Kumar Panigrahi (2022). “A novel approach for DDoS attacks detection in COVID-19 scenario for small entrepreneurs”. In: Technological Forecasting and Social Change 177, p. 121554.

Gont, Fernando and Will (Shucheng) LIU (Aug. 2022). Recommendations on the Filtering of IPv6 Packets Containing IPv6 Extension Headers at Transit Routers. RFC 9288. doi: 10.17487/RFC9288. url: https://www.rfc-editor.org/info/rfc9288.

Goundar, Sam (2012). “Chapter 3-Research Methodology and Research Method”. In: Cloud Computing. Research Gate Publications.

Hasan Kabla, Arkan Hammoodi et al. (2023). “Machine and deep learning techniques for detecting internet protocol version six attacks: a review.” In: International Journal of Electrical & Computer Engineering (2088-8708)13.5.

Holkoviˇc, Martin, Ondˇrej Ryˇsavy`, and Jindˇrich Dudek (2019). “Automating network security analysis at packet-level by using rule-based engine”. In: Proceedings of the 6th Conference on the Engineering of Computer Based Systems, pp. 1–8.

Housman Oxicusa Gugi Isnaini Hafida Sumadi, Fauzi Dwi Setiawan (2020). “SDN:DDOS ICMP,TCP,UDP”. In: Mendeley Data, V1 1.1, p. 1. url: https://data.mendeley.com/datasets/hkjbp67rsc/1.

Hwang, Ren-Hung et al. (2020). “An unsupervised deep learning model for early network traffic anomaly detection”. In: IEEE Access 8, pp. 30387–30399.

Ieracitano, Cosimo et al. (2020). “A novel statistical analysis and autoencoder driven intelligent intrusion detection approach”. In: Neurocomputing 387, pp. 51–62.

Internet Control Message Protocol (Sept. 1981). RFC 792. doi: 10.17487/RFC0792. url: https://www.rfc-editor.org/info/rfc792.

Issa, AS Ahmed and Zafer Albayrak (2023). “Ddos attack intrusion detection system based on hybridization of cnn and lstm”. In: Acta Polytechnica Hungarica 20.2, pp. 105–123.

Jing, Xuyang et al. (2019). “Network traffic fusion and analysis against DDoS flooding attacks with a novel reversible sketch”. In: Information Fusion 51, pp. 100–113.

Kalutharage, Chathuranga Sampath et al. (2023). “Explainable AI-based DDOS attack identification method for IoT networks”. In: Computers 12.2, p. 32.

Kaur, Parneet, Manish Kumar, and Abhinav Bhandari (2017). “A review of detection approaches for distributed denial of service attacks”. In: Systems Science & Control Engineering 5.1, pp. 301–320.

KDDCup University of California, Irvine (October 28, 1999). “KDD Cup 1999 Data, Fifth International Conference on Knowledge Discovery and Data Mining.” In: The UCI KDD Archive Information and Computer Science University of California, Irvine 92697-3425, p. 1. url: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

Kumar, Krishan, Sunny Behal, et al. (2021). “Distributed Denial of Service Attack Detection using Deep Learning Approaches”. In: pp. 491–495.

Li, Yubing et al. (2022). “P4-NSAF: defending IPv6 networks against ICMPv6 DoS and DDoS attacks with P4”. In: ICC 2022-IEEE International Conference on Communications. IEEE, pp. 5005–5010.

Liang, Xiaoyu and Taieb Znati (2019). “An empirical study of intelligent approaches to DDoS detection in large scale networks”. In: 2019 International Conference on Computing, Networking and Communications (ICNC). IEEE, pp. 821–827.

Ma, Yongsen, Gang Zhou, and Shuangquan Wang (2019). “WiFi sensing with channel state information: A survey”. In: ACM Computing Surveys (CSUR) 52.3, pp. 1–36. url: https://dl-acm-org.ezproxy.staffs.ac.uk/ doi/pdf/10.1145/3310194%0A.

Malliga, Subramaniam, PS Nandhini, and Shanmuga Vadivel Kogilavani (2022). “A comprehensive review of deep learning techniques for the detection of (distributed) denial of service attacks”. In: Information Technology and Control 51.1, pp. 180–215.

Manjula, HT and Neha Mangla (2023). “An approach to on-stream DDoS blitz detection using machine learning algorithms”. In: Materials Today: Proceedings 80, pp. 3492–3499.

Mardiana, Siti (2020). “Modifying research onion for information systems research”. In: Solid State Technology 63.4, pp. 5304–5313.

Mateen, Hafsa and Malik Shahzad (2021). “Factors Effecting Businesses due to Distributed Denial of Service (DDoS) Attack”. In: 2021 International Conference on Innovative Computing (ICIC). IEEE, pp. 1–7.

MAWI Fontugne, Romain et al. (Dec. 2010). “MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking”. In.

Melnikovas, Aleksandras (2018). “Towards an explicit research methodology: Adapting research onion model for futures studies”. In: Journal of Futures Studies 23.2, pp. 29–44. url: https://jfsdigital.org/wp-content/ uploads/2019/01/03-Melnikovas-Onion-Research-Model.pdf.

Mishra, Nivedita and Sharnil Pandya (2021a). “Internet of things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review”. In: IEEE Access 9, pp. 59353–59377.

Mittal, Meenakshi, Krishan Kumar, and Sunny Behal (2023a). “Deep learning approaches for detecting DDoS attacks: A systematic review”. In: Soft computing 27.18, pp. 13039–13075.

Mohammed, Ammar and Rania Kora (2023). “A comprehensive review on ensemble deep learning: Opportunities and challenges”. In: Journal of King Saud University-Computer and Information Sciences 35.2, pp. 757–774.

Mohmand, Muhammad Ismail et al. (2022). “A machine learning-based classification and prediction technique for DDoS attacks”. In: IEEE Access 10, pp. 21443–21454.

Nazih, Waleed et al. (2020). “Countering ddos attacks in sip based voip networks using recurrent neural networks”. In: Sensors 20.20, p. 5875.

Neira, Anderson Bergamini de, Burak Kantarci, and Michele Nogueira (2023). “Distributed denial of service attack prediction: Challenges, open issues and opportunities”. In: Computer Networks 222, p. 109553.

Ojugo, Arnold and Andrew Okonji Eboka (2020). “An Empirical Evaluation On Comparative Machine Learning Techniques For Detection Of The Distributed Denial Of Service (DDoS) Attacks”. In: Journal of Applied Science, Engineering, Technology, and Education 2.1, pp. 18–27.

Omer, Kasim (2020). “An efficient and robust deep learning based network anomaly detection against distributed denial of service attacks”. In: Computer Networks 180, p. 107390.

Ordabayeva, GK et al. (2020). “A systematic review of transition from IPV4 To IPV6”. In: Proceedings of the 6th International Conference on Engineering & MIS 2020, pp. 1–15.

Pedregosa, F. et al. (2011). “Scikit-learn: Machine Learning in Python”. In: Journal of Machine Learning Research 12, pp. 2825–2830. url: https:// scikit-learn.org/stable/auto_examples/model_selection/plot_ roc.html.

Purushotham, Pidugu and Akkalakshmi Muddana (2024). “Classification of Cyberattack detection in Network Traffic using Machine learning techniques”. In: 2024 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI). Vol. 2. IEEE, pp. 1–6

Reyad, Mohamed, Amany M Sarhan, and Mohammad Arafa (2023). “A modified Adam algorithm for deep neural network optimization”. In: Neural Computing and Applications 35.23, pp. 17095–17112.

S.Sumathi, Suresh Rajappa and Lokesh (2021). “Advanced Decision Sciences Based on Deep Learning and Ensemble Learning Algorithms: A Practical Approach Using Python.” In: Computer Science and Technology and Applications 2, pp. 1–357.

Saad, Redhwan MA, Mohammed Anbar, and Selvakumar Manickam (2018). “Rulebased detection technique for ICMPv6 anomalous behaviour”. In: Neural Computing and Applications 30, pp. 3815–3824.

Salamkayala, Om (20 June 2024). “ICMPv6DDoS−Dataset”. In: Mendeley Datasets 0, p. 1. url: https://data.mendeley.com/datasets/g583tzgv5s/1.

Salih, Abdulrahman (2017). An adaptive approach to detecting behavioural covert channels in IPv6. Nottingham Trent University (United Kingdom).

Saqib, Iqra et al. (2023). “Comparison Of Different Firewalls Performance In A Virtual For Cloud Data Center”. In: Journal of Advancement in Computing 1.1, pp. 21–28.

Sayed, Moinul Islam et al. (2022). “A Multi-Classifier for DDoS Attacks Using Stacking Ensemble Deep Neural Network”. In: 2022 International Wireless Communications and Mobile Computing (IWCMC). IEEE, pp. 1125– 1130.

Steingartner, William, Darko Galinec, and Andrija Kozina (2021). “Threat defense: Cyber deception approach and education for resilience in hybrid threats model”. In: Symmetry 13.4, p. 597.

Stiawan, Deris et al. (2020). “CICIDS-2017 dataset feature analysis with information gain for anomaly detection”. In: IEEE Access 8, pp. 132911–132921.

Tajdini, Mostafa (2018). Developing an advanced IPv6 evasion attack detection framework. Liverpool John Moores University (United Kingdom).

Tan, Shuaishuai et al. (2022). “Sneaking Through Security: Mutating Live Network Traffic to Evade Learning-Based NIDS”. In: IEEE Transactions on Network and Service Management 19.3, pp. 2295–2308.

Tandon, Rajat et al. (2022). “AMON-SENSS: Scalable and Accurate Detection of Volumetric DDoS Attacks at ISPs”. In: GLOBECOM 2022-2022 IEEE Global Communications Conference. IEEE, pp. 3399–3404.

Tanenbaum, Andrew s and David J Wetherall (2010). “Computer Networks”. In.

Tavallaee, Mahbod et al. (2009). “A detailed analysis of the KDD CUP 99 data set”. In: 2009 IEEE symposium on computational intelligence for security and defense applications. Ieee, pp. 1–6.

Tayyab, Mohammad, Bahari Belaton, and Mohammed Anbar (2020). “ICMPv6Based DoS and DDoS Attacks Detection Using Machine Learning Techniques, Open Challenges, and Blockchain Applicability: A Review”. In: IEEE Access 8, pp. 170529–170547.

Vint, Cerf and Robert Kahn (1974). “A protocol for packet network interconnection”. In: IEEE Transactions of Communications 22.5, pp. 637–48

Wan, Weijie et al. (2022). “Using Deep Learning Neural Networks and Stacking Ensemble Learning to Predict CSI 300 Index”. In: 2022 9th International Conference on Digital Home (ICDH). IEEE, pp. 81–86.

Wang, Ning et al. (2022). “Manda: On adversarial example detection for network intrusion detection system”. In: IEEE Transactions on Dependable and Secure Computing 20.2, pp. 1139–1153.

Yang, Zhen et al. (2022a). “A systematic literature review of methods and datasets for anomaly-based network intrusion detection”. In: Computers & Security 116, p. 102675.

Zewdie, Temechu Girma and Anteneh Girma (2022). “An Evaluation framework for machine learning methods in detection of DoS and DDoS Intrusion”. In: 2022 International conference on artificial intelligence in information and communication (ICAIIC). IEEE, pp. 115–121.

Zhao, Ruizhe et al. (2022). “A hybrid intrusion detection system based on feature selection and weighted stacking classifier”. In: IEEE Access 10, pp. 71414–71426.

8 APPENDICES

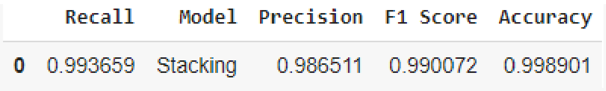
Appendix A: Implementation Code on Primary Datasets with highest score

```
# install required python repositories
pip install self

import numpy as np
import pandas as pd
from os import path

# importing required libraries for normalizing data
from sklearn import preprocessing
from sklearn.preprocessing import (StandardScaler, OrdinalEncoder,LabelEncoder, MinMaxScaler,
OneHotEncoder)
from sklearn.preprocessing import Normalizer, MaxAbsScaler , RobustScaler, PowerTransformer

# importing library for plotting
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import metrics
from sklearn.metrics import accuracy_score # for calculating accuracy of model
from sklearn.model_selection import train_test_split # for splitting the dataset for training and testing
from sklearn.metrics import classification_report # for generating a classification report of model

from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve, auc

import tensorflow as tf
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
from keras.layers import Dense, Conv1D, MaxPool1D, Flatten, Dropout # importing dense layer
from keras.models import Sequential #importing Sequential layer
from keras.layers import Input
from keras.models import Model
# representation of model layers
import keras.utils
from keras import utils as np_utils
import matplotlib.pyplot as plt
from imblearn.over_sampling import ADASYN
from collections import Counter
from tensorflow.keras.layers import Conv1D, MaxPooling1D, LSTM, GRU, Flatten
from sklearn.preprocessing import MinMaxScaler

# Selection of Datasets
```

```
df_train ='/content/scdtsets.csv' #Select the path for using Dataset 1
df_train ='/content/Labdataset.csv' #Select the path for using Dataset 2

train_data=pd.read_csv(df_train)
train_data.head()
train_data.info()
le=LabelEncoder()
clm=['Time','Source', 'Destination Address','Target
Address','Protocol','Type','ICMPv6srcLnklayerLength','Next Header','Frame Number','Info']
for x in clm:
   train_data[x]=le.fit_transform(train_data[x])

train_data.info()
train_data.describe().T
Y = train_data["Class"]
# Drop 'label' column
X = train_data.drop(labels = ["Class"],axis = 1)

Y = train_data['Class'] ## get output label
Y_i = Y.map({'Normal' : 0, 'Attack' : 1}) ## convert into label

## get all input data
X = train_data.drop(columns = 'Class')

X.shape, Y_i.shape

X_train, X_test, y_train, y_test = train_test_split(X, Y_i, test_size=0.2,
                                                    random_state=12,
                                                    stratify = Y_i)
## get shape
X_train.shape, X_test.shape, y_train.shape, y_test.shape

## plotting the class
g = sns.barplot(Y_i, palette="icefire")
plt.title(" Category \n Normal              Attack ")
Y_i.value_counts().plot.bar()
Y_i.value_counts()

## Using ADASYN
ada=ADASYN(sampling_strategy='minority',random_state=12)

X, Y_i = ada.fit_resample(X_train, y_train)

counter = Counter(Y_i)
print('After',counter)

g = sns.barplot(Y_i, palette="deep")
plt.title(" Category \n Normal              Attack")
Y_i.value_counts().plot.bar()
Y_i.value_counts()
```

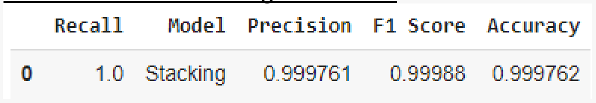
```
## scaling data
encoder = MinMaxScaler()
encoder.fit(X_train)

## transforming data into encoded form
X_train_enc = encoder.transform(X_train)
X_test_enc = encoder.transform(X_test)

X_train_enc.shape, X_test_enc.shape

learning_rate=0.00001
batch_size=5000
epochs = 5

model_save = ModelCheckpoint('./DDoS_ICMP.h5',
                             save_best_only = True,
                             save_weights_only = True,
                             monitor = 'val_loss',
                             mode = 'min', verbose = 1)
early_stop = EarlyStopping(monitor = 'val_loss', min_delta = 0.000001,
                           patience = 6, mode = 'min', verbose = 10,
                           restore_best_weights = True)
reduce_lr = ReduceLROnPlateau(monitor = 'val_loss', factor = 0.1,
                              patience = 6, min_delta = 0.000001,
                              mode = 'min', verbose = 1)

## Building CNN with LSTM Neural Network model
model.add(Conv1D(64,2,input_shape = (X_test.shape[1], 1)))
model.add(LSTM(100, input_shape=(11, 12), return_sequences=True))
model.add(MaxPooling1D(1))
model.add(Conv1D(32,2))
model.add(MaxPooling1D(1))
model.add(Conv1D(16,2))
model.add(MaxPooling1D(1))
model.add(Conv1D(8,2))
model.add(MaxPooling1D(1))
model.add(Flatten())
model.add(Dense(15, activation='relu'))
model.add(Dropout(rate=0.2))
model.add(Dense(10, activation='relu'))
model.add(Dropout(rate=0.2))
model.add(Dense(1,activation='sigmoid'))

## Model Compilation
model.compile(loss='binary_focal_crossentropy', optimizer='Adam',metrics=['accuracy'])


history = model.fit(X_train,
                    y_train,
                    batch_size=batch_size,
                    steps_per_epoch=X_train.shape[0] // batch_size,
                    epochs=epochs,
                    validation_data=(X_test,y_test),
                    callbacks = [model_save, early_stop, reduce_lr],)
                                    -------------------------------

## Building RNN with GRU Neural Network model
model = Sequential()
model.add(SimpleRNN(128,'relu',return_sequences=True, input_shape=(X.shape[1],1)))
model.add(GRU(64, return_sequences=True))
model.add(MaxPooling1D(1))
model.add(Flatten())
model.add(Dropout(0.1))
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(32,activation='relu'))
model.add(Flatten())
model.add(Dense(16, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(8, activation='softmax'))
model.add(Dropout(0.1))
model.add(Dense(1,activation=LeakyReLU(alpha=1)))

## Model Compilation

model.compile(loss='msle', optimizer='Adam', metrics=['accuracy'])

learning_rate=0.01
batch_size=5000
epochs = 5

model_save = ModelCheckpoint('./DDoS_ICMP.h5',
                             save_best_only = True,
                             save_weights_only = True,
                             monitor = 'val_loss',
                             mode = 'min', verbose = 1)
early_stop = EarlyStopping(monitor = 'val_loss', min_delta = 0.001,
                           patience = 6, mode = 'min', verbose = 1,
                           restore_best_weights = True)
reduce_lr = ReduceLROnPlateau(monitor = 'val_loss', factor = 0.6,
                              patience = 6, min_delta = 0.001,
                              mode = 'min', verbose = 1)
history = model.fit(X_train,
                    y_train,
                    batch_size=batch_size,
                    steps_per_epoch=X_train.shape[0] // batch_size,
                    epochs=epochs,
                    validation_data=(X_test,y_test),
                    callbacks = [model_save, early_stop, reduce_lr],)

## STACKING
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import Lasso
from sklearn.feature_selection import SelectFromModel
from sklearn.neural_network import MLPClassifier
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import StackingRegressor
from sklearn.linear_model import LassoCV
from sklearn.linear_model import RidgeCV
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn import preprocessing
from sklearn.metrics import mean_squared_log_error
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
import math
import time
pd.pandas.set_option('display.max_columns', None)
import warnings
warnings.simplefilter(action='ignore')
from self import self
from keras.activations import *
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import roc_auc_score
from statistics import stdev
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import RocCurveDisplay

## CNN FUNCTION (CNN with LSTM)
def CNN_model(self, batch_size=5000, epochs=5, metrics=['Accuracy']):
    model = Sequential()
    model.add(Conv1D(64,2,input_shape = (X_test.shape[1], 1)))
    model.add(LSTM(100, input_shape=(11, 12), return_sequences=True))
    model.add(MaxPooling1D(1))
    model.add(Conv1D(32,2))
    model.add(MaxPooling1D(1))
    model.add(Conv1D(16,2))
    model.add(MaxPooling1D(1))
    model.add(Conv1D(8,2))
    model.add(MaxPooling1D(1))
    model.add(Flatten())
    model.add(Dense(15, activation='relu'))
    model.add(Dropout(rate=0.2))
    model.add(Dense(10, activation='relu'))
    model.add(Dropout(rate=0.2))
    model.add(Dense(1,activation='sigmoid'))
    model.compile(loss='binary_focal_crossentropy', optimizer='Adam',metrics=['accuracy'])
    return model

## RNN FUNCTION (RNN and GRU)
def RNN_model(Self, batch_size=5000, epochs=5, metrics=['Accuracy']):
    model = Sequential()
    model.add(SimpleRNN(128,'relu',return_sequences=True, input_shape=(X.shape[1],1)))
    model.add(GRU(64, return_sequences=True))
    model.add(MaxPooling1D(1))
    model.add(Flatten())
    model.add(Dropout(0.1))
    model.add(Dense(64,activation='relu'))
    model.add(Dropout(0.1))
    model.add(Dense(32,activation='relu'))
    model.add(Flatten())
    model.add(Dense(16, activation='relu'))
    model.add(Dropout(0.1))
    model.add(Dense(8, activation='softmax'))
    model.add(Dropout(0.1))
    model.add(Dense(1,activation=LeakyReLU(alpha=1)))
    model.compile(loss='msle', optimizer='Adam', metrics=['accuracy'])
    return model

MLPClassifier(alpha=1, max_iter=5000) == CNN_model(self)

MLPRegressor(alpha=1, max_iter=5000) == RNN_model(self)

CNN = MLPClassifier()
RNN = MLPRegressor()
final_estimator=LogisticRegression()

estimator_list = [('CNN',MLPClassifier()),('RNN',MLPRegressor())]

stacking_regressor = StackingRegressor(cv=3, estimators=estimator_list,
final_estimator=LogisticRegression())

StackingRegressor(estimators=estimator_list, final_estimator=LogisticRegression())

estimator_list = [('CNN',MLPClassifier()),('RNN',MLPRegressor())]

# Build stack model
stack_model = StackingClassifier(cv=3,estimators=estimator_list, final_estimator=LogisticRegression())

stack_model.fit(X_train, y_train)

# Make predictions
y_train_pred = stack_model.predict(X_train)
y_test_pred = stack_model.predict(X_test)

stack_model_train_accuracy = accuracy_score(y_train, y_train_pred) # Calculate Accuracy
#stack_model_train_mcc = matthews_corrcoef(y_train, y_train_pred) # Calculate MCC
```

```
stack_model_train_rec = recall_score(y_train, y_train_pred) # Calculate Recall
stack_model_train_f1 = f1_score(y_train, y_train_pred, average='weighted') # Calculate F1-score
stack_model_train_prec = precision_score(y_train, y_train_pred)# Calculate Precision

stack_model_test_accuracy = accuracy_score(y_test, y_test_pred) # Calculate Accuracy
#stack_model_test_mcc = matthews_corrcoef(y_test, y_test_pred) # Calculate MCC
stack_model_test_rec = recall_score(y_test, y_test_pred) # Calculate Recall
stack_model_test_f1 = f1_score(y_test, y_test_pred, average='weighted') # Calculate F1-score
stack_model_test_prec = precision_score(y_test, y_test_pred)# Calculate Precision

print('Model performance for Training set')
print('- Accuracy: %s' % stack_model_train_accuracy)
#print('- MCC: %s' % stack_model_train_mcc)
print('- Rec Score: %s' % stack_model_train_rec)
print('- F1 score: %s' % stack_model_train_f1)
print('- Pre score: %s' % stack_model_train_prec)
print('----------------------------------')
print('Model performance for Test set')
print('- Accuracy: %s' % stack_model_test_accuracy)
#print('- MCC: %s' % stack_model_test_mcc)
print('- Rec- Score: %s' % stack_model_test_rec)
print('- F1 score: %s' % stack_model_test_f1)
print('- Pre score: %s' % stack_model_test_prec)

# Train Test results of the stacking classifier
from sklearn.ensemble import StackingClassifier
SC = StackingClassifier(estimators=estimator_list,final_estimator=LogisticRegression())
SC.fit(X_train, y_train)
y_pred = SC.predict(X_test)

print(f"\nStacking classifier training Accuracy: {SC.score(X_train, y_train):0.2f}")
print(f"Stacking classifier test Accuracy: {SC.score(X_test, y_test):0.2f}")

SC_Recall = recall_score(y_test, y_pred)
SC_Precision = precision_score(y_test, y_pred)
SC_f1 = f1_score(y_test, y_pred)
SC_accuracy = accuracy_score(y_test, y_pred)
SC_roc_auc = roc_auc_score(y_test, y_pred)

score = cross_val_score(SC, X_train, y_train, cv=3, scoring='recall', error_score="raise")
SC_cv_score = score.mean()
SC_cv_stdev = stdev(score)
print('Cross Validation Recall scores are: {}'.format(score))
print('Average Cross Validation Recall score: ', SC_cv_score)
print('Cross Validation Recall standard deviation: ', SC_cv_stdev)
ndf = [(SC_Recall, SC_Precision, SC_f1, SC_accuracy)]
SC_score = pd.DataFrame(data = ndf, columns=['Recall','Precision','F1 Score', 'Accuracy'])
SC_score.insert(1, 'Model', 'Stacking')
SC_score
#OUT PUT while using Dataset 1
```



	Recall	Model	Precision	F1 Score	Accuracy
0	0.993659	Stacking	0.986511	0.990072	0.998901

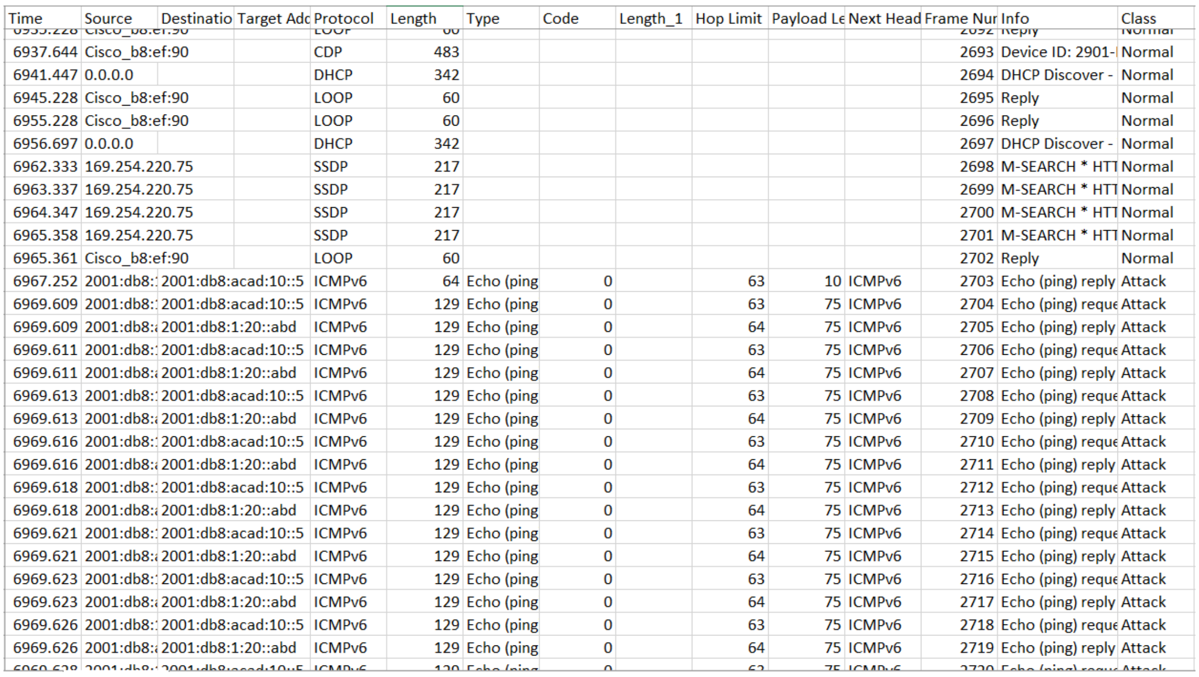
#OUT PUT while using Dataset 2



	Recall	Model	Precision	F1 Score	Accuracy
0	1.0	Stacking	0.999761	0.99988	0.999762

```
#Confusion Matrix
def confusion_matrix(actual, predicted, title):
    sns.heatmap(metrics.confusion_matrix(actual, predicted),
            cbar=False, annot=True, fmt='3g', cmap="Blues",
            annot_kws={"size": 16})
    plt.title("confusion matrix " + str(title), fontsize=25)
    plt.xlabel('predicted label', fontsize=20)
    plt.ylabel('true label', fontsize=20)
    return plt.show()
y_train_pred = stack_model.predict(X_train)
y_test_pred = stack_model.predict(X_test)
confusion_matrix(y_test, y_test_pred, 'cm')
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_test_pred)
print(cm)

# ROC curve
y_proba = SC.predict_proba(X_test)

def plot_auc_roc_curve(y_test, y_pred):
    fpr, tpr, _ = roc_curve(y_test, y_pred)
    roc_display = RocCurveDisplay(fpr=fpr, tpr=tpr).plot()
    roc_display.figure_.set_size_inches(5,5)
    plt.plot([0, 1], [0, 1], color = 'g')
plot_auc_roc_curve(y_test, y_proba[:, 1])
```

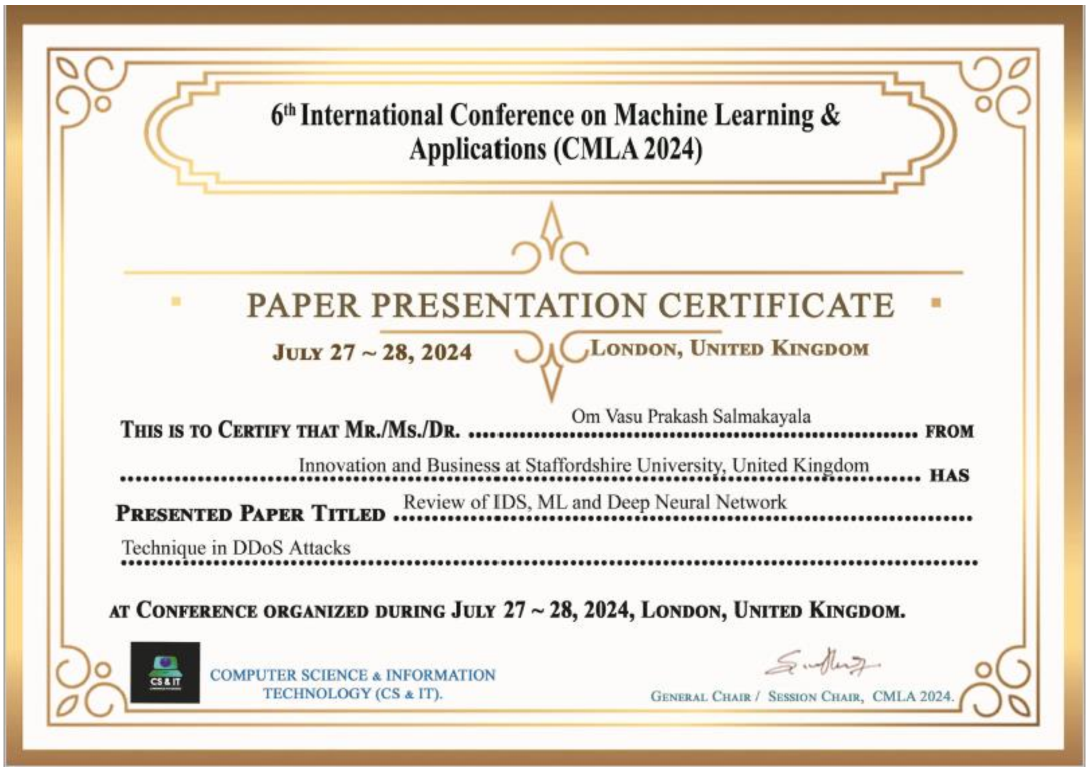
Appendix B: A sample of Dataset



Time	Source	Destinatio	Target Adc	Protocol	Length	Type	Code	Length_1	Hop Limit	Payload Le	Next Head	Frame Nur	Info	Class
6935.228	Cisco_b8:ef:90			LOOP	60							2692	Reply	Normal
6937.644	Cisco_b8:ef:90			CDP	483							2693	Device ID: 2901-	Normal
6941.447	0.0.0.0			DHCP	342							2694	DHCP Discover -	Normal
6945.228	Cisco_b8:ef:90			LOOP	60							2695	Reply	Normal
6955.228	Cisco_b8:ef:90			LOOP	60							2696	Reply	Normal
6956.697	0.0.0.0			DHCP	342							2697	DHCP Discover -	Normal
6962.333	169.254.220.75			SSDP	217							2698	M-SEARCH * HTT	Normal
6963.337	169.254.220.75			SSDP	217							2699	M-SEARCH * HTT	Normal
6964.347	169.254.220.75			SSDP	217							2700	M-SEARCH * HTT	Normal
6965.358	169.254.220.75			SSDP	217							2701	M-SEARCH * HTT	Normal
6965.361	Cisco_b8:ef:90			LOOP	60							2702	Reply	Normal
6967.252	2001:db8:	2001:db8:acad:10::5		ICMPv6	64	Echo (ping	0		63	10	ICMPv6	2703	Echo (ping) reply	Attack
6969.609	2001:db8:	2001:db8:acad:10::5		ICMPv6	129	Echo (ping	0		63	75	ICMPv6	2704	Echo (ping) reque	Attack
6969.609	2001:db8:	2001:db8:1:20::abd		ICMPv6	129	Echo (ping	0		64	75	ICMPv6	2705	Echo (ping) reply	Attack
6969.611	2001:db8:	2001:db8:acad:10::5		ICMPv6	129	Echo (ping	0		63	75	ICMPv6	2706	Echo (ping) reque	Attack
6969.611	2001:db8:	2001:db8:1:20::abd		ICMPv6	129	Echo (ping	0		64	75	ICMPv6	2707	Echo (ping) reply	Attack
6969.613	2001:db8:	2001:db8:acad:10::5		ICMPv6	129	Echo (ping	0		63	75	ICMPv6	2708	Echo (ping) reque	Attack
6969.613	2001:db8:	2001:db8:1:20::abd		ICMPv6	129	Echo (ping	0		64	75	ICMPv6	2709	Echo (ping) reply	Attack
6969.616	2001:db8:	2001:db8:acad:10::5		ICMPv6	129	Echo (ping	0		63	75	ICMPv6	2710	Echo (ping) reque	Attack
6969.616	2001:db8:	2001:db8:1:20::abd		ICMPv6	129	Echo (ping	0		64	75	ICMPv6	2711	Echo (ping) reply	Attack
6969.618	2001:db8:	2001:db8:acad:10::5		ICMPv6	129	Echo (ping	0		63	75	ICMPv6	2712	Echo (ping) reque	Attack
6969.618	2001:db8:	2001:db8:1:20::abd		ICMPv6	129	Echo (ping	0		64	75	ICMPv6	2713	Echo (ping) reply	Attack
6969.621	2001:db8:	2001:db8:acad:10::5		ICMPv6	129	Echo (ping	0		63	75	ICMPv6	2714	Echo (ping) reque	Attack
6969.621	2001:db8:	2001:db8:1:20::abd		ICMPv6	129	Echo (ping	0		64	75	ICMPv6	2715	Echo (ping) reply	Attack
6969.623	2001:db8:	2001:db8:acad:10::5		ICMPv6	129	Echo (ping	0		63	75	ICMPv6	2716	Echo (ping) reque	Attack
6969.623	2001:db8:	2001:db8:1:20::abd		ICMPv6	129	Echo (ping	0		64	75	ICMPv6	2717	Echo (ping) reply	Attack
6969.626	2001:db8:	2001:db8:acad:10::5		ICMPv6	129	Echo (ping	0		63	75	ICMPv6	2718	Echo (ping) reque	Attack
6969.626	2001:db8:	2001:db8:1:20::abd		ICMPv6	129	Echo (ping	0		64	75	ICMPv6	2719	Echo (ping) reply	Attack
6969.628	2001:db8:	2001:db8:acad:10::5		ICMPv6	129	Echo (ping	0		63	75	ICMPv6	2720	Echo (ping) reque	Attack

Total 100130 rows in Dataset 1
Total 1048575 rows in Dataset 2

Appendix C: Certificates related to Paper Presentation in Conferences



6th International Conference on Machine Learning & Applications (CMLA 2024)

PAPER PRESENTATION CERTIFICATE

JULY 27 ~ 28, 2024 LONDON, UNITED KINGDOM

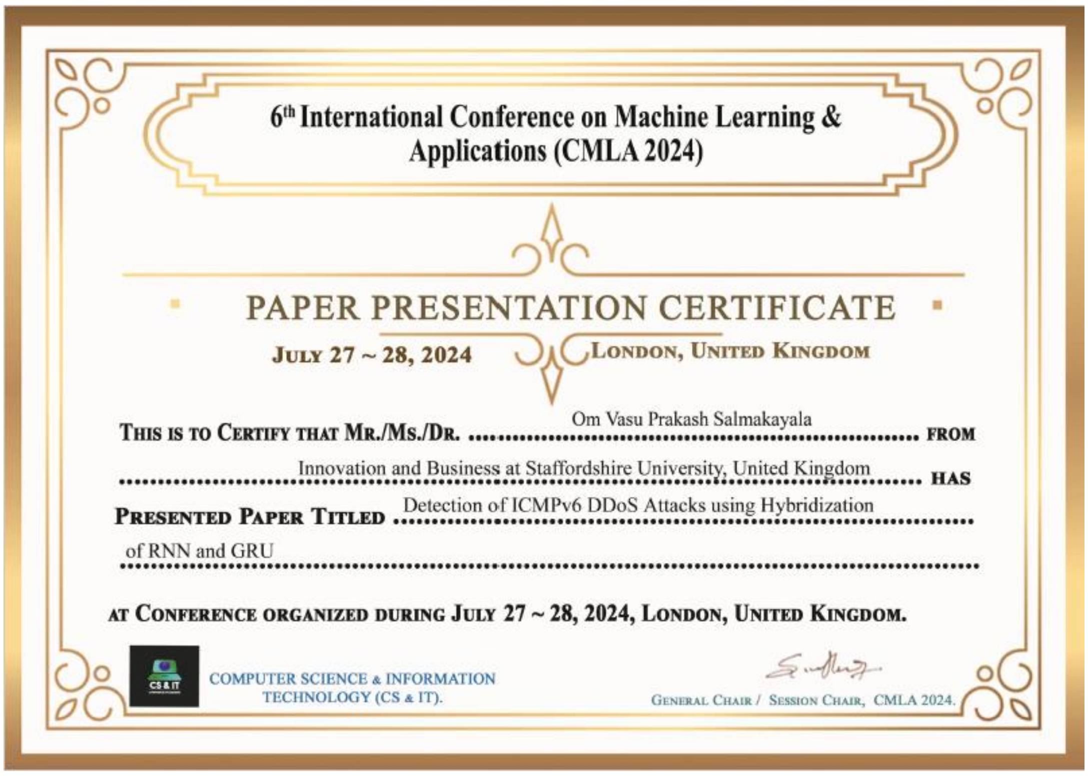
THIS IS TO CERTIFY THAT MR./MS./DR. Om Vasu Prakash Salmakayala FROM Innovation and Business at Staffordshire University, United Kingdom HAS PRESENTED PAPER TITLED Review of IDS, ML and Deep Neural Network Technique in DDoS Attacks

AT CONFERENCE ORGANIZED DURING JULY 27 ~ 28, 2024, LONDON, UNITED KINGDOM.

COMPUTER SCIENCE & INFORMATION TECHNOLOGY (CS & IT).

[illegible]

GENERAL CHAIR / SESSION CHAIR, CMLA 2024.



6th International Conference on Machine Learning & Applications (CMLA 2024)

PAPER PRESENTATION CERTIFICATE

JULY 27 ~ 28, 2024 LONDON, UNITED KINGDOM

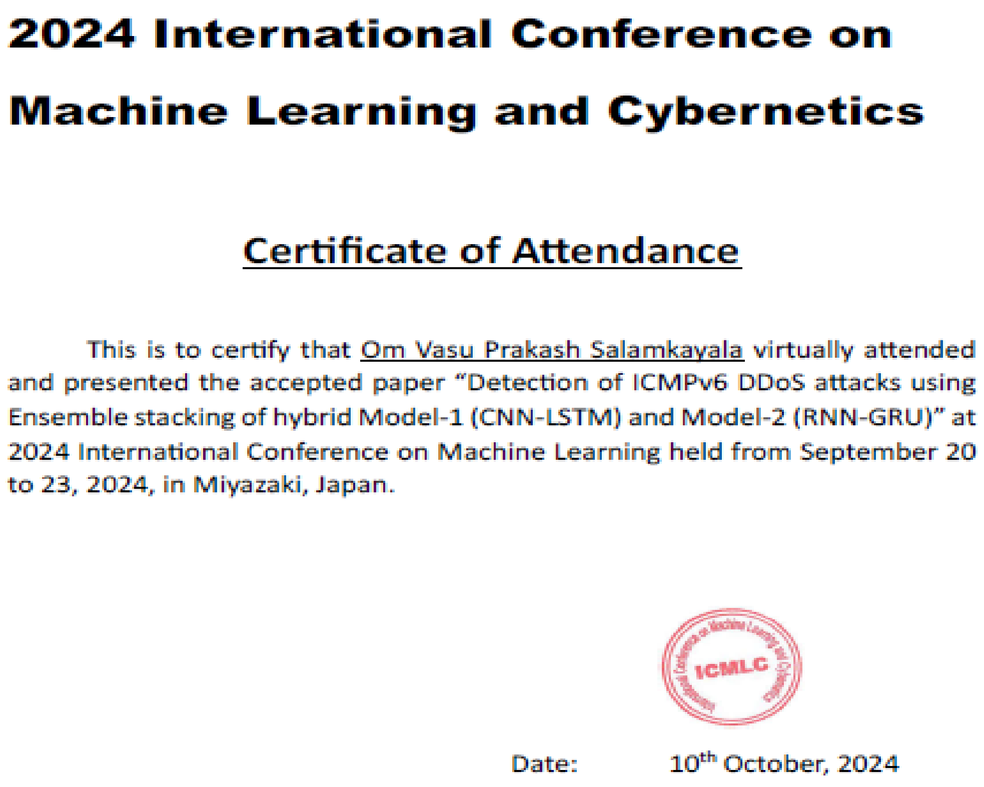
THIS IS TO CERTIFY THAT MR./MS./DR. Om Vasu Prakash Salmakayala FROM Innovation and Business at Staffordshire University, United Kingdom HAS PRESENTED PAPER TITLED Detection of ICMPv6 DDoS Attacks using Hybridization of RNN and GRU

AT CONFERENCE ORGANIZED DURING JULY 27 ~ 28, 2024, LONDON, UNITED KINGDOM.

COMPUTER SCIENCE & INFORMATION TECHNOLOGY (CS & IT).

[illegible]

GENERAL CHAIR / SESSION CHAIR, CMLA 2024.



2024 International Conference on Machine Learning and Cybernetics

Certificate of Attendance

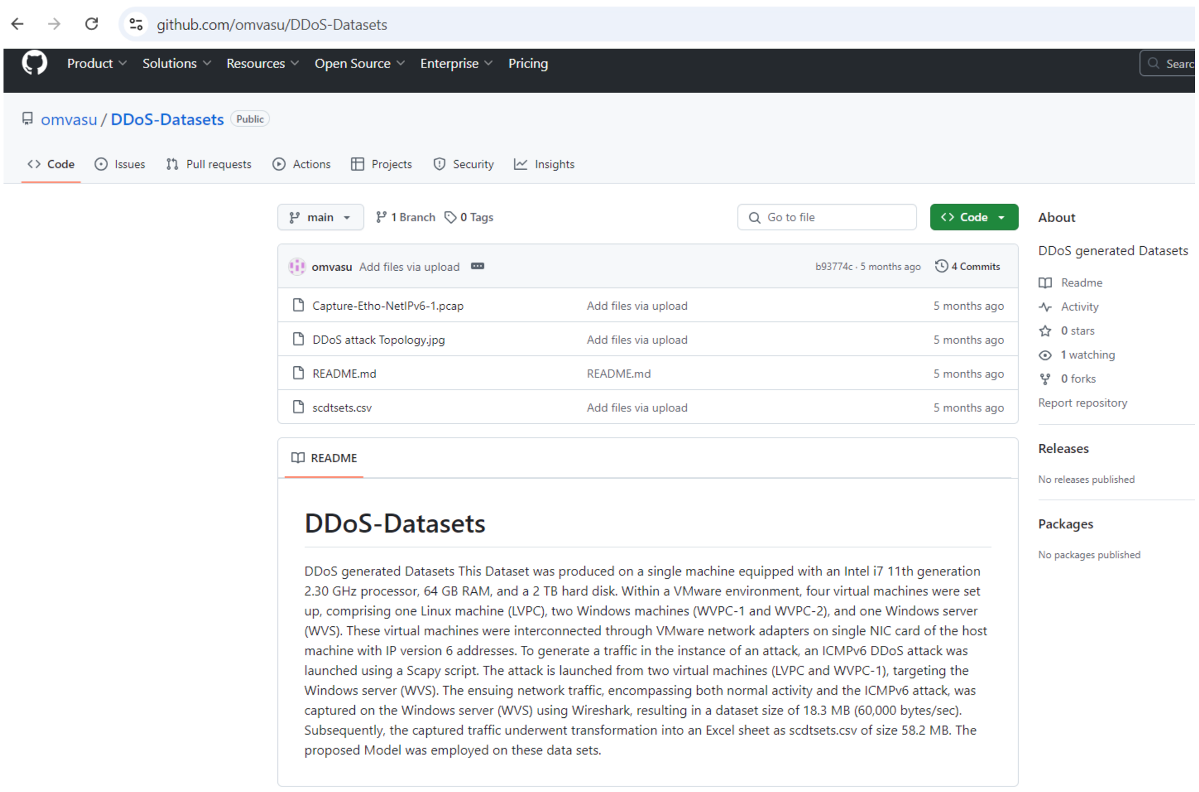
This is to certify that Om Vasu Prakash Salamkayala virtually attended and presented the accepted paper "Detection of ICMPv6 DDoS attacks using Ensemble stacking of hybrid Model-1 (CNN-LSTM) and Model-2 (RNN-GRU)" at 2024 International Conference on Machine Learning held from September 20 to 23, 2024, in Miyazaki, Japan.

ICMLC

Date: 10th October, 2024

Datasets:

GitHub :



github.com/omvasu/DDoS-Datasets

omvasu / DDoS-Datasets Public

main · 1 Branch · 0 Tags

omvasu Add files via upload · b93774c · 5 months ago · 4 Commits

Capture-Etho-NetIPv6-1.pcap	Add files via upload	5 months ago
DDoS attack Topology.jpg	Add files via upload	5 months ago
README.md	README.md	5 months ago
scdtsets.csv	Add files via upload	5 months ago

About

DDoS generated Datasets

Readme · Activity · 0 stars · 1 watching · 0 forks

Releases: No releases published

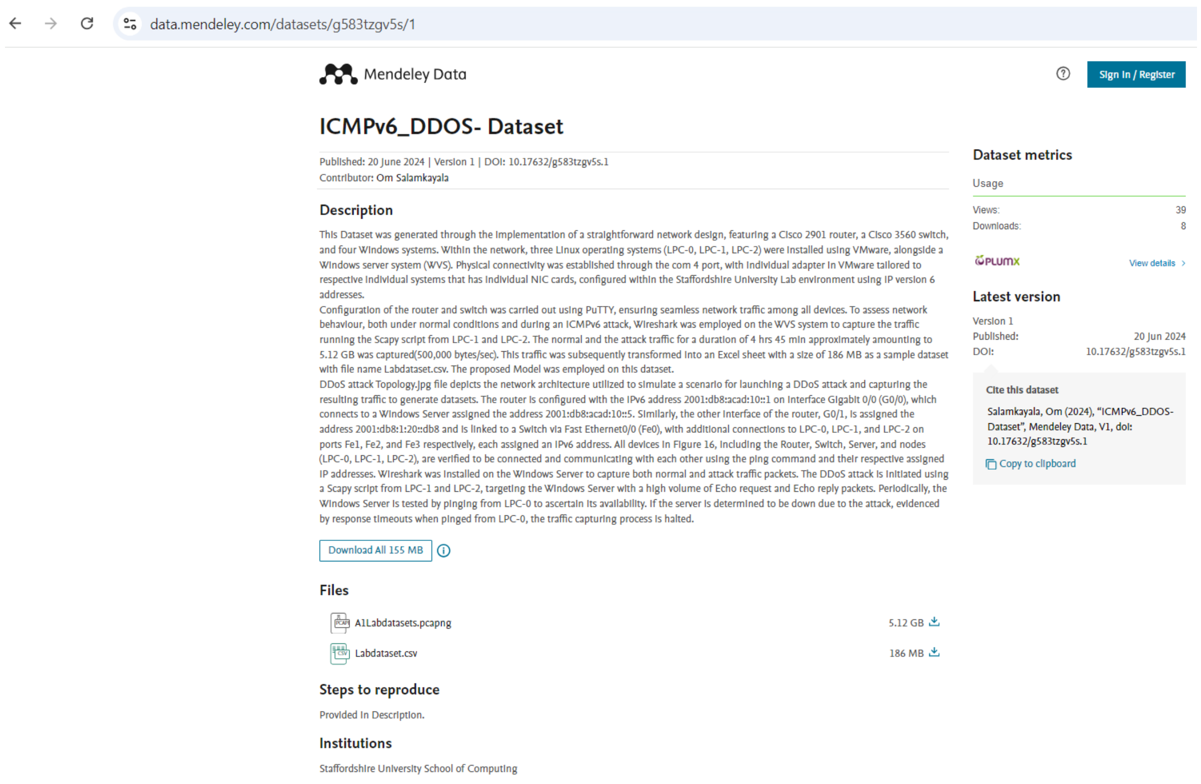
Packages: No packages published

README

DDoS-Datasets

DDoS generated Datasets This Dataset was produced on a single machine equipped with an Intel i7 11th generation 2.30 GHz processor, 64 GB RAM, and a 2 TB hard disk. Within a VMware environment, four virtual machines were set up, comprising one Linux machine (LVPC), two Windows machines (WVPC-1 and WVPC-2), and one Windows server (WVS). These virtual machines were interconnected through VMware network adapters on single NIC card of the host machine with IP version 6 addresses. To generate a traffic in the instance of an attack, an ICMPv6 DDoS attack was launched using a Scapy script. The attack is launched from two virtual machines (LVPC and WVPC-1), targeting the Windows server (WVS). The ensuing network traffic, encompassing both normal activity and the ICMPv6 attack, was captured on the Windows server (WVS) using Wireshark, resulting in a dataset size of 18.3 MB (60,000 bytes/sec). Subsequently, the captured traffic underwent transformation into an Excel sheet as scdtsets.csv of size 58.2 MB. The proposed Model was employed on these data sets.

Mendeley:



data.mendeley.com/datasets/g583tzgv5s/1

Mendeley Data

Sign In / Register

ICMPv6_DDOS- Dataset

Published: 20 June 2024 | Version 1 | DOI: 10.17632/g583tzgv5s.1
Contributor: Om Salamkayala

Description

This Dataset was generated through the implementation of a straightforward network design, featuring a Cisco 2901 router, a Cisco 3560 switch, and four Windows systems. Within the network, three Linux operating systems (LPC-0, LPC-1, LPC-2) were installed using VMware, alongside a Windows server system (WVS). Physical connectivity was established through the com 4 port, with individual adapter in VMware tailored to respective individual systems that has individual NIC cards, configured within the Staffordshire University Lab environment using IP version 6 addresses.
Configuration of the router and switch was carried out using PuTTY, ensuring seamless network traffic among all devices. To assess network behaviour, both under normal conditions and during an ICMPv6 attack, Wireshark was employed on the WVS system to capture the traffic running the Scapy script from LPC-1 and LPC-2. The normal and the attack traffic for a duration of 4 hrs 45 min approximately amounting to 5.12 GB was captured(500,000 bytes/sec). This traffic was subsequently transformed into an Excel sheet with a size of 186 MB as a sample dataset with file name Labdataset.csv. The proposed Model was employed on this dataset.
DDoS attack Topology.jpg file depicts the network architecture utilized to simulate a scenario for launching a DDoS attack and capturing the resulting traffic to generate datasets. The router is configured with the IPv6 address 2001:db8:acad:10::1 on interface Gigabit 0/0 (G0/0), which connects to a Windows Server assigned the address 2001:db8:acad:10::5. Similarly, the other interface of the router, G0/1, is assigned the address 2001:db8:1:20::db8 and is linked to a Switch via Fast Ethernet0/0 (Fe0), with additional connections to LPC-0, LPC-1, and LPC-2 on ports Fe1, Fe2, and Fe3 respectively, each assigned an IPv6 address. All devices in Figure 16, including the Router, Switch, Server, and nodes (LPC-0, LPC-1, LPC-2), are verified to be connected and communicating with each other using the ping command and their respective assigned IP addresses. Wireshark was installed on the Windows Server to capture both normal and attack traffic packets. The DDoS attack is initiated using a Scapy script from LPC-1 and LPC-2, targeting the Windows Server with a high volume of Echo request and Echo reply packets. Periodically, the Windows Server is tested by pinging from LPC-0 to ascertain its availability. If the server is determined to be down due to the attack, evidenced by response timeouts when pinged from LPC-0, the traffic capturing process is halted.

Download All 155 MB

Files

A1Labdatasets.pcapng	5.12 GB
Labdataset.csv	186 MB

Steps to reproduce

Provided in Description.

Institutions

Staffordshire University School of Computing

Dataset metrics

Usage

Views:	39
Downloads:	8

PLUMX View details

Latest version

Version 1

Published:	20 Jun 2024
DOI:	10.17632/g583tzgv5s.1

Cite this dataset

Salamkayala, Om (2024), "ICMPv6_DDOS-Dataset", Mendeley Data, V1, doi: 10.17632/g583tzgv5s.1

Copy to clipboard