

An Efficient Self Attention-Based 1D-CNN-LSTM Network for IoT Attack Detection and Identification Using Network Traffic

by

Tinshu Sasi

Bachelor of Technology (Computer Science and Engineering), Manav Rachna
International University, 2014

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

Master of Computer Science

In the Graduate Academic Unit of Computer Science

Supervisor(s): Rongxing Lu, PhD, Faculty of Computer Science
Arash Habibi Lashkari, PhD, Faculty of Computer Science
Examining Board: Sajjad Dadkhah , PhD, Faculty of Computer Science
Saqib Hakak, PhD, Faculty of Computer Science
Hamed Asgari Moslehabadi, PhD, Department of
Mechanical Engineering

This thesis is accepted by the
Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

June, 2024

© Tinshu Sasi, 2024

Abstract

In the last 10 years, the Internet of Things (IoT) has played a crucial role in the digital transformation of society. However, it is also facing increased security vulnerabilities because of the wide range of devices it encompasses. This research presents a novel mechanism called the Self Attention-Based 1D-CNN-LSTM Network for detecting IoT attacks. The proposed mechanism achieves an impressive accuracy of 99.96% and efficiently differentiates between malicious and benign samples. By employing Shapley Additive Explanations (SHAP), we were able to identify important predictive features from the preprocessed data, which were retrieved using CICFlowmeter. This has strengthened the dependability of the model. In addition, we enhanced the model by training it on a smaller collection of features, resulting in shorter training time while preserving accuracy. We have also generated novel IoT tabular datasets consisting of nine widely accessible IoT datasets, as specified in Table 5.1, to evaluate the model's robustness and showcase its efficacy in IoT security.

Dedications

I dedicate my thesis to my spouse, Sweety Sinha, for her unwavering support and encouragement.

Acknowledgments

I express my heartfelt gratitude to my professors, Dr. Rongxing Lu and Dr. Arash Habibi Lashkari, for their unwavering assistance and direction. They have provided me with motivation and support throughout the whole process.

I express my gratitude to the University of New Brunswick, specifically the Faculty of Computer Science, for providing me with the chance and assistance during my program. I am also thankful to all the professors who imparted their knowledge to me throughout the course of the program.

I express my gratitude to the members of my examining committee for their valuable input and suggestions.

Table of Contents

Abstract	ii
Dedication	iii
Acknowledgments	iv
Table of Contents	v
List of Tables	viii
List of Figures	x
Abbreviations	xii
1 Introduction	1
1.1 Introduction	1
1.2 What is an IoT architecture?	2
1.3 What are IoT attacks?	5
1.4 What is the difference between IoT and IT attacks?	6
1.5 Problem Statement	7
1.6 Summary of Contributions	8
1.7 Thesis Organization	8
2 Background of IoT Attacks	10
2.1 Technical Terms	10

2.2	Common IoT Vulnerabilities	11
2.3	Summary of IoT Attacks in Training dataset	15
2.4	Concluding Remarks	23
3	Literature Review	24
3.1	Overview	24
3.2	Literature Review	24
3.3	Different techniques used to perform IoT attack detections	30
3.4	Concluding Remarks	34
4	Proposed Method	36
4.1	Motivation	36
4.2	CICFlowmeter	37
4.3	Data Pre-Processing	37
4.4	What are Convolutional Neural Networks?	39
4.4.1	1D-CNN	40
4.4.2	2D-CNN	40
4.5	CNN Operations	40
4.5.1	Convolution	41
4.5.2	Activation Function	42
4.5.3	Pooling	43
4.5.4	Fully-Connected Layers	44
4.5.5	Dropout Layer	45
4.5.6	Loss Function	46
4.5.7	Optimizer	47
4.6	What are Residual blocks?	48
4.7	What are Longest Short Term Memory Network?	49
4.8	What are Attention Layers?	51

4.8.1	Self-Attention	52
4.9	What are SHAP values?	55
4.9.1	SHAP Feature Importance Scores	57
4.10	Proposed Mechanism	58
4.10.1	Pre-processing - Stage I	58
4.10.2	Proposed Architecture - Stage II	61
4.11	Concluding Remarks	63
5	Experiments & Results	65
5.1	Overview	65
5.2	Experimental Setup	65
5.3	Datasets	66
5.4	Features	78
5.5	Finalizing the Proposed Model	84
5.5.1	Performance Metrics	85
5.6	Experimental Results	87
5.7	Concluding Remarks	100
6	Conclusion & Future Works	102
6.1	Conclusions	102
6.2	Future Works	103
	Bibliography	105
	Vita	

List of Tables

1	List of Acronyms and Abbreviations	xii
1.1	IoT Attacks Vs IT Attacks	6
2.1	Top 10 OWASP IoT Vulnerabilities	11
3.1	Comparison of IoT Attack Surveys	25
4.1	Activation Functions [10]	42
4.2	Activation Functions Part 2 [10]	43
4.3	Pooling Functions [36]	44
4.4	Loss Functions [6]	46
4.5	Optimizers [28] [20]	47
4.6	Hyperparameters List	62
5.1	Augmented Datasets	66
5.2	List Of All Malicious Activities Present In All Datasets in Table 5.1 [60] [77]	73
5.3	CICFlowmeter Extracted Features List	78
5.4	Baseline Models Evaluation Results	88
5.5	Our Model Accuracy List	92
5.6	Confusion Metrics For Initial Model Across All datasets	92
5.7	Top 7 Best Features For Retraining	95
5.8	Reduced Set's Features Accuracy List	97

5.9	Confusion Metrics For 7 Feature Reduced Model Across All datasets .	100
-----	---	-----

List of Figures

1.1	IoT Architecture Levels	3
3.1	Overview of Available IoT Attack Detection and Identification Methods	31
4.1	Data Pre-processing	37
4.2	CNN Architecture	39
4.3	Dropout Layer	45
4.4	Residual Block Cell	48
4.5	LSTM Cell	50
4.6	LSTM Network	50
4.7	Self Attention Mechanism	52
4.8	Model Architecture	59
4.9	Residual Block	60
5.1	Comparison between Baseline Models and Our Model	88
5.2	Our Model's Training History Results	90
5.3	Our Model's Confusion Matrix Results	91
5.4	SHAP Summary Plot for Our Model (Trained on CIC-BCCC-NRC-IoT-2023 dataset)	93
5.5	Feature Importance Graph for CIC-BCCC-NRC-IoT-2023 dataset . .	94
5.6	Reduced Set Feature's Training History Results	99
5.7	Reduced Set Feature's Confusion Matrix Results	99

5.8	Performance Comparison between Initial Model vs Reduced Feature	
	Set Model	101

Abbreviations

Table 1: List of Acronyms and Abbreviations

Acronym/Abbreviation	Description
<i>IoT</i>	Internet of Things
<i>CAN</i>	Controller Area Network
<i>IIoT</i>	Industrial Internet of Things
<i>AGVs</i>	Automated Guided Vehicles
<i>DTPs</i>	Data Transfer Protocols
<i>MQTT</i>	Message Queue Telemetry Transport
<i>CoAP</i>	Constrained Application Protocol
<i>DDS</i>	Data Distribution Service
<i>AMQP</i>	Advanced Message Queuing Protocol
<i>IT</i>	Information Technology
<i>TCP</i>	Transport Control Protocol
<i>CNN</i>	Convolutional Neural Networks
<i>LSTM</i>	Long Short Term Memory
<i>SHAP</i>	Shapley Additive Explanations
<i>OWASP</i>	Open Worldwide Application Security Project
<i>ACL</i>	Access Control List
<i>DDoS</i>	Distributed Denial-of-Service

Continued on next page

Table 1 – Continued from previous page

Acronym/Abbreviation	Description
<i>ICMP</i>	Internet Control Message Protocol
<i>MTU</i>	Maximum Transmission Unit
<i>UDP</i>	User Datagram Protocol
<i>DNS</i>	Domain Name System
<i>C&C)</i>	Command & Control
<i>GRE</i>	Generic Routing Encapsulation
<i>ETH</i>	Ethernet
<i>MitM</i>	Man-in-the-Middle
<i>ARP</i>	Address Resolution Protocol
<i>MAC</i>	Media Access Control Address
<i>HTTPS</i>	Hypertext Transfer Protocol Secure
<i>BC</i>	Blockchain
<i>FC</i>	Fog Computing
<i>EC</i>	Edge Computing
<i>ML</i>	Machine Learning
<i>SIEM</i>	Security Information and Event Management
<i>DL</i>	Deep learning
<i>RNN</i>	Recurrent Neural Network
<i>KNN</i>	K-Nearest Neighbors
<i>ReLU</i>	Rectified Linear Unit
<i>ELU</i>	Exponential Linear Unit
<i>BN</i>	Batch Normalization
<i>MSE</i>	Mean Squared Error
<i>MAE</i>	Mean Absolute Error

Continued on next page

Table 1 – Continued from previous page

Acronym/Abbreviation	Description
<i>SGD</i>	Stochastic Gradient Descent
<i>Adam</i>	Adaptive Moment Estimation
<i>RMSProp</i>	Root Mean Squared Propagation
<i>AdaGrad</i>	Adaptive Gradient Algorithm
<i>CIC</i>	Canadian Institute for Cybersecurity

Chapter 1

Introduction

1.1 Introduction

The digital revolution has significantly transformed our lives, with Internet of Things (IoT) playing a pivotal role. However, the rapid development of IoT in most corners of life leads to various emerging cybersecurity threats. Therefore, detecting and preventing potential attacks in IoT networks have recently attracted paramount interest from academia and industry. Among various attack detection approaches, machine learning-based methods, especially deep learning, have demonstrated outstanding potential thanks to their early detection capability [78].

Over the last several years, there has been a substantial increase in the number of attacks specifically aimed against Internet of Things (IoT) devices. These encompass various types of cyber attacks, such as infiltrating wireless webcams to gain unauthorized access to surveillance cameras and violate user privacy; targeting implantable cardiac devices to potentially deplete battery life, disrupt pacing, and cause electric shocks, thereby endangering patients' lives; compromising children's smartwatches to expose their location data and personal information, pose multiple safety risks; and manipulating the Controller Area Network (CAN) bus of vehicles to potentially

alter speed and direction, thereby posing threats to public safety. An examination into smart house hacking found that for one week, fraudsters and unidentified groups launched over 12,000 attacks against various smart home devices, including TVs, thermostats, smart kettles, and security systems. [79] [80].

1.2 What is an IoT architecture?

The IoT architecture refers to the organization and setup of IoT devices to fulfill users' particular requirements and requests. An IoT system is divided into three to seven layers, depending on their complexity, and each layer has a specific function. The lack of established protocols in the architecture of the Internet of Things (IoT) presents further difficulties regarding interoperability, security, and several other issues. The Internet of Things (IoT) architecture can include up to seven levels. [70]. Figure 1.1 provides a detailed illustration of the Internet of Things (IoT) and Industrial Internet of Things (IIoT).

- *Perception Layer*: The perception layer, also known as the device layer, includes a variety of sensors such as RFID scanners, security cameras, GPS modules, and so on. These devices, such as conveyor systems, industrial robots, and automated guided vehicles (AGVs), may be used with industrial gear. These gadgets collect sensory data, monitor the production floor and surroundings, transport raw materials, and so forth [64].
- *Transport Layer/ Network Layer*: The Transport/Network layer is responsible for transferring data to the processing systems of the subsequent layer [64]. IoT gateways must first transform the incoming input from analogue to digital format. Subsequently, the gateway can transmit the data to a local or cloud data center via several data transfer protocols (DTPs). Some of the leading IoT protocols are Bluetooth, Wi-Fi, Zigbee, Z-Wave, 6LoWPAN (IPv6 over Low-

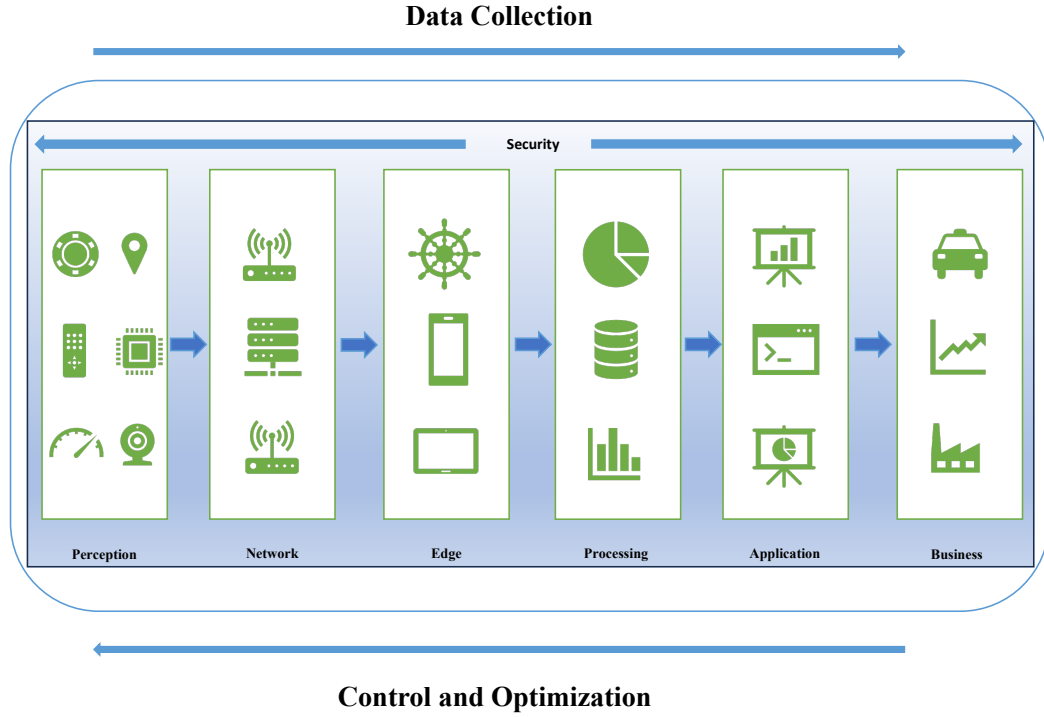


Figure 1.1: IoT Architecture Levels

Power Wireless Personal Area Networks), MQTT (Message Queue Telemetry Transport), CoAP (Constrained Application Protocol), DDS (Data Distribution Service), AMQP (Advanced Message Queuing Protocol) [70].

- *Edge Layer:* The edge layer in an IoT configuration consists of the physical hardware, embedded operating system, and device firmware. With the growing number of interconnected devices, latency becomes a prominent issue in more extensive IoT networks. Edge computing, aided by the edge layer, resolves this problem by allowing data processing and analysis close to the data source. Latency, the period of time between the detection of an event and the execution of an action, is a critical concern for devices that are linked to a network. Reducing latency may be accomplished by placing processing resources close to the sensors or at the network edge, enabling prompt connection and data

exchange between devices [56].

- *Processing Layer*: The Processing layer, sometimes called the Middleware layer, comprises servers and databases. Its primary functions include decision-making, executing optimization algorithms, and storing large amounts of data [64]. This layer consists of cloud computing platforms that can analyze and interpret data from the physical environment. The system processes unprocessed sensor data and transforms it into relevant insights using cloud services and comprehensive data modules. Furthermore, the processing layer allows the system to promptly respond to inputs and outputs. It can make assessments and carry out tasks based on the information it receives. The data received in the perception step is used in this layer to generate predictions and provide insights [69].
- *Application Layer*: The application layer of IoT infrastructure is responsible for data analysis to solve business issues or achieve specific goals. The application layer provides customized functionality to meet the unique requirements of end users. The applications and services in this layer are constructed on top of the processing layer. Software tools facilitate converting data from the processing layer into meaningful information that humans can understand or use by automated processes [4].
- *Business Layer*: The business layer acts as the central point where choices and solutions are developed from the data analysis conducted in the application layer. The application layer may consist of several instances inside this layer. At the business layer, identifiable patterns from the application layer are used to gain a deeper understanding of business insights, predict future trends, and make operational decisions that improve productivity, security, cost-effectiveness, customer satisfaction, and other important business factors. In addition, the business layer is accountable for overseeing commercial transactions

and models related to interconnected devices. It includes the administration of business processes, data analysis, and rules implementation. It serves as the basis for controlling business logic and setting up procedures to achieve all the business goals of an IoT system [70] [69] [4].

- *Security Layer*: The security layer is present across all levels of the IoT architecture and is essential for the efficiency of an IoT solution. Considering that Internet of Things (IoT) devices often deal with confidential information, it is crucial to implement strong security measures [70] [4].

1.3 What are IoT attacks?

IoT attacks constitute cyberattacks leveraging IoT devices to access consumers' sensitive data. Typically, attackers deploy malware on these devices, causing damage or infiltrating additional organizations' data. Due to insufficiently designed security mechanisms, IoT devices emerge as prominent vulnerabilities within organizational infrastructures, posing substantial security risks. Basic IoT devices often lack robust built-in security measures to counter cyber threats. Given their limited functionalities and purposes, security considerations for such devices are frequently overlooked, rendering them susceptible to cyberattacks. Hackers and organizations can use common flaws and "zero-day exploits" to attack IoT devices in various ways [60].

1.4 What is the difference between IoT and IT attacks?

Table 1.1: IoT Attacks Vs IT Attacks

Category	IoT Attacks	IT Attacks
Attack Surface	Limited resources, higher vulnerability.	Robust security, lower vulnerability.
Diversity of Devices	Varied types, complex security.	Standardized, simplified security.
Impact	Harmful physical consequences.	Data theft, service disruption.
Legacy Devices	Older devices, no updates, higher risk.	Regular updates, lower risk.

Unlike conventional Information Technology (IT) attacks, Internet of Things (IoT) attacks provide distinct issues that necessitate specific security solutions to minimize the associated dangers fully. The distinction between IT and IoT attacks is outlined in Table 1.1. Some of the different ways are:

- *Attack surface*: IoT devices often possess limited processing capabilities and resources, resulting in potential deficiencies in security features compared to traditional IT systems. Consequently, IoT devices are more susceptible to attacks due to reduced defences. [60].
- *Diversity of devices*: The wide array of IoT device types, varying in form factor, operating systems, and network connectivity, complicates establishing standardized security measures. This diversity renders certain devices more prone to vulnerabilities and targeted attacks. [60].

- *Physical impact:* Many IoT devices are integral to critical infrastructure or life-sustaining systems, such as medical equipment, thus exposing them to cyberattacks with severe physical ramifications. In contrast, typical IT attacks aim to compromise data integrity or disrupt services. [60].
- *Legacy devices:* IoT devices often have extended lifespans, resulting in a proliferation of older, unsupported devices. The inability of legacy devices to receive software updates or security patches renders them particularly vulnerable to exploitation or compromise. [60].

1.5 Problem Statement

As discussed in the previous sections, the proliferation of IoT devices across various industries underscores the importance of safeguarding these devices against potential cyberattacks. Identifying and discerning malicious and benign instances by analyzing time-related features extracted from IoT TCP flow network traffic data presents a formidable challenge within the network security domain. Determining the optimal features for detecting malicious and benign instances among the extracted features and establishing their efficacy and reliability constitutes a critical inquiry. Given the escalating complexity and volume of IoT network data, it is imperative to devise robust detection methodologies to distinguish between malicious and benign activities accurately. Addressing this imperative is crucial for fortifying IoT systems and networks against potential security vulnerabilities, thereby upholding the integrity and reliability of interconnected devices and services.

1.6 Summary of Contributions

This research endeavour mainly focuses on safeguarding IoT devices by identifying IoT attacks and analyzing time-related features derived from IoT network traffic TCP flow data. This will be achieved using deep learning algorithms to distinguish between malicious and benign samples. To summarize, this thesis has made the following contributions:

- Our proposal uses an Self Attention-based 1D-CNN and LSTM network to identify IoT attacks by analyzing time-related features collected from TCP flow data.
- To enhance the model’s credibility, the model will be evaluated on eight publicly accessible external datasets that have been enhanced and pre-processed using CICFlowmeter.
- Determining the optimal hyperparameter combination for the model with the highest performance metrics.
- Employing Shapley Additive Explanations (SHAP) to compute feature importance scores and utilize them to identify the most optimal features from the extracted feature list.
- Reducing the model size by retraining it on the most optimal features while maintaining comparable performance to the original model.
- Releasing CIC-BCCC-NRC_TabularIoTAttack-2024 dataset which includes over 80 extracted features from eight IoT datasets.

1.7 Thesis Organization

The structure of the remainder of this thesis is outlined as follows:

1. Chapter 2: *Background of IoT Attacks* provides a fundamental understanding of the topic, including a discussion of key technical terms associated with the subject matter, mainly focusing on IoT attacks. It also summarizes the IoT attacks mentioned in the training datasets.
2. Chapter 3: *Literature Review* reviews prior research on IoT Attacks, examining the detection methodologies using machine learning and deep learning techniques. This chapter also offers a comprehensive classification of IoT attack types.
3. Chapter 4: *Proposed Method* outlines the motivation behind the proposed approach, detailing the preprocessing steps and describing the model architecture, including a thorough explanation of each principal component within the architecture.
4. Chapter 5: *Experiments & Results* presents the outcomes of the implemented research. This chapter covers the dataset, experimental setup, features, metrics, and the results obtained from the proposed method.
5. Chapter 6: *Conclusion & Future Works* concludes the thesis by summarizing the contributions, discussing encountered challenges, and providing insights into potential future research directions.

Chapter 2

Background of IoT Attacks

2.1 Technical Terms

Before delving deeper into the intricacies of IoT attacks, acquiring a fundamental understanding of the subject is imperative. It is essential to thoroughly comprehend the various technical terminologies associated with this study area [60].

- *Vulnerability*: It denotes the intrinsic weaknesses of a system or its design, which permit unauthorized entities to execute commands, access data without appropriate authorization, and possibly initiate denial-of-service attacks. Such vulnerabilities can be detected across various domains of IoT systems. They may appear in the system's hardware or software, the protocols and procedures implemented within these systems, and even in the behaviours and actions of the users interacting with the system [60].
- *Exposure*: It refers to a problem or mistake within the system configuration that allows an unauthorized person to undertake actions to obtain information [60].
- *Threats*: It refers to an intentional or unintentional action that exploits vulnerabilities within a system [60].

- *Attacks*: It is defined as deliberate actions to damage a system or disrupt its normal operations by exploiting vulnerabilities using various strategies and tools. Attackers engage in these hostile acts to achieve specific goals, which may be motivated by personal gratification or financial gain [60].

2.2 Common IoT Vulnerabilities

Table 2.1: Top 10 OWASP IoT Vulnerabilities

Rank	Vulnerability	Description
1	Weak, Guessable, or Hard-coded Passwords	Inadequate credentials susceptible to brute force attacks or publicly accessible, including backdoors in firmware or client software.
2	Insecure Network Services	Presence of unnecessary or vulnerable network services, particularly online access, threaten information confidentiality, integrity, and availability.
3	Insecure Ecosystem Interfaces	Presence of insecure online, back-end API, cloud, or mobile interfaces outside the device ecosystem, potentially compromising device security.

Rank	Vulnerability	Description
4	Lack of Secure Update Mechanism	Inability of devices to undergo secure updates, lacking firmware validation, secure delivery, anti-rollback procedures, or alerts on security changes.
5	Use of Insecure or Outdated Components	Utilization of outdated or vulnerable software or hardware components, potentially exposing devices to unauthorized access.
6	Insufficient Privacy Protection	Insecure storage or access of user's personal information within the ecosystem.
7	Insecure Data Transfer and Storage	Absence of encryption or access control measures for sensitive data throughout the ecosystem.
8	Lack of Device Management	Absence of adequate security support for production devices, leading to deficiencies in asset and update management.
9	Insecure Default Settings	Distribution of devices with unsecured default configurations or limited user control over settings.

Rank	Vulnerability	Description
10	Lack of Physical Hardening	Absence of Physical Security Measures, enabling attackers to access critical information or assume local control.

Table 2.1 showcases the ten most significant vulnerabilities in IoT according to OWASP that make IoT devices susceptible to IoT attacks [7].

- *Weak/Default Passwords*: Absence of a strong password recovery system; Weak or default password; Non-implementation of stricter password rules; Inability to change the username and password associated with the account.
- *Insecure Network Services*: Adversaries exploit vulnerabilities in IoT devices' communication protocols and services to gain unauthorized access and undermine the confidentiality of sensitive information transmitted between the device and a server.
- *Insecure Ecosystem Interfaces*: The vulnerability of the device or its connected components arising from the insecure web, backend API, cloud, or mobile interfaces in the external ecosystem.
- *Lack of Secure Update Mechanism*: Insufficient ability to update devices securely. These deficiencies encompass the absence of device-based firmware validation, the unsecured transmission of data without encryption, the lack of anti-rollback methods, and the failure to provide notifications regarding security changes resulting from updates.
- *Use of Insecure or Outdated Components*: The utilization of obsolete or insecure software components or libraries that could expose the device to potential

attacks. This entails the utilization of external software or hardware obtained from a compromised supply chain, as well as the unsecured modification of system platforms. The security of the IoT ecosystem may be compromised by vulnerabilities in software dependencies or outdated systems.

- *Insufficient Privacy Protection:* Users' personal information is saved on the device or in the ecosystem and used accidentally, improperly, or illegally. Information about one's health, energy use, and driving habits can fall into this category of privacy concerns. Privacy is at risk without adequate safeguards, and there can be legal consequences for failing to take the necessary precautions.
- *Insecure Data Transfer and Storage:* The absence of encryption or access control for sensitive data at any point in the ecosystem, including whether it is stored, transferred, or processed. Data is critical in ensuring the reliability and integrity of IoT applications since it is used in automated controls and decision-making processes. Unauthorized access or usage will result in adverse consequences.
- *Lack of Device Management:* Lack of security support for production-ready devices, including asset management, update management, secure decommissioning, systems monitoring, and response capabilities. Unauthorized devices can access business networks, monitor activity, and intercept data if exposed to the IoT ecosystem.
- *Insecure Default Settings:* Systems or devices that lack the ability to enhance system security by restricting users from modifying configurations or that come with insecure default settings. Once the settings have been acquired, attackers can exploit hardcoded default passwords, concealed backdoors, and weaknesses in the device firmware. The user encounters difficulty in simultaneously modifying various parameters.

- *Lack of Physical Hardening*: The absence of physical safeguards allows potential attackers to get sensitive information that could be utilized in subsequent large-scale attacks or local device takeover. Internet of Things (IoT) devices are deployed in distant and dispersed environments. An attacker can disrupt the services offered by IoT devices by gaining access to the physical layer and making alterations.

2.3 Summary of IoT Attacks in Training dataset

1. *ACK Fragmentation*: A Fragmented ACK attack is a modified version of the ACK & PSH-ACK Flood, where 1500-byte packets are utilized to monopolize the target network's bandwidth while maintaining a modest packet rate. Applying application-level filters on network equipment, such as routers, would require the equipment to reassemble the packets, consuming a significant amount of its resources. Without any filters, these attack packets can traverse various network security devices, such as routers, ACLs, and firewalls, without being noticed. The fragmented packets typically consist of irrelevant or useless data, as the attacker's objective is to fully utilize the target network's available capacity. Similar to other DDoS attacks, the objective of a DDoS Fragmented ACK attack is to obstruct the service for other users by impeding or causing the target to crash by using irrelevant data [57].
2. *ICMP Flood*: An ICMP flood is a form of denial-of-service attack, or DoS attack, which exploits the Internet Control Message Protocol (ICMP) by utilizing echo-requests and echo-replies, often known as pings, to assess the operational status and connectivity of a device. An ICMP flood attack, also known as a "ping flood attack," occurs when attackers overrun the bandwidth of a specific network router or IP address. They achieve this by inundating the router or IP

address with carefully prepared ICMP packets, causing it to become overloaded and unable to transfer traffic to the next downstream hop. When the device attempts to react, it uses up all of its available resources (such as memory, processing power, and interface rate), which prevents it from fulfilling genuine requests or serving consumers [5].

3. *ICMP Fragmentation*: An IP/ICMP fragmentation DDoS attack is a prevalent type of volumetric denial of service (DoS) attack. Datagram fragmentation algorithms are employed to inundate the network during such an attack.

IP fragmentation involves dividing IP datagrams into smaller packets, transmitting them over a network, and then reassembling them back into the original datagram during regular communication. This technique is essential to adhering to the size constraints that each network can handle. The maximum transmission unit (MTU) is an upper bound on the data size that can be transmitted.

A packet that exceeds the maximum size must be divided into smaller fragments to ensure successful transmission. As a result, multiple packets are transmitted, one of which includes comprehensive information about the packet, such as the source/destination ports, length, and other relevant details.

The remaining pieces lack further components and only contain an IP header and a data payload. These fragments lack information regarding protocol, size, or ports.

The attacker might specifically utilize IP fragmentation to target communication systems and security components. ICMP-based fragmentation attacks commonly include submitting forged fragments that cannot be reassembled. Consequently, the fragments are stored temporarily, occupying memory and potentially depleting all available memory resources. This DDoS attack involves the transmission of counterfeit UDP or ICMP packets. These packets

are intentionally crafted to appear more significant than the network's maximum transmission unit (MTU). However, only certain portions of the packets are transmitted. Since the packets are counterfeit and cannot be reconstructed, the server's resources are rapidly depleted, resulting in its unavailability to genuine traffic [54].

4. *PSH ACK Flood*: ACK or PUSH ACK packets are utilized bidirectionally to transmit data until the session is terminated after establishing a connection between the host and the client. A victim server, vulnerable to an ACK flood attack, receives faked ACK packets that do not correspond to any sessions in the server's connection list.

The targeted server exhausts its system resources to ascertain the legitimacy of the falsified packets within a session, leading to a decline in performance and limited service availability [24].

5. *DDoS RST FIN Flood*: To terminate a TCP SYN session, the client and the host exchange RST or FIN packets. During an RST or FIN flood, the targeted server is bombarded with a high volume of faked RST or FIN packets not associated with any of the sessions stored in the server's database. The affected server must commit substantial system resources to correlate incoming packets with existing connections, leading to diminished server performance and partial unavailability [26].

6. *SYN Flood*: A SYN flood, also known as a half-open attack, is a denial-of-service (DDoS) attack that seeks to render a server inaccessible to genuine traffic by depleting its available resources. Through the repetitive transmission of initial connection request (SYN) packets, the attacker can inundate all accessible ports on the server machine being targeted, resulting in the targeted device responding to genuine traffic with sluggishness or not responding at all [14].

7. *Synonymous IP Flood*: In this attack, the victim server is inundated with a substantial influx of falsified TCP SYN packets with identical source and destination addresses in the header, which corresponds to the victim's address. The designated server initiates the utilization of system resources to process every packet [27].
8. *DDoS TCP Flood*: A TCP SYN Flood attack aims to exploit the TCP three-way handshake procedure, which is fundamental for establishing connections in TCP/IP networks. A TCP SYN Flood attack involves the deliberate sending of several SYN requests to a target server, with the purpose of not sending the final ACK. As a result, the server remains idle, awaiting a response that it never receives, which leads to the utilization of resources for each of these partially established connections [34].
9. *UDP Flood*: A UDP Flood attack is a form of volumetric Denial of Service (DoS) attack that takes advantage of the User Datagram Protocol (UDP). UDP, unlike TCP, lacks session and connection features, making it an exceptional target for attackers. A UDP Flood attack involves an attacker inundating the victim system with an enormous volume of UDP packets directed at random ports. This influx of packets compels the host to:
 - Scan for active programs on each port.
 - Recognize that no active applications are listening on several ports.
 - Reply with an ICMP Destination Unreachable packet using the Internet Control Message Protocol (ICMP).

The large quantity of UDP packets forces the targeted system to emit a plethora of ICMP packets. This can cause the system to become inaccessible to authorized customers. To further conceal their harmful actions, attackers may falsify

the IP address of the UDP packets. This guarantees that the influx of return ICMP packets is prevented from reaching them, essentially concealing their whereabouts [35].

10. *UDP Fragmentation*: It is one of the variations of UDP flood. The distinguishing factor lies in utilizing packets of the greatest permissible size to saturate the channel with the least possible amount of packets. Given that these packet pieces are counterfeit and unrelated to genuine data, the targeted server that receives them will allocate resources to reconstruct non-existent packets from the counterfeit fragments. Eventually, this will lead to the depletion of system resources and the subsequent server crash or result in the overflow of channels. Like a UDP flood, this attack is challenging to screen and has a greater risk of channel overflow [25].
11. *DNS Spoofing*: DNS spoofing is a cyberattack in which a hacker deceives a computer or network into thinking it is interacting with a genuine website or server. Essentially, the computer engages with a counterfeit website or server established by the attacker. This deceitful practice directs people to fraudulent websites, exposing them to the risks of identity theft, financial fraud, malware, and other online concerns [63].
12. *HTTP Flood*: An HTTP flood is a form of Distributed Denial of Service (DDoS) attack where the attacker takes advantage of apparently valid HTTP GET or POST requests to attack a web server or application. HTTP flood attacks are volumetric attacks that typically involve a botnet, a collection of compromised machines that have been fraudulently taken over, frequently with the help of software such as Trojan Horses. HTTP floods, a type of Layer 7 attack, do not rely on faulty packets, spoofing, or reflection techniques. They can bring down a targeted site or server with less bandwidth than other attacks. Consequently,

they require a thorough comprehension of the specific site or application being attacked, and each attack must be meticulously tailored to ensure its effectiveness. This dramatically enhances the difficulty of detecting and obstructing HTTP flood attacks [33].

13. *Mirai*: The Mirai botnet is malware that was created to take control of Internet of Things (IoT) devices and transform them into remotely operated "bots" that can carry out highly impactful volumetric distributed denial of service (DDoS) attacks. The Mirai botnet conducts scans to identify susceptible IoT devices that possess open ports or utilize default usernames and passwords. Upon identifying these susceptible devices, it employs vulnerabilities to acquire entry and contaminates them with its malicious code. Once infected, the device becomes part of the Mirai botnet, enabling the attacker to issue commands from a central server called a "command & control" server (C&C). Once established, this command and control (C&C) server can be utilized to initiate extensive distributed denial-of-service (DDoS) attacks on websites, networks, and other digital infrastructure by harnessing the collective power of all the bots within the Mirai Botnet simultaneously [58].

- *Mirai GRE-ETH Flood*: GRE, short for Generic Routing Encapsulation, is a protocol for creating virtual point-to-point connections over an IP network. It allows for the encapsulation of many network layer protocols. DDoS scrubbing providers utilize GRE as part of their mitigation architecture.

GRE-ETH Flood is a network attack that targets explicitly network devices, such as routers and switches, by overwhelming them with excessive GRE (Generic Routing Encapsulation) and Ethernet (ETH) frames.

This attack involves the attacker producing many GRE and Ethernet

frames and directing them towards the target device to overpower its processing capabilities. The inundation of packets can deplete system resources such as CPU, memory, and bandwidth, making the device unusable or incapable of processing valid data. The primary aim of a GRE-ETH flood attack can differ. Still, typical objectives involve disrupting network services, inducing denial of service (DoS) or distributed denial of service (DDoS) scenarios, and perhaps exploiting vulnerabilities in the target device’s management of GRE and Ethernet traffic [81].

- *Mirai GRE-IP Flood:* A GRE-IP Flood is a network attack characterized by the deliberate inundation of a network with a substantial number of GRE (Generic Routing Encapsulation) and IP (Internet Protocol) packets. The attack involves the attacker creating a substantial quantity of GRE-encapsulated IP packets and directing them towards the target network to exhaust its resources. The inundation of packets depletes network bandwidth, router computational capacity, and other resources, resulting in network congestion, deceleration, or even total unavailability of services. The purpose of a GRE-IP Flood attack might vary. Still, typical objectives include interrupting network operations, inducing denial of service (DoS) circumstances, or exploiting weaknesses in network infrastructure [81].
- *Mirai UDP Plain:* A UDP Plain attack, often referred to as a UDP flood attack, is a network-based denial-of-service (DoS) attack that explicitly targets a server or network infrastructure by overwhelming it with a large number of User Datagram Protocol (UDP) packets. In a UDP Plain attack, the attacker inundates the target server or network with a substantial volume of UDP packets, causing it to become overwhelmed and unable to handle the incoming packets effectively. In contrast to TCP, UDP is a connectionless protocol. It does not necessitate a handshake to establish

a connection, allowing for the rapid generation and transmission of many packets. The UDP Plain attack uses UDP's stateless feature, allowing the attacker to transmit packets to the target without establishing a connection beforehand. This facilitates the initiation of extensive attacks by utilizing botnets or other automated mechanisms. The effects of a User Datagram Protocol (UDP) Simple attacks can cause various problems, such as slowing down networks, degrading services, or completely denying access, depending on the strength and ability of the targeted infrastructure. Furthermore, due to the absence of inherent procedures in UDP for confirming the delivery of packets or maintaining their order, the attack might potentially lead to the loss of packets or their delivery in an incorrect sequence, causing additional communication disruption.

14. *MITM ARP Spoofing*: ARP spoofing, also known as ARP poisoning, is a type of Man-in-the-Middle (MitM) attack that allows attackers to intercept communication between network devices. The methodology of this attack comprises the subsequent stages: Initially, the attacker acquires entry into the network and does a comprehensive examination of the network to ascertain the IP addresses of a minimum of two devices, usually a workstation and a router. Afterwards, the attacker uses spoofing tools such as Arpspoof or Driftnet to send fake ARP answers. The falsified responses falsely claim that the attacker's MAC address matches both the router's IP addresses and the workstation, deceiving them into sending their traffic to the attacker's device instead of talking with each other directly. As a result, the ARP cache entries of the targeted devices are modified, redirecting their communication through the attacker's system. This allows the attacker to have access to all conversations. After successfully carrying out an ARP spoofing attack, the attacker can secretly listen in on conversations, except for those that are encrypted using methods like HTTPS; take control of sessions

by obtaining session IDs to gain unauthorized entry to logged-in accounts; tamper with communication by, for example, sending harmful files or redirecting users to malicious websites; and initiate Distributed Denial of Service (DDoS) attacks by supplying the MAC address of the target server instead of their own, resulting in overwhelming traffic if done across multiple IPs [32].

2.4 Concluding Remarks

This chapter included a comprehensive summary of the essential information required for the thesis. Initially, we delve into a comprehensive examination of the fundamental comprehension of the subject matter, encompassing a thorough analysis of crucial technical terminology linked to the topic, with a specific emphasis on attacks related to the Internet of Things (IoT). We extensively examine the specific aspects of IoT attack summaries used in training datasets. In Chapter 3, we will examine the work conducted in IoT attack detection, specifically concentrating on various machine learning and deep learning techniques, after comprehending the context and concepts provided in this chapter.

Chapter 3

Literature Review

3.1 Overview

The rapid proliferation of the Internet of Things across various businesses has increased security problems. The first part of this chapter provides an overview of the research on attacks on the Internet of Things. Then, based on this research, what were the various methods utilized to carry out Internet of Things attack detections?

- Machine Learning
- Deep Learning

3.2 Literature Review

Although there has been a lot of research on attacks targeting the Internet of Things (IoT), comprehensive literature on classifying these attacks is currently lacking. Therefore, our prior study introduced a new classification system that analyzes existing surveys and taxonomies [60].

Many academic publications thoroughly analyze many aspects of risks and attacks in the IoT field. The mentioned literary works [72] [55] [85] [29] [41] extensively

examine the classification of risks and intrusions linked to the Internet of Things (IoT). These papers primarily focus on two main categories: the architectural features of the Internet of Things (IoT) and the protocols and standards used in the IoT area. Although there is a wealth of literature on threats and attack taxonomies, it is essential to highlight that just a few studies specifically focus on viable solutions.

Table 3.1: Comparison of IoT Attack Surveys

Year	Title	# Attacks discussed	Taxonomy	Attack to Vulnerability Mapping	Detection methods
2020	A survey on privacy and security of Internet of Things [55]	12	Yes	No	Yes
2017	A survey of intrusion detection in Internet of Things [85]	0	Yes	No	Yes

Year	Title	# Attacks discussed	Taxonomy	Attack to Vulnerability Mapping	Detection methods
2019	Intrusion detection systems in the Internet of things: A comprehensive investigation [29]	9	Yes	No	Yes
2018	Internet of things security: A top-down survey [41]	0	Yes	No	Yes
2021	State-of-the-Art Review on IoT Threats and Attacks: Taxonomy, Challenges and Solutions [42]	59	Yes	No	Yes

Year	Title	# Attacks discussed	Taxonomy	Attack to Vulnerability Mapping	Detection methods
2020	Machine learning based solutions for the security of Internet of Things (IoT): A survey [73]	31	Yes	No	Yes
2018	IoT security: Review, blockchain solutions, and open challenges [39]	19	Yes	Yes	Yes
2018	A Comprehensive IoT Attacks Survey Based on a Building-blocked Reference Model [2]	51	Yes	No	No

Year	Title	# Attacks discussed	Taxonomy	Attack to Vulnerability Mapping	Detection methods
2020	A Comprehensive Survey on Attacks, Security Issues and Blockchain Solutions for IoT and IIoT [64]	22	Yes	No	No
2022	A Survey on IoT Security: Attacks, Challenges and Countermeasures[12]	17	Yes	No	Yes
2021	A Survey on Security Attacks and Solutions in the IoT Network[45]	17	Yes	No	No

Year	Title	# Attacks discussed	Taxonomy	Attack to Vulnerability Mapping	Detection methods
2021	A survey on Classification of Cyber-attacks on IoT and IIoT devices[66]	32	Yes	No	Yes
2023	A Comprehensive Survey on IoT Attacks: Taxonomy, Detection Mechanisms and Challenges [60]	149	Yes	Yes	Yes

However, it is essential to mention that the works discussed above do not offer any solutions to the risks and threats that emerge from the extensive use of pervasive technologies like blockchain (BC), fog computing (FC), edge computing (EC), and machine learning (ML). The authors have conducted surveys on diverse pervasive technologies for analyzing risks and attacks in a fragmented manner, as documented in the following sources: [22], [72], [75], [50], [74], and [42].

The authors of the study [15] did thorough research on security vulnerabilities in the Internet of Things (IoT) and presented an overview of machine learning methods

used to counter these attacks. The authors analyzed 78 publications published until 2017, focusing on the solutions, issues, and areas of research that have not yet been addressed in this field. The authors conducted a survey [82] in 2018 to explore several attack strategies that target the Internet of Things (IoT). These models include spoofing attacks, denial-of-service attacks, jamming, and eavesdropping. The authors also suggested possible security techniques to reduce these risks, such as IoT authentication, access control, malware detection, and safe offloading. The security solutions proposed in this study prominently included utilizing machine learning techniques [73].

3.3 Different techniques used to perform IoT attack detections

Identifying and mitigating these attacks is crucial to protecting IoT ecosystems' security. Several methods have been developed to address the problems of recognizing and responding to attacks. The following methods apply to the identification of attacks in the IoT domain. Also, Figure 3.1 provides a thorough overview of several techniques for detecting attacks in the Internet of Things (IoT) domain.:

- *Anomaly Detection*: Anomaly identification, also known as outlier or event detection, is the analytical process of identifying unusual situations inside a particular system. The anomaly detection algorithms assess incoming traffic at multiple levels, ranging from the IoT network level to the data centre. Anomaly detection is crucial because it enables the identification and analysis of anomalies within IoT data. Although rare, these anomalies can offer useful insights and practical information in many sectors, including healthcare, industry, finance, transportation, and energy. Anomaly detection in the Internet of Things (IoT) is employed in the betting and gambling sector to discover insider trading cases

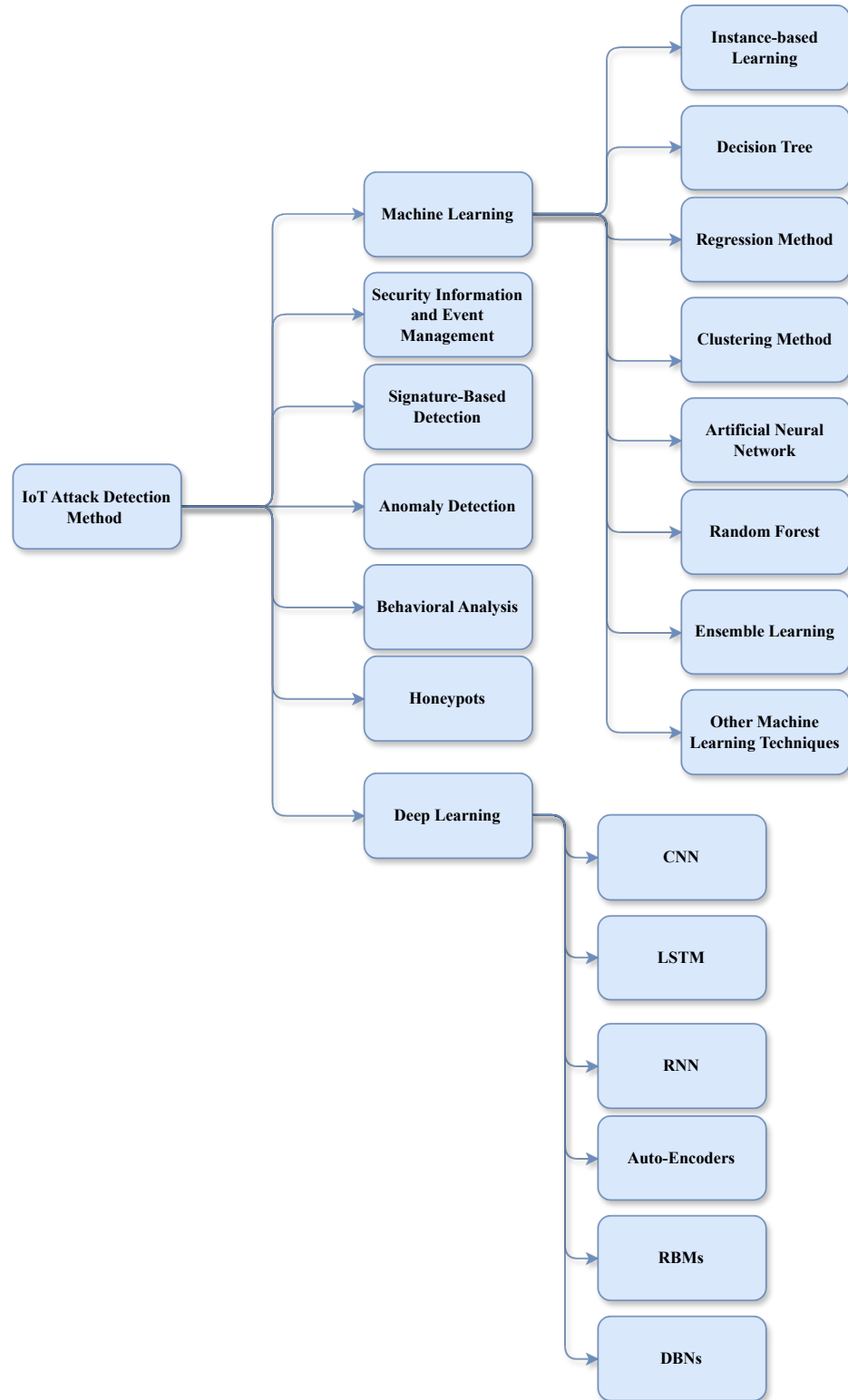


Figure 3.1: Overview of Available IoT Attack Detection and Identification Methods

by analyzing trade activity patterns [13].

- *Behavioural Analysis*: Dynamic code analysis refers to identifying and resolving issues with potentially harmful software within a physical or virtual setting. The program's source code is run with different test inputs to identify security vulnerabilities that may occur due to its code when interacting with other programs or systems. Dynamic analysis is a technique used to study the behaviours of attacks on IoT devices [47].

Utilizing behaviour analysis for detecting IoT attacks offers numerous benefits compared to static analysis. Dynamic analysis can identify known and zero-day threats by examining similar patterns of behaviour exhibited by several attackers. Dynamic analysis is performed by employing sandbox tools like Cuckoo Sandbox or CWSandbox, which allow for the monitoring of malware behaviours in real-time [47].

- *Signature-based Detection*: Signature-based detection (SGD) requires security professionals to create predefined rules or signatures to recognize known attack patterns. This method is especially efficient in identifying well-known attacks with signatures stored in the database. On the other hand, the database cannot identify unknown attacks without signatures.
- *Honey Pots*: A honeypot is a cybersecurity tool that creates a realistic and valuable network to lure potential attackers. It functions within a network environment that is both isolated and segregated. The aforementioned system can be seen as a simulated entity created to imitate a genuine system to attract potential attackers to interact with it. This method allows for the surveillance of the subsequent interaction between the attackers and the compromised device [59].
- *Security Information and Event Management*: Security Information and Event

Management (SIEM) is a security system that assists enterprises in detecting and addressing any security threats and weaknesses, preventing potential disruptions to business operations. Security Information and Event Management (SIEM) systems are essential tools for corporate security teams to detect anomalies in user behaviour. Moreover, these systems utilize artificial intelligence (AI) to simplify and automate specific labour-intensive operations associated with detecting possible threats and subsequent incident response [1].

- *Machine Learning*: Machine learning (ML) approaches are crucial in identifying and reducing IoT attacks. These techniques use different algorithms to recognize unusual patterns in network traffic and device activity. The algorithms are trained using extensive datasets, including regular and malicious IoT traffic. This allows them to learn about unique characteristics that distinguish various attacks. Machine learning models, such as decision trees, support vector machines, and random forests, can accurately categorize network traffic as benign or malicious by analyzing patterns they have learned. Furthermore, machine learning algorithms can adjust to changing attack techniques by consistently retraining on updated datasets, thus improving their ability to identify attacks as time goes on. Furthermore, intrusion detection systems that utilize machine learning may function in real-time, promptly notifying security administrators when they identify suspicious behaviours. This allows for quick reactions to possible attacks [60].
- *Deep Learning*: Deep learning (DL), a branch of machine learning (ML), provides sophisticated capabilities for identifying attacks in the Internet of Things (IoT) by automatically extracting complex features from raw data without requiring manual feature engineering. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are frequently used in deep learning-based

intrusion detection systems to secure the Internet of Things (IoT). CNNs are highly effective at identifying spatial patterns in network traffic data, but RNNs are skilled at capturing temporal relationships in sequences of device action. Using deep learning models, Internet of Things (IoT) security systems can attain enhanced precision in detecting and defending against advanced attack strategies. Deep learning algorithms can acquire hierarchical data representations, enabling them to identify intricate and nuanced attack patterns.

Furthermore, deep learning models can process and analyze large-scale datasets efficiently. They can adjust and perform effectively in dynamic Internet of Things (IoT) contexts. This makes them highly suitable for identifying familiar and new IoT attacks [60].

3.4 Concluding Remarks

To summarize, the swift growth of the Internet of Things (IoT) has resulted in a proportional rise in security difficulties across many sectors. Although there is a significant amount of study on IoT threats, there is a lack of comprehensive literature specifically focusing on their classification. In our earlier study, we presented a new classification method that combines existing surveys and taxonomies to fill this gap. Although many academic articles analyze the various aspects of risks and threats in IoT, there is a lack of focus on practical solutions. Furthermore, the widespread adoption of advanced technologies such as blockchain, fog computing, edge computing, and machine learning raises additional security issues that require further investigation. However, the initiatives mentioned in our assessment offer useful insights into possible remedies against attacks on the Internet of Things (IoT).

Within the domain of attack detection, numerous techniques have been devised to accurately identify and counteract Internet of Things (IoT) vulnerabilities. The meth-

ods include anomaly detection, behavioural analysis, signature-based detection, honeypots, security information and event management (SIEM), machine learning (ML), and deep learning (DL). Machine learning techniques utilize algorithms trained on large datasets to identify abnormal patterns in network traffic and device behaviour. This enables the early detection and reaction to potential threats. DL algorithms extract complex features from raw data, allowing for the precise and efficient identification of elaborate attack patterns. By adopting these sophisticated approaches, IoT security systems can enhance their ability to withstand emerging attack strategies, thereby protecting IoT ecosystems and their related assets from possible threats.

Chapter 4

Proposed Method

4.1 Motivation

This chapter presents a Convolutional Neural Network (CNN) combined with a Long Short-Term Memory (LSTM) model, enhanced with a Self-Attention mechanism, to identify Internet of Things (IoT) attacks in TCP network flow data. The suggested architecture enables efficient detection of IoT attacks without requiring feature engineering. The primary objective of our initial discussion is to examine the rationale behind the proposed strategy. In this context, we emphasize the fundamental significance of our proposed approach. In the next section, we present a comprehensive outline of the different components of our suggested approach.

Subsequently, we present an elaborate outline of the pre-processing stage and the suggested framework for detecting Internet of Things (IoT) attacks. The pre-processing step involves extracting features from raw benign and malicious pcap files by processing them using CIC Flowmeter. The extracted features are then saved into CSV files. Furthermore, the process involves constructing a model combining Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) with a self-attention network to detect attacks.

4.2 CICFlowmeter

CICFlowMeter is a tool that generates and analyzes network traffic flow. This tool enables the creation of bidirectional flows, where the first packet determines the direction of data transmission. As a result, over 83 statistical network traffic features, such as Duration, Number of packets, Number of bytes, and Length of packets, can be calculated independently for both the forward (source to destination) and backward (destination to source) directions.

Other capabilities encompass choosing features from the available feature list, incorporating new features, and managing the duration of flow timeout. The application generates a CSV file with six columns: FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort, and Protocol. The file contains over 83 network traffic analysis elements. It is essential to understand that TCP flows typically end when the connection is torn down using a FIN message, but a flow timeout ends UDP flows. The individual scheme can provide an arbitrary value for the flow timeout, such as 600 seconds for both TCP and UDP [43].

4.3 Data Pre-Processing

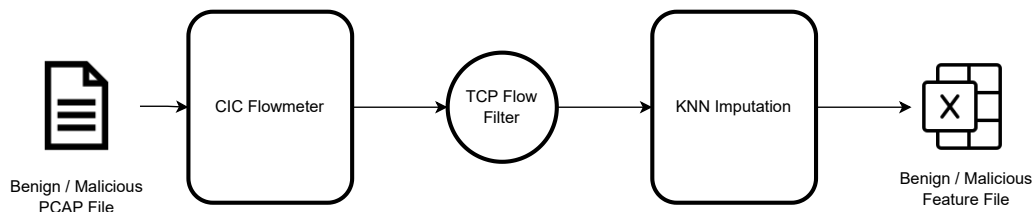


Figure 4.1: Data Pre-processing

During the pre-processing stage, we identify a dataset that includes raw benign and malicious pcap files to extract features. The CICflowmeter can be executed through the command line or within Java IDEs such as IntelliJ or Eclipse. Upon launching the application, we can choose between offline or online mode. The offline mode

enables us to import raw Pcap files and analyze them using the application, extracting features from the Pcap files and saving them as CSV files. We can derive 83 temporal statistical features from the Pcap files.

In addition, we have applied a filter to the dataset so that it would only include rows where the assigned protocol number is 6, which corresponds to the TCP protocol [8]. Due to missing values in our data set, we employed KNN Imputation to fill in these missing values.

K-Nearest Neighbors (KNN) is a machine learning technique for classification and regression tasks. Additionally, it can be utilized to impute missing data. The K-nearest neighbours (KNN) imputation approach involves identifying the K-nearest neighbours to the observation with missing data and subsequently imputing the missing values based on the non-missing values in those neighbours [3].

K-nearest neighbours (KNN) imputation is a highly favoured technique for replacing missing data in time series because it offers numerous significant benefits. Firstly, its non-parametric character enables it to accommodate a wide range of data distributions observed in time series without making assumptions about underlying data patterns. KNN imputation utilizes the principle that data points with similar characteristics typically have similar values. This method estimates missing values by considering the values of nearby points, thus maintaining the local structure of the data. Unlike approaches that presume linearity, KNN imputation is appropriate for time series data with nonlinear or complex connections between variables.

Furthermore, its ability to withstand outliers guarantees stability in irregular data points, as it prioritizes nearby clusters rather than overall patterns. KNN imputation can adapt to changing patterns in time series data by considering the nearest neighbours inside a sliding window. This makes it highly skilled at capturing evolving data dynamics. Furthermore, the straightforward application and little need for adjusting parameters, such as determining the number of neighbours (k) and distance metric,

enhance its attractiveness as a powerful and adaptable method for filling in missing values in time series datasets [71].

4.4 What are Convolutional Neural Networks?

The Convolutional Neural Network (CNN) is a feedforward neural network adept at automatically extracting features from data using convolution structures, eliminating the need for manual feature extraction as in traditional methods. Inspired by biological visual perception, CNN architecture mirrors the organization of the visual cortex: artificial neurons correspond to biological neurons, CNN kernels simulate different receptors detecting various features, and activation functions mimic the threshold for neural signal transmission. Loss functions and optimizers are designed to guide CNN learning. Compared to fully connected (FC) networks, CNN offers several advantages. Firstly, it employs local connections, where each neuron connects to only a few neurons in the previous layer, reducing parameters and speeding up convergence. Secondly, weight sharing allows groups of connections to share weights, reducing parameters. Finally, downsampling through pooling layers leverages image local correlation to reduce data volume while retaining crucial information, minimizing parameters, and discarding trivial features. These distinctive characteristics establish CNN as a prominent algorithm in the realm of deep learning [44].

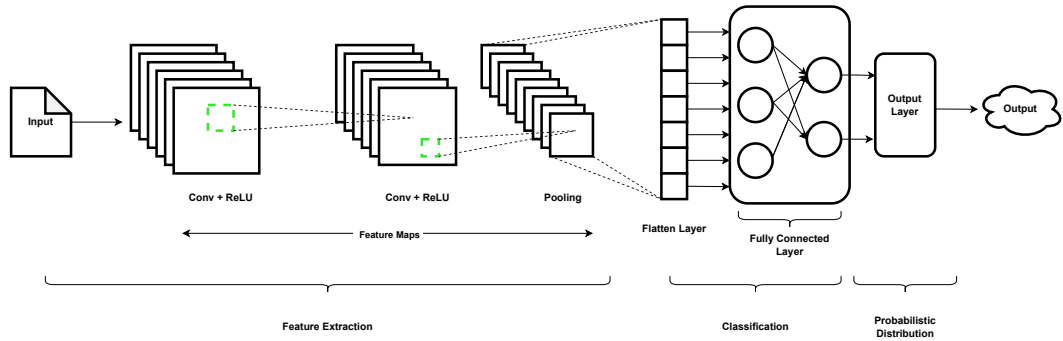


Figure 4.2: CNN Architecture

4.4.1 1D-CNN

A one-dimensional convolutional neural network (1D CNN) is a specific neural network that employs convolutional layers operating in one dimension. It analyzes data that follows a temporal or sequential pattern, where each data point consists of only one set of features. A one-dimensional convolutional neural network (1D CNN) applies one-dimensional filters to the data, extracting important patterns or features from specific parts of the input sequence.

4.4.2 2D-CNN

A 2D CNN is a prevalent form of convolutional neural network specifically created to analyze two-dimensional data, such as photographs. A 2D CNN utilizes convolutional layers to apply filters that operate in two dimensions on the input data. This allows the network to capture spatial hierarchies and patterns such as edges, textures, and forms. These characteristics become more conceptual and less concrete as we move up the network layers.

4.5 CNN Operations

Typically, eight components are required to construct a CNN model.

- *Convolution*
- *Activation Function*
- *Pooling*
- *Fully-Connected Layers*
- *Dropout Layer*
- *Batch Normalization Layer*

- *Loss Function*
- *Optimizer*

4.5.1 Convolution

Convolution is a crucial step in the process of extracting features. The results of convolution might be referred to as feature maps. When establishing a convolution kernel of a specific size, we will inevitably lose information at the boundary. Padding increases the input size by adding zero values, hence indirectly adjusting the size. In addition, the stride is used to regulate the density of convolution. As the stride increases, the density decreases. Following the convolution process, the feature maps include many features, which increases the risk of encountering the overfitting issue. To eliminate redundancy, pooling (also known as downsampling) is suggested as a solution, which includes techniques such as max pooling and average pooling [44].

The basic 2D convolution operation is defined as follows:

$$Y(i, j) = \sum_m \sum_n X(i + m, j + n) \cdot K(m, n) \quad (4.1)$$

Where:

- $Y(i, j)$ is the output of the convolution at position (i, j) .
- $X(i + m, j + n)$ represents the pixel values of the input image.
- $K(m, n)$ is the kernel or filter applied to the image.
- The summations over m and n traverse all the rows and columns of the kernel K , respectively.

When incorporating stride and padding, the convolution formula is modified to accommodate these parameters, enabling control over the output size and the field of

view of the convolution operation:

$$Y(i, j) = \sum_m \sum_n X(s \cdot i + m - p, s \cdot j + n - p) \cdot K(m, n) \quad (4.2)$$

Where:

- s is the stride, which dictates the step size of the filter as it slides over the image.
- p is the padding, which involves adding layers of zeros outside the original image to preserve spatial dimensions.

4.5.2 Activation Function

Table 4.1: Activation Functions [10]

Activation Function	Formula
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$
Tanh	$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
ReLU (Rectified Linear Unit)	$\text{ReLU}(x) = \max(0, x)$
Leaky ReLU	$\text{Leaky ReLU}(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.01x, & \text{otherwise} \end{cases}$
ELU (Exponential Linear Unit)	$\text{ELU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(e^x - 1), & \text{otherwise} \end{cases}$

Activation functions in Convolutional Neural Networks (CNNs) play a critical role by introducing non-linearity into the network, allowing it to learn complex patterns in the data. Without activation functions, a CNN would essentially become a linear model, incapable of handling the intricacies and nuances required for tasks such as image recognition or natural language processing. The most commonly used activation function in CNNs is the Rectified Linear Unit (ReLU), favoured for its computational simplicity and ability to maintain the gradient flow during training, thus helping to mitigate the vanishing gradient problem. ReLU functions by outputting the input

Table 4.2: Activation Functions Part 2 [10]

Activation Function	Pros	Cons
Sigmoid	Smooth gradient, Outputs between 0 and 1	Susceptible to vanishing gradient problem, Outputs not zero-centered
Tanh	Outputs between -1 and 1, Zero-centered output	Susceptible to vanishing gradient problem
ReLU	Computationally efficient, Avoids vanishing gradient problem	Prone to dying ReLU problem (neurons become inactive)
Leaky ReLU	Addresses dying ReLU problem, Non-zero gradient for negative inputs	More complex than ReLU, Not zero-centred output
ELU	Avoids dying ReLU problem, Zero-centered output	More computationally expensive than ReLU

directly if it is positive. Otherwise, it outputs zero. Other activation functions like sigmoid, tanh, and Leaky ReLU are also used, each with advantages depending on the network architecture and specific application. These functions are applied at specific layers throughout the CNN to help the model differentiate and correctly classify the input signals into outputs [44].

4.5.3 Pooling

Pooling layers in Convolutional Neural Networks (CNNs) reduce the spatial dimensions of the input feature maps, decreasing the number of parameters and computation required. It also helps detect features that are invariant to scale and orientation changes. This subsampling step improves the network's efficiency and robustness by abstracting higher-level features while retaining the most essential information. The most common types of pooling are max pooling and average pooling. Max pooling returns the maximum value from each cluster of neurons at the prior layer, effectively highlighting the most prominent features. In contrast, average pooling calculates the average value, smoothing out the feature responses. By reducing the resolution of the

Table 4.3: Pooling Functions [36]

Pooling Function	Formula	Pros	Cons
Max Pooling	$y = \max(x)$	Preserves dominant features, Translation invariance	May discard less dominant features, Can lead to overfitting
Average Pooling	$y = \frac{1}{n} \sum_{i=1}^n x_i$	Simple and efficient, Reduces noise	Can lose important details, Not robust to outliers
Global Max Pooling	$y = \max(x)$ over all elements	Captures most important feature, Reduces dimensionality	Ignores spatial information, Not suitable for tasks requiring spatial information
Global Average Pooling	$y = \frac{1}{n} \sum_{i=1}^n x_i$ over all elements	Reduces computation complexity, Less prone to overfitting	Loss of spatial information, Not suitable for tasks requiring spatial information

feature maps, pooling layers also help prevent overfitting by providing an abstracted form of the representation. These layers are typically placed between successive convolutional layers and play a pivotal role in the architecture of deep learning models designed for tasks like image and video recognition [40].

4.5.4 Fully-Connected Layers

Fully-Connected layers in Convolutional Neural Networks (CNNs) are crucial components that typically come after the convolutional and pooling layers. These layers are called "fully connected" because every neuron in a fully connected layer is connected to all the neurons in the previous layer. The main function of these layers is to perform high-level reasoning by integrating the localized features extracted by the earlier convolutional and pooling layers into the final decision-making process of the network. This is where the abstract features from the entire image or input are used to classify or make predictions about the input data. The fully-connected layers map

the extracted features into the final output, such as classification labels. This makes them crucial for tasks like image recognition, where you must determine the presence of specific features across the whole image. These layers often use softmax activation functions in classification tasks to convert the output into probability distributions over predicted output classes [19] [44].

4.5.5 Dropout Layer

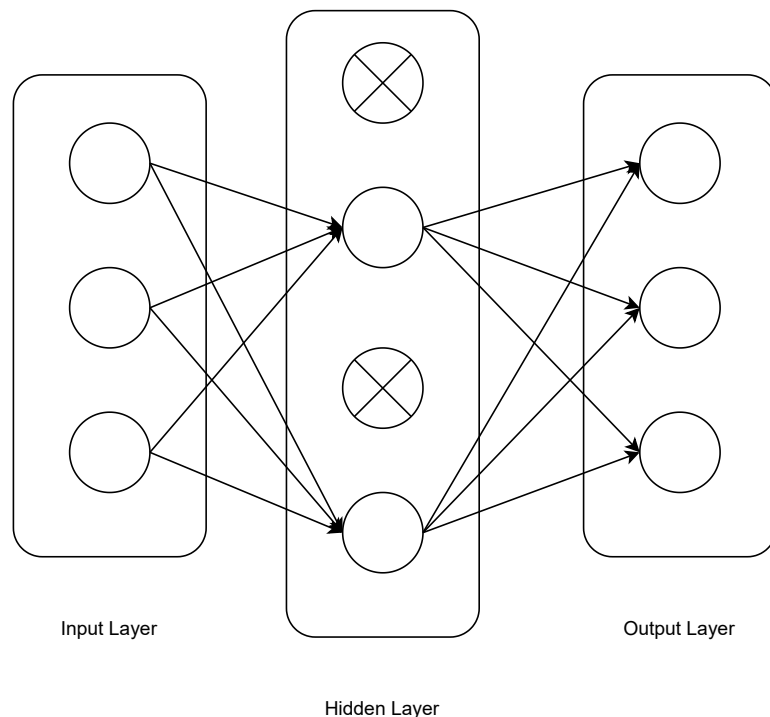


Figure 4.3: Dropout Layer

Dropout layers in Convolutional Neural Networks (CNNs) are a regularization approach specifically developed to mitigate overfitting, a prevalent issue in deep learning models, especially those with a substantial number of parameters. During the training phase, dropout selectively turns off a random subset of neurons in the layer to which it is applied. This means these neurons do not participate in the forward pass, and their weights are not adjusted during the backward pass. The implementation of random deactivation compels the network to acquire more resilient characteristics that

do not depend on a limited group of neurons. This diminishes the model’s reliance on specific traits and encourages a more universal learning pattern. By employing this technique, dropout guarantees that the neural network performs highly on new, unobserved data rather than solely on the data it was trained on. The dropout rate, which refers to the likelihood of each neuron being deactivated, is a hyperparameter that can be tuned to enhance performance. Dropout is typically more prevalent in fully-connected layers than convolutional layers [23].

4.5.6 Loss Function

Table 4.4: Loss Functions [6]

Loss Function	Formula	Pros	Cons
Cross-Entropy Loss	$-\sum_{c=1}^M y_{o,c} \log(p_{o,c})$	- Directly models probability distributions - Highly effective for classification	- Can be numerically unstable without proper implementation
Mean Squared Error (MSE)	$\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$	- Easy to understand and implement - Differentiable function	- Poor performance in classification - Penalizes outliers heavily
Mean Absolute Error (MAE)	$\frac{1}{N} \sum_{i=1}^N y_i - \hat{y}_i $	- Less sensitive to outliers compared to MSE - Easy to understand	- Gradient can be constant, which may affect convergence

Loss functions in Convolutional Neural Networks (CNNs) are critical components that measure the discrepancy between the predicted outputs of the network and the actual target values during training. The choice of a loss function depends on the specific task the CNN is designed to perform. For classification problems, a common loss function is the categorical cross-entropy, which calculates the difference between the predicted probability distribution over classes and the actual distribution (typically represented

as a one-hot encoded vector). For regression tasks, mean squared error (MSE) or mean absolute error (MAE) are commonly used, as they measure the average of the squares or absolute differences between predicted and actual values, respectively. The role of the loss function is to provide a quantitative assessment of model performance, which the training process seeks to minimize through gradient descent and backpropagation. By continuously improving the model parameters to reduce the loss, the CNN learns to make more accurate predictions, effectively tuning itself to the complexity of the data it processes [44].

4.5.7 Optimizer

Table 4.5: Optimizers [28] [20]

Optimizer	Formula	Pros	Cons
SGD	$w_{t+1} = w_t - \eta \nabla J(w_t)$	- Simple and easy to understand - Effective in large-scale datasets	- Slow convergence - Sensitive to hyperparameters
Adam	$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t$	- Fast convergence - Automatically adjusts the learning rate	- Computationally more intensive - Potential for non-convergence on non-convex functions
RMSprop	$w_{t+1} = w_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$	- Diverges less - Good in on-line and non-stationary settings	- Less popular than Adam - Sensitive to initialization
AdaGrad	$w_{t+1} = w_t - \frac{\eta}{\sqrt{G_t + \epsilon}} g_t$ where G_t is the sum of the squares of the past gradients	- Handles sparse gradients well - Good for problems with large and sparse datasets	- Gradient scaling can cause early stopping of learning

Optimizers in Convolutional Neural Networks (CNNs) are algorithms or methods used

to change the attributes of the neural network, such as weights and learning rate, in order to reduce the losses. Optimizers guide the training process by deciding how to update weights based on the gradients of the loss function with respect to those weights. Common optimizers include Stochastic Gradient Descent (SGD), Adam, and RMSprop, among others. SGD is simple and has been the traditional choice, where each update is performed using only a subset of all data (mini-batch) to compute the gradient, making it computationally efficient. Adam (Adaptive Moment Estimation) combines the benefits of two other extensions of SGD, AdaGrad and RMSProp, to handle sparse gradients on noisy problems. RMSprop adjusts the learning rate for each parameter, dividing the learning rate for weight by a running average of the magnitudes of recent gradients for that weight. These optimizers differ mainly in how they use the amount of past weight update information (momentum) and how they adapt the learning rate during training, making them suited to different types of neural networks and convergence problems [44].

4.6 What are Residual blocks?

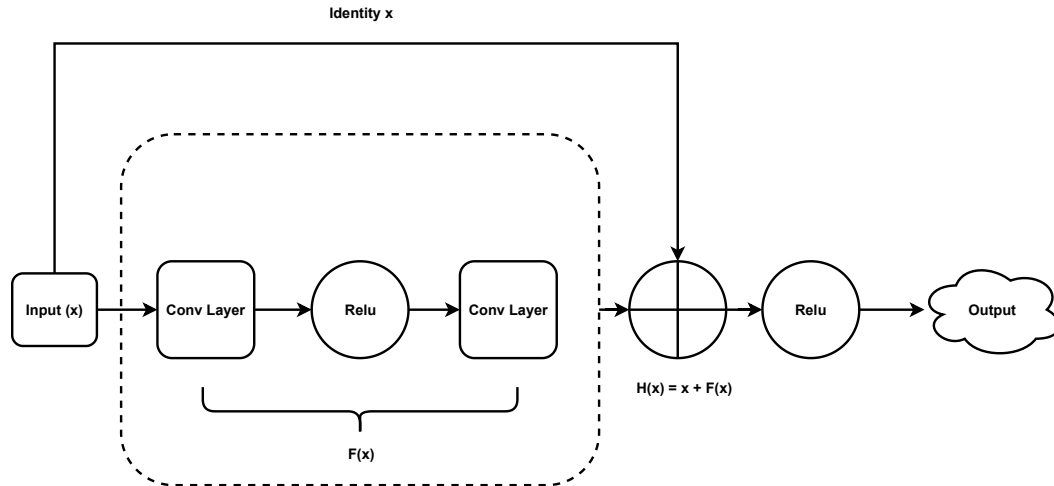


Figure 4.4: Residual Block Cell

Residual blocks, also known as residual connections or shortcut connections, are in-

tegral components of deep neural networks, particularly in architectures like ResNet (Residual Networks), aimed at tackling the vanishing gradient problem prevalent in very deep networks during training. These blocks incorporate skip connections that enable the output of one or more layers to bypass subsequent layers and be added directly to the output of deeper layers. This design allows the model to learn residuals, representing the difference between the desired output and the input to a specific layer and facilitates the learning of identity mapping. By providing a direct path for gradient flow during backpropagation, residual blocks mitigate the vanishing gradient problem, making it feasible to train exceedingly deep networks. Consequently, they contribute to easier optimization, improved generalization performance, and faster convergence, rendering them a standard and indispensable element in constructing deep neural network architectures across diverse domains, including computer vision, natural language processing, and speech recognition.

4.7 What are Longest Short Term Memory Network?

Long Short-Term Memory Networks (LSTMs) are a specialized kind of Recurrent Neural Network (RNN) designed to address the problem of learning long-term dependencies in sequence data. Traditional RNNs often struggle with vanishing or exploding gradients as the sequence length increases, hampering their ability to learn from data where distant past information is crucial for predicting future states. LSTMs overcome this challenge through their unique structure, which includes memory cells and multiple gates—namely, the input, forget, and output gates. These gates manage the flow of information into and out of the memory cell, effectively allowing the network to retain or discard information dynamically. This selective memory capability enables LSTMs to maintain long-range dependencies, making them highly effective for a wide

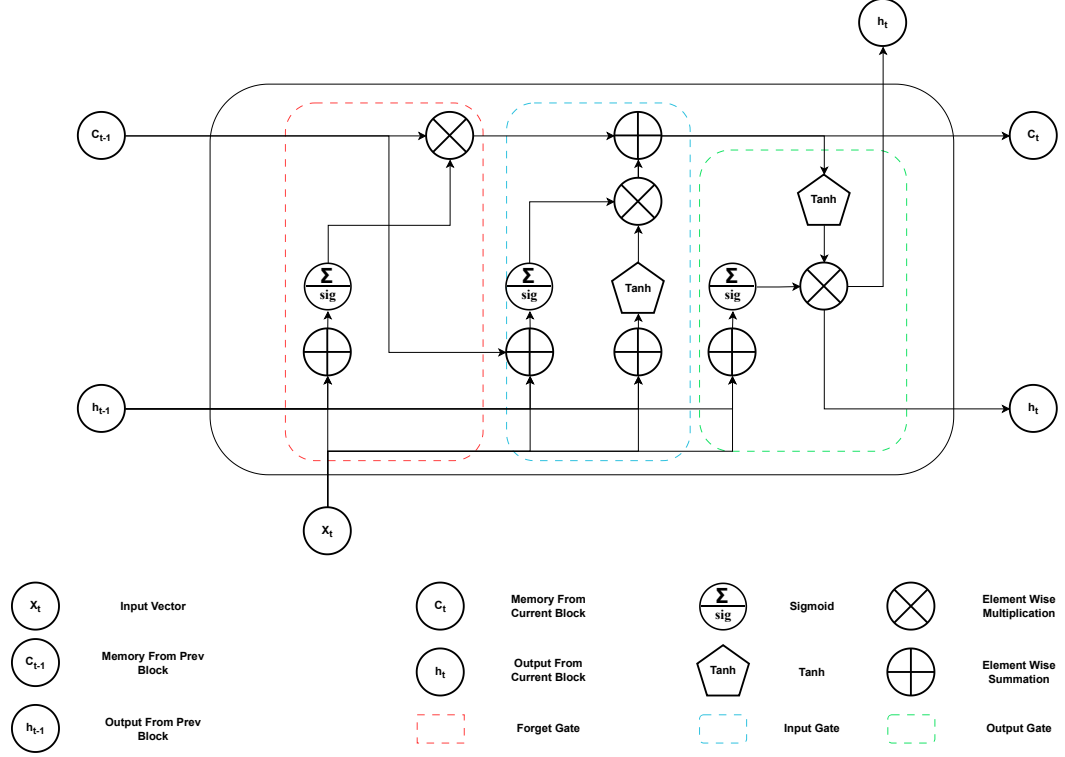


Figure 4.5: LSTM Cell

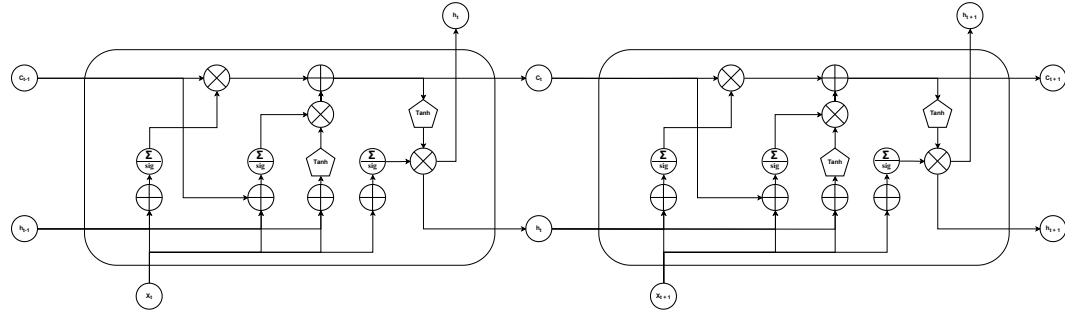


Figure 4.6: LSTM Network

range of sequential tasks such as natural language processing, speech recognition, and time series prediction. Their ability to remember and forget information selectively across long sequences makes them particularly powerful in fields where context and history significantly influence current outcomes [61].

The LSTM updates for timestep t given inputs x_t (current input), h_{t-1} (previous output), and C_{t-1} (previous cell state) are as follows:

Forget Gate:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Input Gate:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Cell State Update:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Output Gate:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

4.8 What are Attention Layers?

An attention function is a mathematical operation that takes a query and a set of key-value pairs as input and produces an output. The query, keys, values, and result are all represented as vectors in this operation. The output is calculated by taking a weighted total of the values. The weight allocated to each value is determined by a compatibility function that compares the query with the relevant key [76].

The concept of attention in Long Short-Term Memory Networks (LSTMs) enhances these models by allowing them to focus selectively on parts of the input sequence that are most relevant to the task, improving their ability to manage long-range dependencies in complex sequences. Attention mechanisms dynamically assign weighting to different inputs at each time step, indicating the importance of each part of the data for predicting the current output. In practice, this means that instead of treating all parts of the input equally, the model learns to pay "attention" to specific segments of

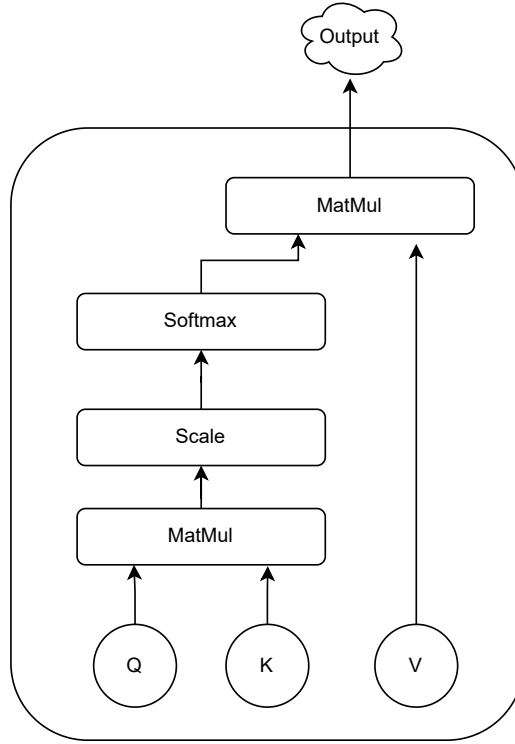


Figure 4.7: Self Attention Mechanism

the input that are more informative for the current decision or prediction. This capability is particularly useful in tasks such as machine translation, where the relevance of input words can vary significantly depending on the context within the sequence. By integrating attention with LSTMs, the model not only retains information over long periods but also adapts its focus according to the evolving semantic importance of different parts of the input, leading to improved performance on tasks requiring sophisticated contextual interpretation [68] [49] [38] [48].

4.8.1 Self-Attention

The self-attention mechanism simulates the human brain's attention by allocating resources to focus on relevant rather than irrelevant information. The self-attention

mechanism contributes to the variance in the relevance of hidden information that the LSTM network does not recognize by assigning different weights to hidden features at different phases. Thus, Self Attention Mechanism retrieves advanced traits while improving long-term dependency [84].

Input Representation

Let $X \in \mathbb{R}^{n \times d}$ be the input matrix where n is the sequence length and d is the dimension of the embeddings [18].

Linear Transformations

The input X is linearly transformed into three matrices: Query (Q), Key (K), and Value (V) using learned weight matrices W_Q , W_K , and W_V .

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

Where:

$$W_Q \in \mathbb{R}^{d \times d_k}$$

$$W_K \in \mathbb{R}^{d \times d_k}$$

$$W_V \in \mathbb{R}^{d \times d_v}$$

d_k and d_v are the dimensions of the key and value vectors respectively.

Dot-Product Attention

The attention scores are calculated by taking the dot product of the query Q with the key K transposed.

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

Breaking it down into steps:

Score Calculation

$$\text{scores} = QK^T$$

Scaling

$$\text{scaled_scores} = \frac{QK^T}{\sqrt{d_k}}$$

Scaling the scores by $\sqrt{d_k}$ helps stabilize the gradients.

Softmax

$$\text{attention_weights} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

The softmax function converts the scores into a probability distribution.

Weighted Sum

$$\text{context_vector} = \text{attention_weights} \cdot V$$

Explanation

- **Input Representation:** The input X is a matrix of shape (n, d) , where each row represents a token in the sequence, and each column represents a feature of the embedding.
- **Linear Transformations:** The input matrix X is transformed into query Q , key K , and value V matrices using weight matrices W_Q , W_K , and W_V . These transformations allow the model to learn different representations for querying, keying, and valuing the input data.
- **Dot-Product Attention:**
 - **Score Calculation:** The dot product QK^T results in a score matrix that represents the similarity between each query and key.
 - **Scaling:** Dividing the scores by $\sqrt{d_k}$ helps to prevent the softmax function from having extremely small gradients.

- **Softmax**: The softmax function normalizes the scores into probabilities, indicating each token’s importance.
- **Weighted Sum**: The context vector is computed as the weighted sum of the value vectors, where the weights are the attention scores. This context vector captures the relevant information from the entire sequence for each token.

This mechanism allows the model to focus on different parts of the input sequence dynamically, depending on the context, which is particularly useful for tasks involving long-range dependencies [18].

4.9 What are SHAP values?

SHAP (SHapley Additive exPlanations) values offer a method for elucidating the outcomes of any machine learning model by employing a game-theoretic approach that gauges each player’s contribution to the final result. Each feature receives an important value in machine learning, denoting its impact on the model’s output. These SHAP values reveal the individual influence of each feature on every prediction, assess the relative significance of each feature, and ascertain the model’s reliance on feature interactions [9] [67]. The SHAP value (ϕ) for a feature j in a prediction model f is given by the formula:

$$\phi_j(f) = \sum_{S \subseteq F \setminus \{j\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f(S \cup \{j\}) - f(S)]$$

Where:

- F is the set of all features.
- S is a subset of features excluding j .

- $f(S)$ is the model’s output using only the features in set S .
- $|S|$ and $|F|$ are the cardinalities of sets S and F , respectively.
- $\phi_j(f)$ is the SHAP value for feature j .

Widely utilized in machine learning, SHAP values afford a consistent and unbiased explanation of how each feature influences the model’s predictions. They derive from game theory principles, attributing importance metric to each feature, where positive SHAP values denote a positive influence on predictions, while negative values imply a negative impact, with magnitude indicating the strength of the effect [67] [9] [46] [51].

Notably, SHAP values are model-agnostic, making them applicable to diverse machine learning models such as linear regression, decision trees, random forests, gradient boosting models, and neural networks. These values possess several advantageous properties, including additivity, local accuracy, missingness, and consistency. Their additivity allows for independent computation of each feature’s contribution to the prediction, facilitating efficient computation even for high-dimensional datasets [67] [9][46] [51].

Furthermore, SHAP values provide an accurate and localized interpretation of a model’s prediction for a given input while remaining robust to missing or irrelevant features. Importantly, they offer a consistent interpretation of a model’s behaviour, ensuring stability even in the face of changes in model architecture or parameters. Overall, SHAP values furnish a reliable and objective means to gain insights into a machine learning model’s prediction mechanisms and the features exerting the greatest influence [9] [46] [51].

4.9.1 SHAP Feature Importance Scores

SHAP (SHapley Additive exPlanations) feature importance provides a comprehensive method for understanding the impact of individual features on a machine learning model's predictions. Unlike traditional methods like permutation importance, which assesses feature importance by measuring the decrease in model performance when a feature's values are randomly permuted, SHAP values consider the interaction between features and provide a more nuanced understanding of their influence on predictions [46] [51].

While permutation importance offers a straightforward measure of feature importance, SHAP values capture the contribution of each feature in the context of other features, allowing for a more accurate and interpretable assessment of their impact. In situations where feature interactions are significant in model predictions, SHAP feature importance provides more insightful and reliable results. However, permutation importance may suffice for simpler models with fewer interactions between features and could be computationally less intensive. Ultimately, the choice between SHAP feature importance and permutation importance depends on the complexity of the model and the importance of capturing feature interactions for accurate interpretation [51].

While SHAP values typically provide local explanations for individual predictions, aggregating these values across a dataset offers insights into the global importance of each feature. Global feature importance quantifies the overall impact of features across all predictions, measuring how significant each feature is in the model's decision-making process.

The global importance of a feature j is calculated by summing the absolute SHAP values for that feature across all data points in the dataset:

$$\text{Global Importance}(j) = \sum_{i=1}^N |\phi_j^{(i)}|$$

Where:

- N represents the total number of data points in the dataset.
- $\phi_j^{(i)}$ denotes the SHAP value for feature j at the i -th instance.
- The absolute values of SHAP values are summed to account for both positive and negative contributions uniformly.

4.10 Proposed Mechanism

The proposed approach consists of two distinct steps. The initial phase involves the conversion of data, extraction of features, generation of datasets for training and testing the model, and the development of the model itself. The second module involves training the model using the training dataset, incorporating all features, and determining feature importance by utilizing SHAP values. Subsequently, the model is retrained using a decreased set of features while maintaining comparable performance metrics to the original model. Therefore, determining the optimal characteristics to use in training the model

4.10.1 Pre-processing - Stage I

This section will provide a detailed analysis of the components involved in the pre-processing stage technique, and Figure 4.1 provides a high-level overview of the pre-processing workflow.

- *Data Collection*: In data acquisition, we necessitated benign and malicious pcap files sourced from an IoT attack dataset, significantly facilitating the advancement of security analytics applications within IoT environments. Specifically,

we employed the CIC IoT dataset 2023 for training, validation, and testing. Additionally, after completing the training phase, we utilized eight additional IoT attack datasets to evaluate the efficacy of the developed model [53].

- *Data Conversation and Feature Extraction:* As previously outlined in Section 4.2, we employed CIC flower, a network traffic flow generator and analyzer, to extract temporal statistical features from raw pcap files. Subsequently, we extracted over 83 features from these files and stored them in CSV format, facilitating their utilization in both model training and testing phases.
- *Data Filtration, Imputation and Creation:* We implemented a filtering mechanism on the dataset, specifically retaining rows associated with an assigned protocol number of 6, indicative of the TCP protocol [8]. To address missing data within our dataset, we opted for KNN Imputation to complete these missing values. Subsequently, by merging these CSV files, we synthesized the necessary dataset.
- *Model Creation:* We examine the components utilized in the model's construction during this stage. This model incorporates convolutional layers, residual blocks, LSTM layers with an attention mechanism, and some final processing and output layers. Let's break down each module's purpose and workings:

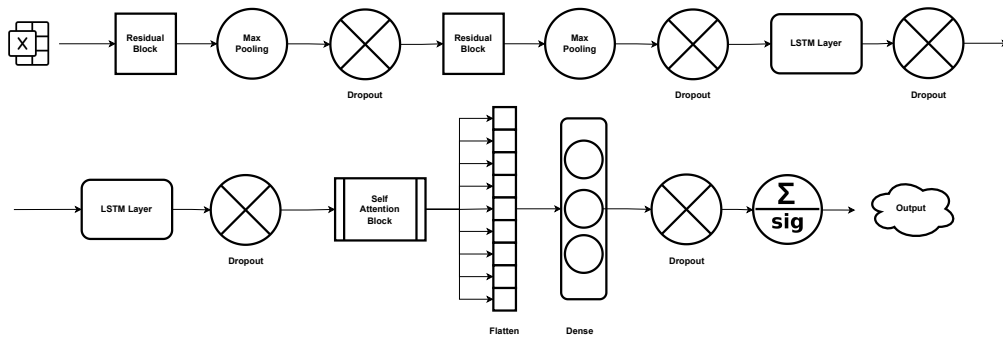


Figure 4.8: Model Architecture

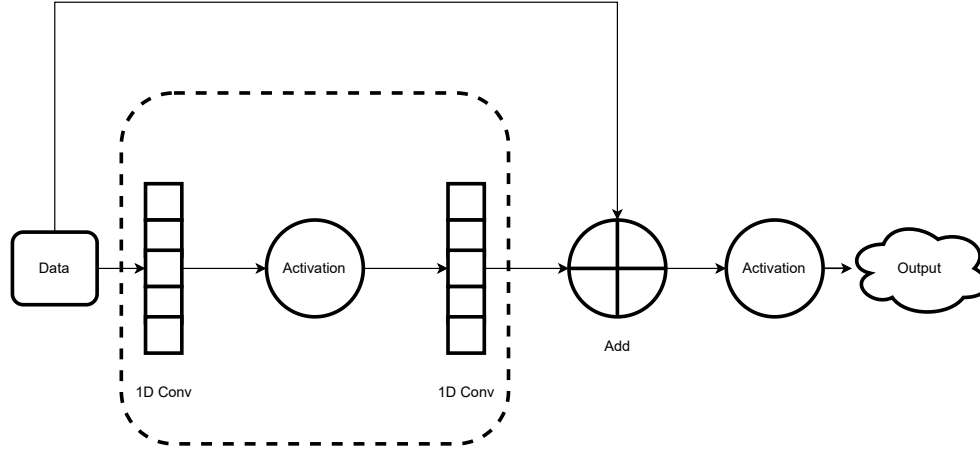


Figure 4.9: Residual Block

- *Input layer*: This layer defines the input shape required to process the data for model training and testing. For our model, we have identified the input shape for the input layer to be $(feature_len, 1)$, where `feature_len` is the number of columns in one batch.
- *Residual Blocks*: Start the model with CNN based residual block which is used to deepen the network without losing the ability to train effectively.
- *Max Pooling*: Reduces the spatial dimensions of the output from convolutional layers, summarizing features.
- *Dropout*: Applied after pooling and LSTM layers to prevent overfitting by randomly dropping units during training.
- *LSTM Layers*: Processes the output of convolutional layers to learn from the temporal patterns in the data. `return_sequences = True` keeps the time dimension for attention processing.
- *Attention Layers*: Applied to the output of the LSTM layers to focus the model's learning on important temporal elements.
- *Flatten*: Converts the multi-dimensional output of the attention layer into a single-dimensional array suitable for input to the fully connected layer.

- *Dense Layer*: A dense layer serves as the intermediate stage where the spatially distributed features extracted by convolutional , pooling layers and LSTMs are flattened into a single vector representation. By doing so, Dense layers enable the network to learn high-level abstractions and relationships among the extracted features, making them more suitable for classification or regression tasks. These intermediate Dense layers typically incorporate non-linear activation functions like ReLU to introduce non-linearity and capture complex patterns in the data. Additionally, they may include dropout or batch normalization layers to regularize the network and prevent overfitting.
- *Output Layer*: A dense layer that outputs the final prediction of the model. The activation function is chosen based on the task. Since we are performing binary classification, we are using the sigmoid function.

The Figure 4.8 and Figure 4.9 provide a high-level overview of the model architecture. This architecture is particularly effective for complex sequence modelling tasks that benefit from spatial feature extraction and the ability to remember and emphasize important parts of the sequence data over time.

4.10.2 Proposed Architecture - Stage II

- *Hyperparameter Tuning*: Hyperparameter tuning, also known as hyperparameter optimization, refers to selecting the optimal hyperparameters for a machine learning model to maximize its performance on a given dataset. Hyperparameters are parameters set before the learning process begins and control aspects such as the complexity of the model, the regularization strength, and the learning rate. Unlike model parameters learned during training, hyperparameters are not learned from the data and must be specified by the user [65].

Grid search is a hyperparameter tuning technique to find the optimal combination of hyperparameters for a machine learning model. It involves defining a grid of hyperparameter values and exhaustively searching through all possible combinations of these values to identify the combination that yields the best performance on a chosen evaluation metric.

Using grid search, we have successfully determined five optimal combinations of hyperparameters for training, validation, and testing. Among the five options, we have selected the hyperparameter configuration that performs the best for calculating feature importance, retraining the model, and compressing it. This selection was based on testing it on external datasets. Below is the Table 4.6 of hyperparameters utilized for training the model.

Table 4.6: Hyperparameters List

No	Activation	Losses	Optimizers	Batches	Epochs	Shuffles
1	Relu	Binary Cross- entropy	Adam	32	20	True
2	Relu	Binary Cross- entropy	Adam	16	20	True
3	Leaky Relu	Binary Cross- entropy	Adam	16	20	True
4	Relu	Binary Cross- entropy	RMSProp	16	20	True

5	Relu	Binary Cross- entropy	RMSProp	8	20	True
---	------	-----------------------------	---------	---	----	------

- *Model Training and Testing:* To facilitate model training, we divided the dataset into two subsets - training and testing - in an 80:20 ratio. This was achieved using the *train_test_split* function from the sklearn library. In addition, we partitioned the data in the training subset into separate training and validation sets using an 80:20 ratio. This division assisted in training the model according to the method described in the reference [62].
- *SHAP Feature Importance:* As outlined in Section 4.9.1, we employ SHAP feature importance scores to identify the most impactful features in our model. Subsequently, we refine the model by training it with a reduced feature set, guided by insights drawn from the SHAP feature importance graph, particularly focusing on the differences between adjacent bar levels. The SHAP feature importance graph in Figure 5.5 provides a visual representation of these insights.
- *Model Retraining and Model Compression:* After identifying the reduced set of features, we build and train the model on it. Next, we compare the performance characteristics of the initial and reduced set models. If the performance of the reduced model is comparable to the initial model, we will use the reduced model for deployment.

4.11 Concluding Remarks

This chapter has presented a novel method integrating Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and attention mecha-

nisms aimed at improving the detection of Internet of Things (IoT) attacks within TCP network flows. The proposed method leverages the robust feature extraction capabilities of CNNs, the sequence learning strengths of LSTMs, and the focused contextual insights provided by self-attention mechanisms, creating a sophisticated framework.

This chapter emphasises the importance of an efficient data pre-processing stage, utilizing the CIC Flowmeter tool for accurate feature extraction from network traffic data and applying sophisticated machine learning techniques for feature selection and model optimization. The integration of SHAP values for understanding feature importance further underscores our approach’s commitment to transparency and interpretability in machine learning models, which is crucial in the sensitive context of cybersecurity.

Finally, in Chapter 5, we will discuss the experiments and results of the proposed framework. The proposed architecture has demonstrated the potential for high accuracy and adaptability in identifying and responding to dynamic threats in the IoT landscape. The results suggest that by reducing the reliance on extensive feature engineering and manual intervention, the model can efficiently handle real-world data and adapt to new, previously unseen attack vectors.

Chapter 5

Experiments & Results

5.1 Overview

This chapter begins with a detailed overview of the experimental framework and the dataset utilized for conducting the experiments and obtaining the results. Subsequently, an exhaustive examination of the features used in the proposed method is performed, with a comprehensive list of all the features included. As outlined in Chapter 4, the chapter then demonstrates the experimentation and results of the proposed CNN + LSTM with a self-attention-based deep learning technique.

Further in the chapter, the implementation results of the proposed model are presented in detail, along with a thorough discussion of the performance metrics on externally augmented datasets. Finally, the outcomes of these experiments validate the effectiveness of the proposed system.

5.2 Experimental Setup

The implementation and execution of all the modules were processed on Windows 11, 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz 2.30 GHz, 64 GB RAM with a graphic card of NVIDIA Geforce RTX 3060 4GB. All the data preprocessing and

feature extraction were implemented using Python scripts and Python public libraries.

5.3 Datasets

The datasets that are augmented during preprocessing for training, validation, and model testing are listed below in Table 5.1 in detail.

Table 5.1: Augmented Datasets

Year	New Dataset Name	Original Dataset Name	Malicious Activities	Real / Simulated	Dataset Size
2023	CIC-BCCC-NRC-IoT-2023	CIC IoT 2023	Yes	Real	108.58 GB
2023	CIC-BCCC-NRC-ACI-IoT-2023	ACI IoT Network Traffic Dataset 2023	Yes	Real	338.9 MB
2024	CIC-BCCC-NRC-IoMT-2024	CIC IoMT 2024	Yes	Real	19.25 GB
2022	CIC-BCCC-NRC-Edge-IIoTset-2022	Edge-IIoTset	Yes	Real	1.55 GB
2022	CIC-BCCC-NRC-IoT-2022	CIC IoT 2022	Yes	Real	60.6 MB

2019	CIC-BCCC-NRC-IoT-HCRL-2019	IoT Network Intrusion Dataset	Yes	Real	46.4 MB
2020	CIC-BCCC-NRC-MQTTIoT-IDS-2020	MQTT-IoT-IDS-2020	Yes	Simulated	1.45 GB
2021	CIC-BCCC-NRC-TONIOT-2021	TON IoT 2021	Yes	Real	696.3 MB
2022	CIC-BCCC-NRC-UQ-IoT-2022	UQ IoT 2022	Yes	Real	2.31 GB

Below is a summary of datasets augmented during preprocessing for training, validation, and model testing.

- *ACI IoT Network Traffic Dataset 2023*: The principal aim of this study is to introduce a novel dataset tailored explicitly for machine learning (ML) applications in the domain of Internet of Things (IoT) network security. The research provides a distinctive and realistic dataset for training and assessing ML models designed for IoT network environments. By addressing an existing gap in available resources, this dataset aims to drive progress in ML-based solutions, thereby enhancing the security of IoT operations. The experimentation phase was conducted within the Internet of Things Research Lab (IoTRL) at the Army Cyber Institute (ACI), meticulously crafted to replicate a realistic home environment. The IoTRL mirrors the conditions of a typical home IoT network,

encompassing a diverse range of both wired and wirelessly connected devices, including various IoT devices and conventional networked devices commonly found in home settings. Home Assistant, free and open-source software for home automation, was seamlessly integrated within the lab as a central control system to streamline the control and management of IoT devices [11].

Data collection involved strategically deploying network sniffers and monitoring points throughout the lab to capture both wired and wireless network traffic, thus creating a realistic and dynamic environment for experimentation. The dataset stands out in the IoT security research landscape due to several vital contributions: the inclusion of a dynamic mix of IoT devices, simulation of a realistic home environment within the IoTRL, focus on behavioural analysis of IoT devices in normal and adversarial scenarios, and incorporation of multi-modal data representation including network traffic patterns, device communication, and device-specific characteristics. These contributions enable a comprehensive evaluation of IoT security, surpassing the limitations of existing datasets with a single focus [11].

- *CIC IoT dataset 2022*: This study endeavours to produce an advanced dataset to profile, analyse behaviour, and test vulnerabilities of various IoT devices utilizing protocols such as IEEE 802.11, Zigbee-based, and Z-Wave. Regarding network configuration, our lab network was set up with a 64-bit Windows machine equipped with two network interface cards—one linked to the network gateway and the other to an unmanaged network switch. Wireshark, an open-source network protocol analyzer, concurrently monitors both interfaces, capturing and saving packet data. This setup allows IoT devices requiring an Ethernet connection to the switch [16].

Additionally, a smart automation hub, Vera Plus, is linked to the unmanaged

switch, establishing our wireless IoT environment to cater to devices compatible with Wi-Fi, Zigbee, Z-Wave, and Bluetooth. Data collection involved capturing network traffic of IoT devices passing through the gateway using Wireshark and dumpcap across six experiment types: Power, Idle, Interactions, Scenarios, Active, and Attacks. Each experiment type is delineated as follows: Power involved individually powering on devices in isolation and capturing network traffic; Idle captured network traffic during overnight periods of minimal lab activity; Interactions extracted all possible functionality from IoT devices, capturing corresponding network activity; Scenarios simulated network activity within a smart home environment with various device combinations; Active captured network communication throughout the day, allowing fellow researchers to interact with devices; Attacks entailed performing Flood and RTSP-Brute Force attacks on select devices and capturing their attack network traffic [16].

- *CIC IoMT dataset 2024*: The primary objective of this study is to propose a realistic benchmark dataset to facilitate the development and evaluation of security solutions for the Internet of Medical Things (IoMT). This involved executing 18 attacks on an IoMT testbed comprising 40 devices (25 real and 15 simulated), considering the diverse protocols prevalent in healthcare settings, such as Wi-Fi, MQTT, and Bluetooth. These attacks were categorized into five classes, DDoS, DoS, Recon, MQTT, and spoofing, to establish a foundational benchmark that complements existing contributions and aids researchers in exploring novel security mechanisms, including machine learning (ML). Data collection involved the utilization of a network tap to capture traffic between the switch and the Wi-Fi/MQTT-enabled IoMT devices, with real-time packet duplication facilitated by the device to construct the security and profiling dataset. Furthermore, a malicious PC and a smartphone were employed to capture malicious and benign data for Bluetooth Low Energy (BLE) enabled devices [17].

- *Edge-IIoTset*: It is a comprehensive and realistic cyber security dataset tailored for IoT and IIoT applications, intended for utilization in machine learning-based intrusion detection systems operating in centralized and federated learning modes. The dataset is structured across seven layers: Cloud Computing Layer, Network Functions Virtualization Layer, Blockchain Network Layer, Fog Computing Layer, Software-Defined Networking Layer, Edge Computing Layer, and IoT and IIoT Perception Layer. Each layer incorporates cutting-edge technologies meeting the essential requirements of IoT and IIoT applications, including the ThingsBoard IoT platform, OPNFV platform, Hyperledger Sawtooth, Digital twin, ONOS SDN controller, Mosquitto MQTT brokers, Modbus TCP/IP, among others[21].

Data generation involves diverse IoT devices spanning over ten types: low-cost digital sensors for temperature and humidity, ultrasonic sensors, water level detection sensors, pH Sensor Meters, soil moisture sensors, heart rate sensors, and flame sensors. Furthermore, the project identifies and analyzes fourteen attacks associated with IoT and IIoT connectivity protocols, categorized into five threat classes: DoS/DDoS attacks, Information gathering, Man-in-the-middle attacks, Injection attacks, and Malware attacks. Following the processing and analysis of the proposed cyber security dataset, primary exploratory data analysis is conducted, and the performance of machine learning approaches is evaluated in centralized and federated learning modes [21].

- *IoT Network Intrusion Dataset*: This dataset features various network attacks within an Internet of Things (IoT) environment, employing two representative smart home devices: the SKT NUGU (NU 100) and the EZVIZ Wi-Fi Camera (C2C Mini O Plus 1080P). All devices, alongside selected laptops or smartphones, were interconnected within a unified wireless network. The dataset encompasses 42 raw network packet files (pcap) captured at distinct intervals.

These packet files were obtained using the monitor mode of a wireless network adapter, with subsequent removal of wireless headers facilitated by Aircrack-ng. The dataset includes attacks simulated using tools such as Nmap, excluding those categorized under the Mirai Botnet, for which attack packets were generated on a laptop and subsequently altered to appear as originating from an IoT device [37].

- *MQTT-IoT-IDS 2020*: This study aims to develop a dataset pertaining to the Internet of Things (IoT) domain, specifically focusing on the MQTT communication protocol, to provide the research and industrial communities with an initial dataset for their applications. The dataset comprises IoT sensors operating on MQTT, each representing various aspects of a real network. Specifically, the MQTT broker is instantiated using Eclipse Mosquitto, and the network consists of 8 sensors. The scenario emulates a smart home environment where sensors collect data on temperature, light, humidity, CO-Gas, motion, smoke, door status, and fan operation at different time intervals, reflecting the diverse behaviour of each sensor. These 8 MQTT sensors are characterized by distinct features, detailed in a table format, with each sensor associated with a data profile and a topic linked to the MQTT broker [31].

Additionally, the sensors are conceptually divided into two rooms, resembling distribution within a smart house, while the MQTT broker is assigned the IP address 10.16.100.73 and port 1883 for clear text communication. The time intervals for data transmission can either be periodic or random, with periodic intervals reflecting cyclic data transmission, such as that of a temperature sensor, and random intervals representing sporadic data transmission, as in the case of a motion sensor triggered by user activity. This consideration enhances the dataset’s fidelity, simulating and implementing realistic behaviours characteristic of home automation systems [31].

- *TON IoT*: The TON-IoT datasets represent a new iteration of Industry 4.0/Internet of Things (IoT) and Industrial IoT (IIoT) datasets designed to assess the efficacy and authenticity of diverse cybersecurity applications leveraging Artificial Intelligence (AI), including Machine and Deep Learning algorithms. These datasets amalgamate heterogeneous data sources derived from Telemetry datasets of IoT and IIoT sensors, Operating systems datasets encompassing Windows 7 and 10, Ubuntu 14 and 18 TLS, and Network traffic datasets. They were compiled from a realistic and expansive network environment established at the Cyber Range and IoT Labs within the School of Engineering and Information Technology (SEIT) at UNSW Canberra @ the Australian Defence Force Academy (ADFA). A novel testbed network was devised specifically for the Industry 4.0 context, encompassing IoT and IIoT networks. It was implemented through multiple virtual machines and hosts operating on Windows, Linux, and Kali systems, facilitating the integration across IoT, Cloud, and Edge/Fog layers. The datasets encapsulate a range of attacking methodologies, including DoS, DDoS, and ransomware, targeting web applications, IoT gateways, and computer systems within the IoT/IIoT network. The datasets were compiled Through parallel processing techniques to capture normal and cyber-attack events from network traffic, Windows audit traces, Linux audit traces, and telemetry data originating from IoT services [52].
- *UQ IoT IDS 2021*: The UQ-IOT-IDS-2021 dataset, provided by the University of Queensland (UQ), Australia, primarily focuses on cybersecurity within the Internet of Things (IoT) domain. It encompasses benign traffic generated by various IoT devices, including smartphones, smart TVs, IP cameras, smart speakers, and Raspberry Pis, among others, alongside malicious traffic originating from 9 distinct cyberattack types, such as Host Discovery, Port Scanning, Service Detection, ARP Spoofing, Telnet Brute-force, SYN Flooding,

UDP Flooding, HTTP Flooding, and ACK Flooding. The raw data was captured utilizing Wireshark within a realistic network environment established at the School of Information Technology and Electrical Engineering, UQ. Subsequently, this raw data underwent processing by the feature extractor in Kitsune (Mirsky et al., 2018) to extract 100 temporal-statistical features, enhancing the dataset’s utility for cybersecurity research and analysis [30].

Table 5.2: List Of All Malicious Activities Present In All Datasets in Table 5.1 [60] [77]

Malicious Activity	Description
Benign Traffic	Normal network traffic with no malicious intent.
DDoS ACK Frag- mentation	Distributed Denial of Service attack using fragmented ACK packets to overwhelm the target.
DDoS ICMP Flood	DDoS attacks send many ICMP packets to overload the target.
DDoS ICMP Frag- mentation	Fragmented ICMP packets are used to consume the target’s resources in a DDoS attack.
DDoS PSHACK Flood	Flooding the target with TCP packets with PSH and ACK flags set to disrupt services.
DDoS RSTFIN Flood	Using TCP RST and FIN flags to flood and disrupt target systems.
DDoS SYN Flood	Overwhelming a target with TCP SYN packets to exhaust resources and deny service.
DDoS Synonymous IP Flood	Using multiple IP addresses to send DDoS traffic to the target.
DDoS TCP Flood	Flooding the target with TCP packets to cause a denial of service.

Malicious Activity	Description
DDoS UDP Flood	Sending large UDP packets to overwhelm the target's resources.
DDoS UDP Fragmentation	Fragmented UDP packets are used in a DDoS attack to disrupt services.
DNS Spoofing	Manipulating DNS responses to redirect traffic to malicious sites.
DoS HTTP Flood	Denial of Service attack using HTTP requests to overwhelm a web server.
DoS SYN Flood	Sending numerous TCP SYN packets to cause a denial of service on the target.
DoS TCP Flood	Flooding the target with TCP packets to cause a denial of service.
DoS UDP Flood	Overwhelming the target with UDP packets to cause a denial of service.
MITM ARP Spoofing	Man-in-the-Middle attack using ARP spoofing to intercept network traffic.
Mirai GRE-ETH Flood	Mirai botnet attack using GRE over Ethernet packets to flood the target.
Mirai GRE-IP Flood	Mirai botnet attack using GRE over IP packets to flood the target.
Mirai UDP Plain	Mirai botnet attack using plain UDP packets to overwhelm the target.
Dictionary Brute Force	Attempting to gain access by trying numerous passwords from a predefined list.

Malicious Activity	Description
DoS DNS Flood	Overwhelming a DNS server with requests to cause a denial of service.
Recon Host Discovery	Scanning the network to discover active hosts.
Recon OS Scan	Scanning to identify the operating systems of hosts on the network.
Recon Ping Sweep	Sending ICMP Echo requests to multiple hosts to determine their status.
Recon Port Scan	Scanning a host to discover open ports and services.
Recon Vulnerability Scan	Scanning systems to identify potential vulnerabilities.
Backdoor	Creating or exploiting a hidden access point into a system.
DDoS HTTP Flood	Overwhelming a web server with HTTP requests in a DDoS attack.
DDoS TCP SYN Flood	A specific type of SYN flood used in a DDoS attack to exhaust resources.
OS Fingerprinting	Identifying the operating system of a target by analyzing packet signatures.
Password Attack	Attempting to gain unauthorized access by guessing or cracking passwords.
Port Scanning	Identifying open ports and services on a networked device.

Malicious Activity	Description
Ransomware	Malicious software that encrypts files and demands ransom for decryption.
SQL Injection	Inserting malicious SQL queries into input fields to manipulate databases.
Uploading Attacks	Uploading malicious files to a server to exploit vulnerabilities.
Vulnerability Scanner	Automated tool to scan systems for known vulnerabilities
XSS	Cross-site scripting attack to inject malicious scripts into web pages.
DoS ICMP Flood	Overloading a target with ICMP Echo requests to cause a denial of service.
MQTT DDoS Connect Flood	Using the MQTT protocol to send a large number of connection requests in a DDoS attack.
MQTT DDoS Publish Flood	Flooding the target with MQTT publish messages to cause a denial of service.
MQTT DoS Connect Flood	Using MQTT connect messages to cause a denial of service.
MQTT DoS Publish Flood	Using MQTT publish messages to cause a denial of service.
MQTT Malformed	Sending malformed MQTT packets to disrupt the target.
Mirai ACK Flood	Mirai botnet attack using ACK packets to overwhelm the target.

Malicious Activity	Description
Mirai HTTP Flood	Using the Mirai botnet to flood a web server with HTTP requests.
Mirai Host Brute Force	Mirai botnet are attempting to brute force login credentials on targets.
Mirai UDP Flood	Mirai botnet sends a large volume of UDP packets to overwhelm the target.
Scan Host Port	Scanning hosts to identify open ports and services.
Scan Port OS	Scanning ports to identify the target's operating system.
MQTT Brute Force	Attempting to gain unauthorized access by brute forcing MQTT credentials.
Scan Aggressive	Comprehensive scanning to gather detailed information about the target.
Scan UDP Attack	Scanning using UDP packets to identify open ports and services.
Sparta SSH Brute Force	Using the Sparta tool to brute force SSH login credentials.
DDoS	Distributed Denial of Service attack using multiple sources to overwhelm the target.
DoS	Denial of Service attack aimed at making a service unavailable.
Injection	Attacks that involve injecting malicious input into an application.
MITM	Man-in-the-Middle attack where the attacker intercepts communication between two parties.

Malicious Activity	Description
ACK Flood	Flooding the target with TCP ACK packets to disrupt services.
HTTP Flood	Overwhelming a web server with HTTP requests to cause a denial of service.
Recon Service Detection	Scanning to detect active services on a network.
SYN Flood	Flooding a target with SYN packets to exhaust resources and deny service.
Telnet Brute Force	Brute forcing Telnet login credentials to gain unauthorized access.
UDP Flood	Overwhelming a target with UDP packets to cause a denial of service.

5.4 Features

This section describes all the time-related statistical features extracted using CIC Flowmeter from benign and malicious PCAP files for TCP network flow data in Table 5.3.

Table 5.3: CICFlowmeter Extracted Features List

Feature Name	Description
Flow Id	Flow Identifier
Src IP	Source IP
Src Port	Source Port
Dst IP	Destination IP
Dst Port	Destination Port

Table 5.3 – continued from previous page

Feature Name	Description
Flow duration	Duration of the flow in microseconds
Total Fwd Packet	Total packets in the forward direction
Total Bwd packets	Total packets in the backward direction
Total Length of Fwd Packet	Total size of packet in forward direction
Total Length of Bwd Packet	Total size of packet in backward direction
Fwd Packet Length Min	Minimum size of the packet in forward direction
Fwd Packet Length Max	Maximum size of the packet in forward direction
Fwd Packet Length Mean	Mean size of packet in forward direction
Fwd Packet Length Std	Standard deviation size of packet in forward direction
Bwd Packet Length Min	Minimum size of packet in backward direction
Bwd Packet Length Max	Maximum size of the packet in backward direction
Bwd Packet Length Mean	Mean size of the packet in backward direction
Bwd Packet Length Std	Standard deviation size of packet in backward direction
Flow Bytes/s	Number of flow bytes per second

Table 5.3 – continued from previous page

Feature Name	Description
Flow Packets/s	Number of flow packets per second
Flow IAT Mean	Mean time between two packets sent in the flow
Flow IAT Std	Standard deviation time between two packets sent in the flow
Flow IAT Max	Maximum time between two packets sent in the flow
Flow IAT Min	Minimum time between two packets sent in the flow
Fwd IAT Min	Minimum time between two packets sent in the forward direction
Fwd IAT Max	Maximum time between two packets sent in the forward direction
Fwd IAT Mean	Mean time between two packets sent in the forward direction
Fwd IAT Std	Standard deviation time between two packets sent in the forward direction
Fwd IAT Total	Total time between two packets sent in the forward direction
Bwd IAT Min	Minimum time between two packets sent in the backward direction
Bwd IAT Max	Maximum time between two packets sent in the backward direction
Bwd IAT Mean	Mean time between two packets sent in the backward direction

Table 5.3 – continued from previous page

Feature Name	Description
Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
Bwd IAT Total	Total time between two packets sent in the backward direction
Fwd PSH flags	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
Bwd PSH Flags	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
Fwd URG Flags	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
Bwd URG Flags	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
Fwd Header Length	Total bytes used for headers in the forward direction
Bwd Header Length	Total bytes used for headers in the backward direction
FWD Packets/s	Number of forward packets per second
Bwd Packets/s	Number of backward packets per second
Packet Length Min	Minimum length of a packet
Packet Length Max	Maximum length of a packet
Packet Length Mean	Mean length of a packet
Packet Length Std	Standard deviation length of a packet
Packet Length Variance	Variance length of a packet

Table 5.3 – continued from previous page

Feature Name	Description
FIN Flag Count	Number of packets with FIN
SYN Flag Count	Number of packets with SYN
RST Flag Count	Number of packets with RST
PSH Flag Count	Number of packets with PUSH
ACK Flag Count	Number of packets with ACK
URG Flag Count	Number of packets with URG
CWR Flag Count	Number of packets with CWR
ECE Flag Count	Number of packets with ECE
Down/Up Ratio	Download and upload ratio
Average Packet Size	Average size of packet
Fwd Segment Size Avg	Average size observed in the forward direction
Bwd Segment Size Avg	Average size observed in the backward direction
Fwd Bytes/Bulk Avg	Average number of bytes bulk rate in the forward direction
Fwd Packet/Bulk Avg	Average number of packets bulk rate in the forward direction
Fwd Bulk Rate Avg	Average number of bulk rate in the forward direction
Bwd Bytes/Bulk Avg	Average number of bytes bulk rate in the backward direction
Bwd Packet/Bulk Avg	Average number of packets bulk rate in the backward direction

Table 5.3 – continued from previous page

Feature Name	Description
Bwd Bulk Rate Avg	Average number of bulk rate in the backward direction
Subflow Fwd Packets	The average number of packets in a sub-flow in the forward direction
Subflow Fwd Bytes	The average number of bytes in a sub-flow in the forward direction
Subflow Bwd Packets	The average number of packets in a sub-flow in the backward direction
Subflow Bwd Bytes	The average number of bytes in a sub-flow in the backward direction
Fwd Init Win bytes	The total number of bytes sent in the initial window in the forward direction
Bwd Init Win bytes	The total number of bytes sent in the initial window in the backward direction
Fwd Act Data Pkts	Count of packets with at least 1 byte of TCP data payload in the forward direction
Fwd Seg Size Min	Minimum segment size observed in the forward direction
Active Min	Minimum time a flow was active before becoming idle
Active Mean	Mean time a flow was active before becoming idle
Active Max	Maximum time a flow was active before becoming idle

Table 5.3 – continued from previous page

Feature Name	Description
Active Std	Standard deviation time a flow was active before becoming idle
Idle Min	Minimum time a flow was idle before becoming active
Idle Mean	Mean time a flow was idle before becoming active
Idle Max	Maximum time a flow was idle before becoming active
Idle Std	Standard deviation time a flow was idle before becoming active

5.5 Finalizing the Proposed Model

In this section, the proposed model is elaborated upon in detail. After extracting features using the CICFlowmeter, the newly created dataset was filtered to include only TCP flows and KNN Imputation was employed to address missing values in "Flow Bytes/s" and "Flow Packets/s" columns. Subsequently, a 1D CNN + LSTM model with a self-attention mechanism was constructed, and hyperparameter tuning was performed using the Grid Search technique. This process identified the top 5 hyperparameters indicated in Table 4.6.

We evaluated five models across eight augmented test datasets. From these, the top best-performing model was selected for SHAP feature importance calculations. These scores were then utilized to identify reduced model retraining and compression features.

After training the top model with this reduced feature set, a comparison was conducted against the initial model. As depicted in Figure 5.8, the results indicated only

a minor performance decrement between the initial model and those with the reduced feature set.

5.5.1 Performance Metrics

Confusion Matrix

A confusion matrix is a table used to describe the performance of a classification model on a set of data for which the true values are known. It summarizes the number of correct and incorrect predictions made by the model, categorized by each class. This matrix includes the following elements:

- **True Positives (TP):** Cases in which the model correctly predicts the positive class.
- **True Negatives (TN):** Cases in which the model correctly predicts the negative class.
- **False Positives (FP):** Cases in which the model incorrectly predicts the positive class.
- **False Negatives (FN):** Cases in which the model incorrectly predicts the negative class.

The confusion matrix is arranged as follows:

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Accuracy

Accuracy measures the proportion of true results (both true positives and true negatives) among the total number of cases examined. It is calculated as:

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

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

Precision

Precision, or positive predictive value, measures the accuracy of positive predictions. It is defined as the ratio of true positives to the sum of true and false positives:

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

Recall

Recall, also known as sensitivity or true positive rate, measures the ability of a model to find all the relevant cases within a dataset. It is defined as:

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

F1-Score

The F1-score is the harmonic mean of precision and recall, providing a balance between the two when the class distribution is uneven. The F1-score is given by:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

These metrics collectively provide a comprehensive view of model performance, par-

ticularly in classification problems.

Interpreting the Confusion Matrix in Model Context

- **Accuracy:** Measures overall correctness of the model.
- **Precision:** Indicates the accuracy of positive predictions.
- **Recall (Sensitivity):** Measures the model’s ability to detect positive instances.
- **F1 Score:** Harmonic mean of Precision and Recall, particularly useful if the class distribution is imbalanced.

These metrics from the confusion matrix provide a comprehensive view of the model performance across different aspects.

5.6 Experimental Results

This section presents our experiment findings regarding identifying malicious activities within network TCP flow data. Due to the predominance of malicious activities over benign ones, a data balancing procedure was necessitated prior to model training. Specifically, we retained all rows corresponding to benign activities and randomly sampled consecutive rows representing malicious activities from each respective group within the dataset. This approach ensured parity in the total count of benign and malicious activities. Consequently, the size of the resulting training set was 356.3 MB.

Drawing upon the insights the Table 4.6 provided, we opted for the most effective hyperparameter configuration to train our model architecture employing 5-fold cross-validation. Following this, we conducted comprehensive tests and comparisons be-

tween our model and baseline models across all datasets mentioned in Table 5.1. The outcomes of these analyses are displayed in Table 5.4 and Figure 5.1.

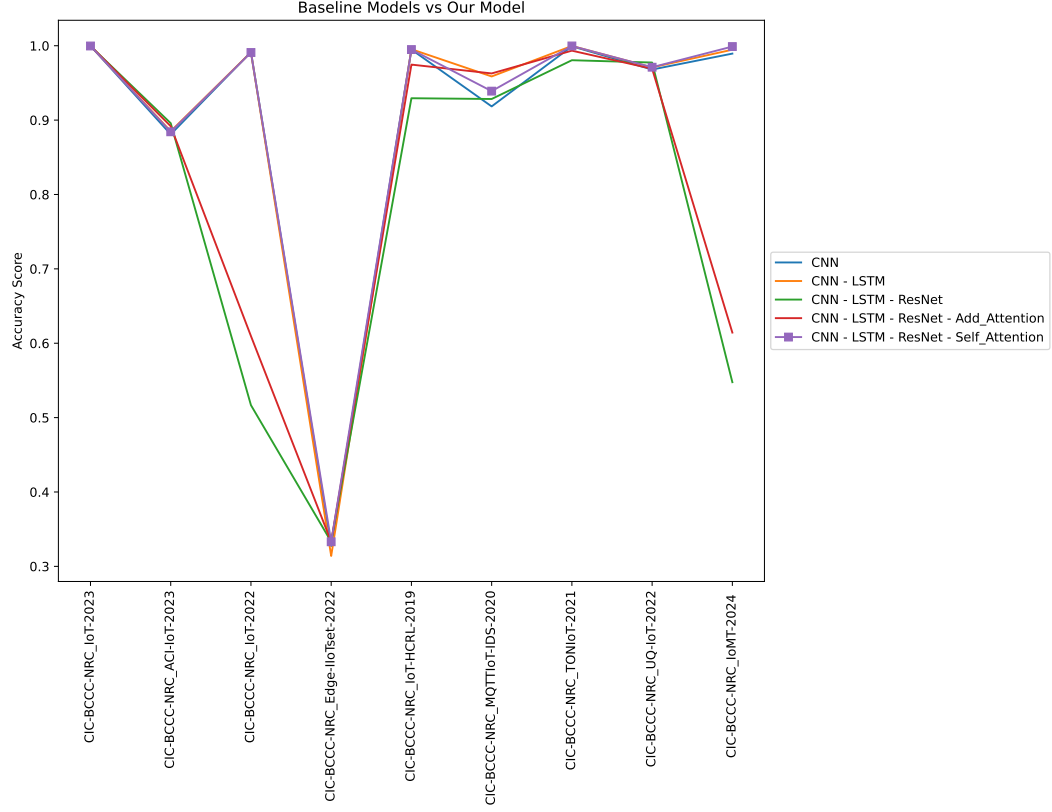


Figure 5.1: Comparison between Baseline Models and Our Model

Table 5.4: Baseline Models Evaluation Results

Dataset	CNN	CNN-LSTM	CNN-LSTM-ResNet	CNN-LSTM-ResNet-Additive Attention	CNN-LSTM-ResNet-Self Attention
CIC-BCCC-NRC_IoT-2023	0.9999	0.9998	0.9995	0.9998	0.9996

Dataset	CNN	CNN- LSTM	CNN- LSTM- ResNet	CNN- LSTM- ResNet- Additive Atten- tion	CNN- LSTM- ResNet- Self Atten- tion
CIC-BCCC- NRC_ACI-IoT- 2023	0.8809	0.8850	0.8958	0.8918	0.8842
CIC-BCCC- NRC_IoT-2022	0.9915	0.9912	0.5167	0.6096	0.9908
CIC-BCCC- NRC_Edge- IIoTset-2022	0.3273	0.3140	0.3331	0.3331	0.3330
CIC-BCCC- NRC_IoT- HCRL-2019	0.9949	0.9953	0.9294	0.9746	0.9948
CIC-BCCC- NRC_MQTTIoT- IDS-2020	0.9184	0.9587	0.9284	0.9627	0.9639
CIC-BCCC- NRC_TONIOT- 2021	0.9992	0.9997	0.9805	0.9934	0.9997
CIC-BCCC- NRC_UQ-IoT- 2022	0.9680	0.9709	0.9774	0.9690	0.9711

Dataset	CNN	CNN-LSTM	CNN-LSTM-ResNet	CNN-LSTM-ResNet-Additive Attention	CNN-LSTM-ResNet-Self Attention
CIC-BCCC-NRC-IoMT-2024	0.9863	0.9899	0.5045	0.5873	0.9988

The Figure 5.2 and Figure 5.3 describe our model’s training history and confusion matrix.

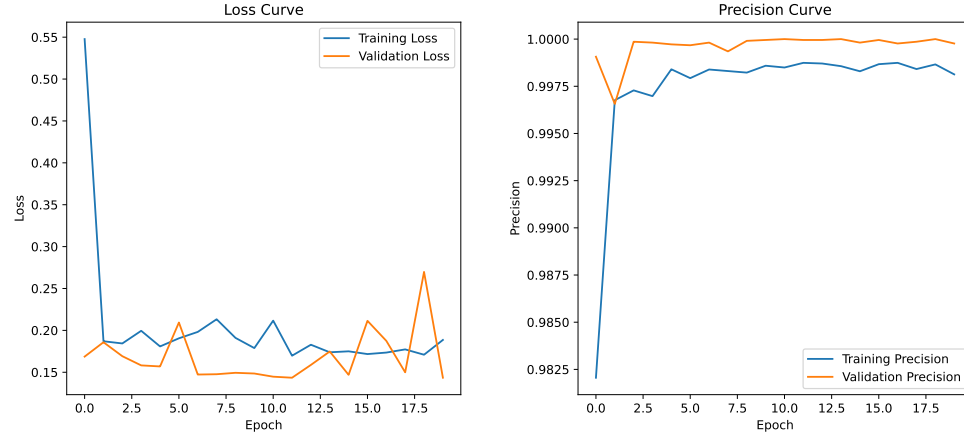


Figure 5.2: Our Model’s Training History Results

Upon evaluating our model using the CIC-BCCC-NRC-IoT-2023 test data and the dataset referenced in Table 5.1, we obtained the results summarized in Table 5.5 and subsequent confusion matrix are given in Table 5.6.

After completing the evaluation process, SHAP analysis is employed to compute feature importance, facilitating the identification of key features that contribute to the

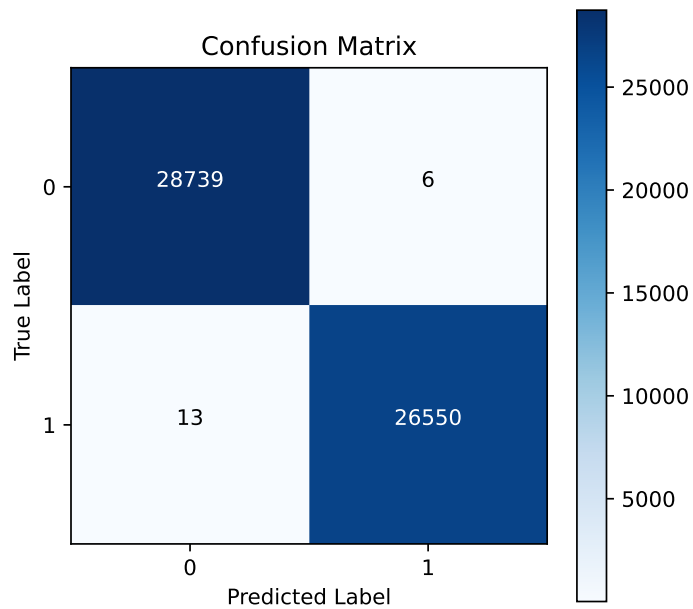


Figure 5.3: Our Model's Confusion Matrix Results

model's predictive outcomes. Subsequently, Figure 5.4 and Figure 5.5 illustrate the summary plot and feature importance scores derived from the analysis of a randomly selected subset of 100 test data samples.

Table 5.5: Our Model Accuracy List

Dataset	Accuracy
CIC-BCCC-NRC-IoT-2023	0.9996
CIC-BCCC-NRC-ACI-IoT-2023	0.8842
CIC-BCCC-NRC-IoT-2022	0.9908
CIC-BCCC-NRC-Edge-IIoTset-2022	0.3330
CIC-BCCC-NRC-IoT-HCRL-2019	0.9948
CIC-BCCC-NRC-MQTTIoT-IDS-2020	0.9633
CIC-BCCC-NRC-TONIoT-2021	0.9997
CIC-BCCC-NRC-UQ-IoT-2022	0.9711
CIC-BCCC-NRC-IoMT-2024	0.9988

Table 5.6: Confusion Metrics For Initial Model Across All datasets

Dataset	Accuracy	Precision	Recall	F1 Score
CIC-BCCC-NRC-IoT-2023	0.9996	0.9998	0.9995	0.9996
CIC-BCCC-NRC-ACI-IoT-2023	0.8842	0.8848	0.9984	0.9381
CIC-BCCC-NRC-IoT-2022	0.9908	0.9916	0.9993	0.9954
CIC-BCCC-NRC-Edge-IIoTset-2022	0.3330	0.3331	1	0.4997
CIC-BCCC-NRC-IoT-HCRL-2019	0.9948	0.9968	0.9981	0.9974
CIC-BCCC-NRC-MQTTIoT-IDS-2020	0.9633	0.9630	0.9741	0.9685
CIC-BCCC-NRC-TONIoT-2021	0.9997	1	0.9997	0.9999
CIC-BCCC-NRC-UQ-IoT-2022	0.9711	0.9710	1	0.9853
CIC-BCCC-NRC-IoMT-2024	0.9988	0.9989	0.9999	0.9994

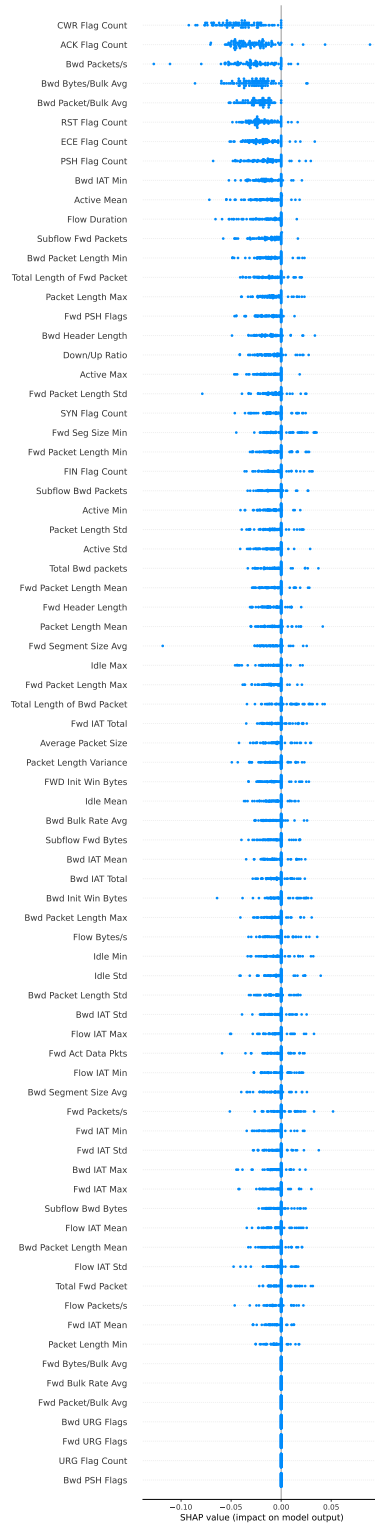


Figure 5.4: SHAP Summary Plot for Our Model (Trained on CIC-BCCC-NRC-IoT-2023 dataset)

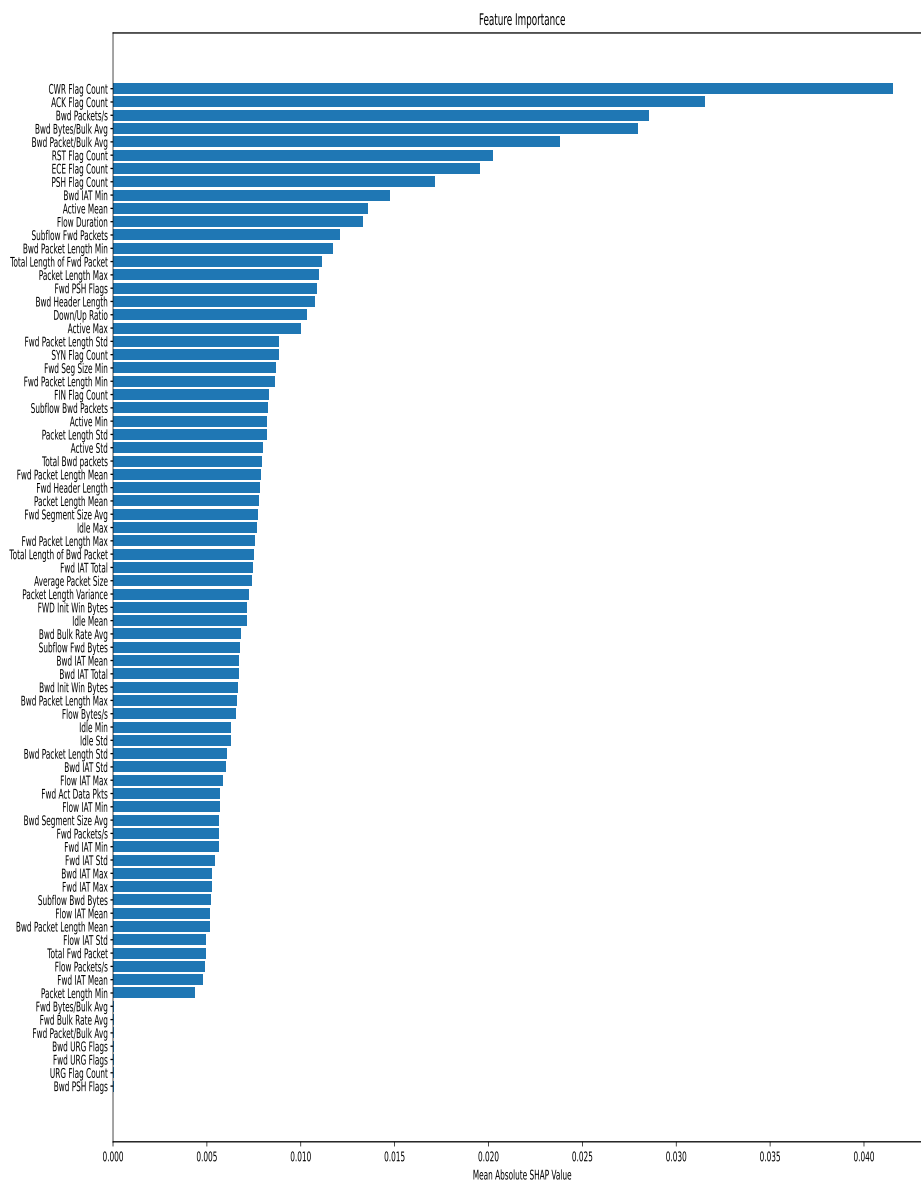


Figure 5.5: Feature Importance Graph for
CIC-BCCC-NRC-IoT-2023 dataset

After identifying the top important features, we begin retraining the model using a reduced set of features from scratch, aiming to determine the optimal subset of features capable of achieving comparable performance to the original model. The gaps or differences between the bars in the Figure 5.5 guide the choice of the reduced feature set. Subsequently, we compile a list of feature subsets and retrain and evaluate the model using the CIC-BCCC-NRC-IoT-2023 dataset and other datasets specified in Table 5.1. Through this evaluation process, we ascertain that a minimum of 7 features is necessary to achieve performance akin to the initial model's. Table 5.7 provides a detailed description of the top 7 features identified, which exhibit similar performance to the original model.

Table 5.7: Top 7 Best Features For Retraining

Feature Name	Description
CWR Flag Count	This represents the count of packets with the Congestion Window Reduced (CWR) flag set. The CWR flag is set by the sending host to indicate that it received a TCP segment with the ECE flag set and had responded in a congestion control mechanism. Anomalies in the CWR flag count could indicate potential congestion or network attacks, such as congestion collapse or denial-of-service (DoS) attacks targeting the TCP congestion control mechanism.

ACK Flag Count	This denotes the count of packets with the Acknowledgment (ACK) flag set. The ACK flag is used to acknowledge receipt of data packets. A sudden surge or drop in the ACK flag count could signify abnormal network behaviour, such as TCP connection hijacking attempts or ACK flooding attacks.
Bwd Packets/s	This indicates the rate of backward (incoming) packets per second. Unusually high or low rates of backward packets might indicate malicious activities like network scanning or packet injection attacks.
Bwd Avg Bytes/Bulk	This represents the average number of bytes transmitted in bulk in the backward direction. Significant deviations from the average bulk byte rate could suggest data exfiltration attempts or large-scale data transfer activities indicative of a security breach.
Bwd Avg Packet/Bulk	This denotes the average number of packets transmitted in bulk in the backward direction. Deviations from the average bulk packet rate could indicate abnormal traffic patterns, such as bulk data transfers associated with malware propagation or botnet activity.

RST Flag Count	This indicates the count of packets with the Reset (RST) flag set. The RST flag is used to reset a TCP connection. Anomalous spikes in the RST flag count may signal attempts to disrupt network communications, such as TCP reset attacks or connection teardown attempts by malware.
ECE Flag Count	This represents the count of packets with the Explicit Congestion Notification (ECE) flag set. The ECE flag is used to indicate congestion on the network path. Abnormalities in the ECE flag count could suggest network congestion or targeted congestion-based attacks, such as congestion-aware DoS attacks or congestion control evasion techniques employed by attackers.

Table 5.8: Reduced Set’s Features Accuracy List

Dataset	4 Fea- tures	5 Fea- tures	7 Fea- tures	10 Fea- tures	11 Fea- tures
CIC-BCCC-NRC_IoT-2023	0.9847	0.9963	0.9973	0.9992	0.9995
CIC-BCCC-NRC_ACI-IoT-2023	0.8797	0.8804	0.8803	0.8801	0.8812
CIC-BCCC-NRC_IoT-2022	0.2603	0.2635	0.9845	0.9914	0.9914

Dataset	4 Fea- tures	5 Fea- tures	7 Fea- tures	10 Fea- tures	11 Fea- tures
CIC-BCCC-NRC_Edge-IIoTset-2022	0.3330	0.3330	0.3330	0.3330	0.3330
CIC-BCCC-NRC_IoT-HCRL-2019	0.9807	0.9027	0.9691	0.8740	0.9715
CIC-BCCC-NRC_MQTTIoT-IDS-2020	0.9639	0.9638	0.9639	0.9642	0.9639
CIC-BCCC-NRC_TONIIoT-2021	0.9988	0.9997	0.9999	0.9998	0.9999
CIC-BCCC-NRC_UQ-IoT-2022	0.9680	0.9679	0.9680	0.9642	0.9690
CIC-BCCC-NRC_IoMT-2024	0.2711	0.2747	0.9987	0.9925	0.9905

The following section outlines the training history (Figure 5.6) and confusion matrix (Figure 5.7) for the CIC-BCCC-NRC-IoT-2023 dataset, along with confusion matrices for the datasets referenced in Table 5.1 are presented in Table 5.9.

Finally, Figure 5.8 shows the performance comparison between the initial model and the reduced feature set model. Since all the datasets are highly imbalanced, with a higher number of attack instances than benign ones, and considering that we utilized

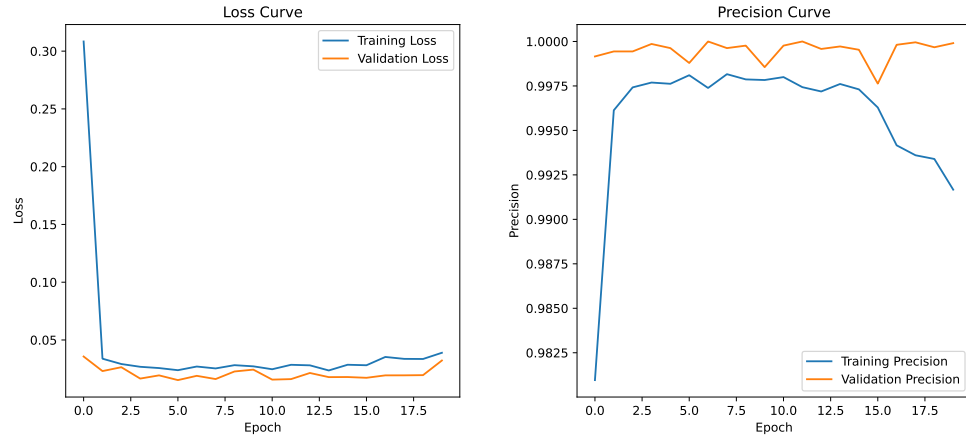


Figure 5.6: Reduced Set Feature's Training History Results

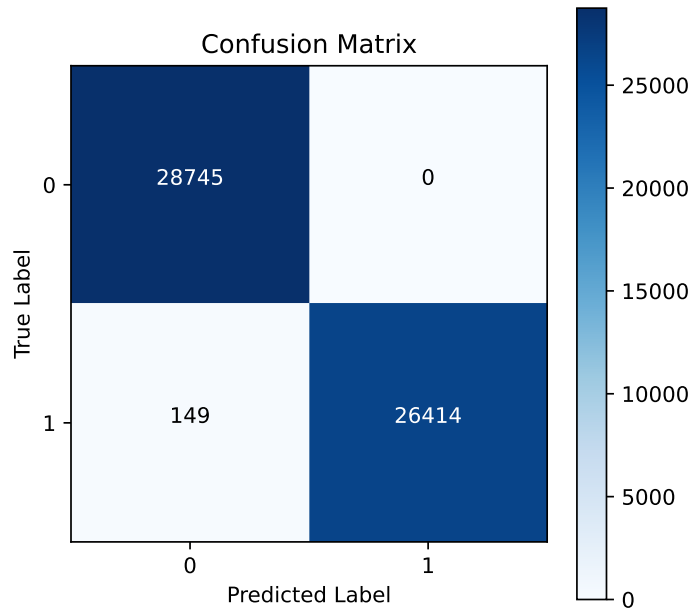


Figure 5.7: Reduced Set Feature's Confusion Matrix Results

these datasets as test datasets without balancing them, it is likely that this imbalance accounts for the high recall and precision observed in our model, as demonstrated in the confusion matrices Table 5.6 and Table 5.9. However, since we trained the model

Dataset	Accuracy	Precision	Recall	F1 Score
CIC-BCCC-NRC-IoT-2023	0.9996	1	0.9944	0.9972
CIC-BCCC-NRC-ACI-IoT-2023	0.8842	0.881	0.9989	0.9363
CIC-BCCC-NRC-IoT-2022	0.9908	0.9914	0.993	0.9922
CIC-BCCC-NRC-Edge-IoTset-2022	0.3330	0.333	1	0.4997
CIC-BCCC-NRC-IoT-HCRL-2019	0.9948	0.9956	0.9714	0.9834
CIC-BCCC-NRC-MQTTIoT-IDS-2020	0.9633	0.964	1	0.9817
CIC-BCCC-NRC-TONIoT-2021	0.9997	1	0.9999	0.9999
CIC-BCCC-NRC-UQ-IoT-2022	0.9711	0.9681	0.9999	0.9837
CIC-BCCC-NRC-IoMT-2024	0.9988	0.9989	0.9998	0.9993

Table 5.9: Confusion Metrics For 7 Feature Reduced Model
Across All datasets

on the balanced CIC-BCCC-NRC-IoT-2023 dataset, we are confident in its ability to accurately identify malicious instances. We anticipate that if the test datasets included a larger proportion of benign data, the performance difference would be negligible compared to the current results.

5.7 Concluding Remarks

This chapter assesses our proposed method’s effectiveness and efficiency, referencing the datasets specified in Table 5.1. Initially, the chapter provides a comprehensive outline of the experimental setup and a detailed description of the datasets utilized in the experiments. Further, it elaborates on the features and the feature extraction process, as introduced in Chapter 4. Additionally, the components constituting the proposed model are detailed in Chapter 4, followed by an exposition of the model

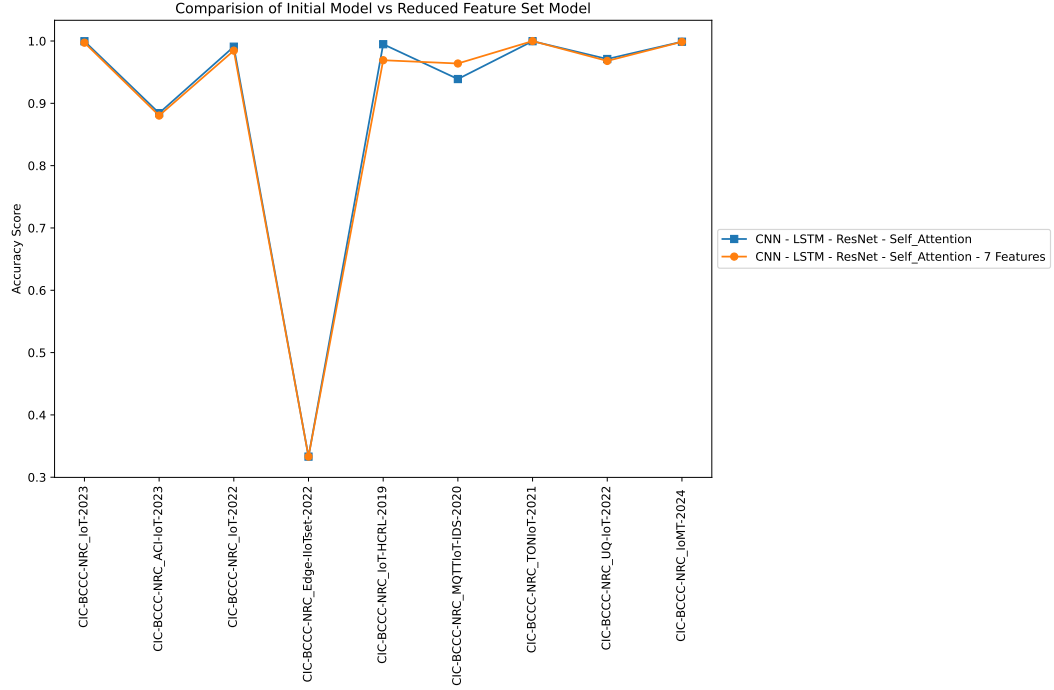


Figure 5.8: Performance Comparison between Initial Model vs Reduced Feature Set Model

creation and experimental workflow steps in Section 5.5.

Subsequently, a comparison of the performance metrics between the initial models and those with a reduced feature set is discussed in Section 5.6. The results of the experiments indicate that the reduced feature set models demonstrate comparable efficiency and accuracy to the initial models in discerning between benign and malicious behaviours in TCP network flow data.

Chapter 6

Conclusion & Future Works

6.1 Conclusions

In this thesis, a novel method integrating Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTMs), and self-attention mechanisms was developed to enhance the detection of Internet of Things (IoT) attacks within TCP network flows [83]. This approach capitalizes on the robust feature extraction capabilities of CNNs, the sequential data processing strengths of LSTMs, and the focused contextual insights provided by self-attention mechanisms, thereby establishing a sophisticated analytical framework.

Significant emphasis has been placed on the efficiency of the data preprocessing stage throughout this research, employing the CICFlowmeter tool for precise feature extraction from network traffic data and implementing advanced machine-learning techniques for feature selection and model optimization. Using SHAP values to elucidate feature importance highlights our commitment to transparency and interpretability in machine learning models, which is essential in cybersecurity.

Furthermore, this thesis tested the initial and reduced models on a newly created IoT tabular dataset, including eight externally available IoT datasets. The comparative

results confirm that the proposed method performs better in detecting IoT attacks across various datasets, providing empirical evidence of the model’s effectiveness in distinguishing malicious and benign samples within an IoT environment.

6.2 Future Works

1. *Model Architecture Optimization*: Continuously refining the architecture of the combined CNN-LSTM model with attention can lead to better performance. This involves experimenting with different network depths, layer configurations, activation functions, and other architectural components to improve model expressiveness and learning capacity.
2. *Attention Mechanism Enhancement*: Further research into attention mechanisms tailored specifically for network flow data can lead to more effective models. This includes exploring different attention mechanisms such as multiplicative attention, multi-head attention, or hierarchical attention to capture different aspects of temporal dependencies and relationships within the data.
3. *Integration of External Context*: Incorporating additional contextual information, such as network topology, device metadata, or domain-specific knowledge, into the model architecture can improve its ability to interpret network flow data accurately. This can enhance the model’s understanding of the underlying network dynamics and improve its predictive performance.
4. *Data Preprocessing and Feature Engineering*: Develop more sophisticated preprocessing techniques and feature engineering strategies specific to network flow data can help extract more meaningful representations from the raw data. This may involve incorporating domain knowledge to extract relevant features, handling missing or noisy data, or applying dimensionality reduction techniques to

reduce computational complexity.

5. *Model Improvement using Federated Learning Approach*: Enhancing the model architecture to operate within a federated framework facilitates the utilization of locally updated models, enabling the training of attack detection algorithms on IoT devices with similar characteristics. This approach yields more refined and precise models tailored to specific categories of IoT devices, consequently enhancing overall model performance and efficacy.

Bibliography

- [1] *What is siem*, Oct 2023, (Accessed on 06/04/2023).
- [2] Hezam Akram Abdul-Ghani, Dimitri Konstantas, and Mohammed Mahyoub, *A comprehensive iot attacks survey based on a building-blocked reference model*, International Journal of Advanced Computer Science and Applications **9** (2018), no. 3.
- [3] Ahmed Abulkhair, *Data imputation demystified time series data*, Jun 2023, (Accessed on 06/04/2023).
- [4] Robert Agar, *Iot architecture guide. major and additional layers of iot system*, Nov 2022, (Accessed on 06/04/2023).
- [5] Akamai, *What is an icmp flood ddos attack?*, (Accessed on 15/04/2024).
- [6] Richmond Alake, *Loss functions in machine learning explained*, Nov 2023, (Accessed on 06/04/2023).
- [7] Anastasios Arampatzis, *Top 10 vulnerabilities that make iot devices insecure*, Oct 2022, (Accessed on 06/04/2023).
- [8] Internet Assigned Numbers Authority, *Protocol numbers*, (Accessed on 06/04/2023).
- [9] Abid Ali Awan, *An introduction to shap values and machine learning interpretability*, Jun 2023, (Accessed on 06/04/2023).

- [10] Pragati Baheti, *Activation functions in neural networks 12 types & use cases*, May 2021, (Accessed on 06/04/2023).
- [11] Nathaniel Bastian, David Bierbrauer, Morgan McKenzie, and Emily Nack, *Aci iot network traffic dataset 2023*, 2023.
- [12] HJ Felcia Bel and S Sabeen, *A survey on iot security: attacks, challenges and countermeasures*, *Webology* **19** (2022), no. 1, 3741–3763.
- [13] Ayan Chatterjee and Bestoun S. Ahmed, *Iot anomaly detection methods and applications: A survey*, *Internet of Things* **19** (2022), 100568.
- [14] Cloudflare, *Syn flood attack*, (Accessed on 15/04/2024).
- [15] Laizhong Cui, Shu Yang, Fei Chen, Zhong Ming, Nan Lu, and Jing Qin, *A survey on application of machine learning for internet of things*, *International Journal of Machine Learning and Cybernetics* **9** (2018), no. 8, 1399–1417.
- [16] Sajjad Dadkhah, Hassan Mahdikhani, Priscilla Kyei Danso, Alireza Zohourian, Kevin Anh Truong, and Ali A. Ghorbani, *Towards the development of a realistic multidimensional iot profiling dataset*, 2022 19th Annual International Conference on Privacy, Security Trust (PST), 2022, pp. 1–11.
- [17] Sajjad Dadkhah, Euclides Carlos Pinto Neto, Raphael Ferreira, Reginald Chukwuka Molokwu, Somayeh Sadeghi, and Ali Ghorbani, *Ciciomt2024: Attack vectors in healthcare devices-a multi-protocol dataset for assessing iomt device security*, Preprints (2024).
- [18] Paulo Freitas de Araujo-Filho, Mohamed Naili, Georges Kaddoum, Emmanuel Thepie Fapi, and Zhongwen Zhu, *Unsupervised gan-based intrusion detection system using temporal convolutional networks and self-attention*, *IEEE Transactions on Network and Service Management* **20** (2023), no. 4, 4951–4963.

- [19] Anamika Dhillon and Gyanendra K. Verma, *Convolutional neural network: a review of models, methodologies and applications to object detection*, Progress in Artificial Intelligence **9** (2020), no. 2, 85–112.
- [20] Sanket Doshi, *Various optimization algorithms for training neural network*, Jan 2019, (Accessed on 06/04/2023).
- [21] Mohamed Amine Ferrag, Othmane Friha, Djallel Hamouda, Leandros Maglaras, and Helge Janicke, *Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications: Centralized and federated learning*, 2022.
- [22] Ghofrane Fersi, *Fog computing and internet of things in one building block: a survey and an overview of interacting technologies*, Cluster Computing **24** (2021), no. 4, 2757–2787.
- [23] Christian Garbin, Xingquan Zhu, and Oge Marques, *Dropout vs. batch normalization: an empirical study of their impact to deep learning*, Multimedia Tools and Applications **79** (2020), no. 19, 12777–12815.
- [24] DDoS Guard, *Ack and push ack flood*, (Accessed on 15/04/2024).
- [25] DDoS Guard, *Fragmented udp flood*, (Accessed on 15/04/2024).
- [26] DDoS Guard, *Rst and fin flood*, (Accessed on 15/04/2024).
- [27] DDoS Guard, *Synonymous ip attack*, (Accessed on 15/04/2024).
- [28] Ayush Gupta, *A comprehensive guide on optimizers in deep learning*, Jan 2024, (Accessed on 06/04/2023).
- [29] Somayye Hajiheidari, Karzan Wakil, Maryam Badri, and Nima Jafari Navimipour, *Intrusion detection systems in the internet of things: A comprehensive investigation*, Computer Networks **160** (2019), 165–191.

- [30] Kim Dan Zhang Zhien Ge Mengmeng Lam Ulysses He, Ke and Jiaqi Yu, *Uq iot ids dataset 2021*, 2021.
- [31] Hanan Hindy, Christos Tachtatzis, Robert Atkinson, Ethan Bayne, and Xavier Bellekens, *Mqtt-iot-ids2020: Mqtt internet of things intrusion detection dataset*, 2020.
- [32] Imperva, *Arp spoofing*, (Accessed on 15/04/2024).
- [33] Imperva, *Http flood*, (Accessed on 15/04/2024).
- [34] Imperva, *Tcp syn flood*, (Accessed on 15/04/2024).
- [35] Imperva, *Udp flood*, (Accessed on 15/04/2024).
- [36] Abhishek Jain, *Pooling and their types in cnn*, Feb 2024, (Accessed on 06/04/2023).
- [37] Hyunjae Kang, Dong Hyun Ahn, Gyung Min Lee, Jeong Do Yoo, Kyung Ho Park, and Huy Kang Kim, *Iot network intrusion dataset*, 2019.
- [38] Qing Kang, Elton J. Chen, Zhong-Chao Li, Han-Bin Luo, and Yong Liu, *Attention-based lstm predictive model for the attitude and position of shield machine in tunneling*, *Underground Space* **13** (2023), 335–350.
- [39] Minhaj Ahmad Khan and Khaled Salah, *Iot security: Review, blockchain solutions, and open challenges*, *Future Generation Computer Systems* **82** (2018), 395–411.
- [40] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J. Inman, *1d convolutional neural networks and applications: A survey*, *Mechanical Systems and Signal Processing* **151** (2021), 107398.

- [41] Djamel Eddine Kouicem, Abdelmadjid Bouabdallah, and Hicham Lakhlef, *Internet of things security: A top-down survey*, Computer Networks **141** (2018), 199–221.
- [42] Ritika Raj Krishna, Aanchal Priyadarshini, Amitkumar V. Jha, Bhargav Apasani, Avireni Srinivasulu, and Nicu Bizon, *State-of-the-art review on iot threats and attacks: Taxonomy, challenges and solutions*, Sustainability **13** (2021), no. 16.
- [43] Arash Habibi Lashkari, *Cicflowmeter (formerly iscxflowmeter)*, Jan 2022, (Accessed on 06/04/2023).
- [44] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou, *A survey of convolutional neural networks: Analysis, applications, and prospects*, IEEE Transactions on Neural Networks and Learning Systems **33** (2022), no. 12, 6999–7019.
- [45] Xingwei Liang and Yoohwan Kim, *A survey on security attacks and solutions in the iot network*, 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), 2021, pp. 0853–0859.
- [46] Scott M. Lundberg and Su-In Lee, *A unified approach to interpreting model predictions*, Proceedings of the 31st International Conference on Neural Information Processing Systems (Red Hook, NY, USA), NIPS’17, Curran Associates Inc., 2017, pp. 4768–4777.
- [47] Pascal Maniriho, Abdun Naser Mahmood, and Mohammad Javed Morshed Chowdhury, *A study on malicious software behaviour analysis and detection techniques: Taxonomy, current trends and challenges*, Future Generation Computer Systems **130** (2022), 1–18.

- [48] Manu, *A comprehensive guide to attention mechanism in deep learning for everyone*, Jan 2021, (Accessed on 06/04/2023).
- [49] Manu, *A simple overview of rnn, lstm and attention mechanism*, Jan 2021, (Accessed on 06/04/2023).
- [50] Raheel Ahmed Memon, Jian Ping Li, Junaid Ahmed, Muhammad Irshad Nazeer, Muhammad Ismail, and Khursheed Ali, *Cloud-based vs. blockchain-based iot: a comparative survey and way forward*, *Frontiers of Information Technology & Electronic Engineering* **21** (2020), no. 4, 563–586.
- [51] Christoph Molnar, *Interpretable machine learning: A guide for making black box models explainable*, Aug 2023, (Accessed on 06/04/2023).
- [52] Nour Moustafa, *Toniotdatasets*, 2019.
- [53] Euclides Carlos Pinto Neto, Sajjad Dadkhah, Raphael Ferreira, Alireza Zohourian, Rongxing Lu, and Ali A. Ghorbani, *Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment*, *Sensors* **23** (2023), no. 13.
- [54] Netscout, *Ip/icmp fragmentation attacks*.
- [55] Mark Mbock Ogonji, George Okeyo, and Joseph Muliaro Wafula, *A survey on privacy and security of internet of things*, *Computer Science Review* **38** (2020), 100312.
- [56] Brett Francis Craig Williams Philip Fitzsimons Harish Rajagopalan Ryan Dsouza Olawale Oladehin, Dan Griffin Catalin Vieru, *Iot lens: Edge layer*, Mar 2023, (Accessed on 06/04/2023).
- [57] Radware, *Fragmented ack attack*, (Accessed on 15/04/2024).

- [58] Radware, *What is the mirai botnet ?*, (Accessed on 15/04/2024).
- [59] Mohamad Faiz Razali, Muhammad Nazim Razali, Fawwaz Zamir Mansor, Gopinath Muruti, and Norziana Jamil, *Iot honeypot: A review from researcher's perspective*, 2018 IEEE Conference on Application, Information and Network Security (AINS), 2018, pp. 93–98.
- [60] Tinshu Sasi, Arash Habibi Lashkari, Rongxing Lu, Pulei Xiong, and Shahrear Iqbal, *A comprehensive survey on iot attacks: Taxonomy, detection mechanisms and challenges*, Journal of Information and Intelligence (2023).
- [61] Shipra Saxena, *What is lstm? introduction to long short-term memory*, Jan 2024, (Accessed on 06/04/2023).
- [62] scikit learn.org, *train_test_split*, (Accessed on 06/04/2023).
- [63] Panda Security, *What is dns spoofing? + 5 tips to prevent it*, Mar 2024, (Accessed on 15/04/2024).
- [64] Jayasree Sengupta, Sushmita Ruj, and Sipra Das Bit, *A comprehensive survey on attacks, security issues and blockchain solutions for iot and iiot*, Journal of Network and Computer Applications **149** (2020), 102481.
- [65] Amazon Web Services, *What is hyperparameter tuning?*, Aug 2023, (Accessed on 06/04/2023).
- [66] Yash Shah and Shamik Sengupta, *A survey on classification of cyber-attacks on iot and iiot devices*, 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 2020, pp. 0406–0413.
- [67] Bhawana Sharma, Lokesh Sharma, Chhagan Lal, and Satyabrata Roy, *Explainable artificial intelligence for intrusion detection in iot networks: A deep learning based approach*, Expert Systems with Applications **238** (2024), 121751.

- [68] Eugene Shevchenko, *Attention mechanism for lstm used in a sequence-to-sequence task*, Oct 2023, (Accessed on 06/04/2023).
- [69] Steve Shock, *The 5 layers of iot architecture that give it super powers*, Jan 2023, (Accessed on 06/04/2023).
- [70] Adam Simmons, *Internet of things (iot) architecture: Layers explained*, Nov 2022, (Accessed on 06/04/2023).
- [71] Bhanup singh, *Handling missing data with knn imputer*, Aug 2023, (Accessed on 06/04/2023).
- [72] Li Sun and Qinghe Du, *A review of physical layer security techniques for internet of things: Challenges and solutions*, Entropy **20** (2018), no. 10.
- [73] Syeda Manjia Tahsien, Hadis Karimipour, and Petros Spachos, *Machine learning based solutions for security of internet of things (iot): A survey*, Journal of Network and Computer Applications **161** (2020), 102630.
- [74] Paul J. Taylor, Tooska Dargahi, Ali Dehghantanha, Reza M. Parizi, and Kim-Kwang Raymond Choo, *A systematic literature review of blockchain cyber security*, Digital Communications and Networks **6** (2020), no. 2, 147–156.
- [75] Nguyen Khoi Tran, M. Ali Babar, and Jonathan Boan, *Integrating blockchain and internet of things systems: A systematic review on objectives and designs*, Journal of Network and Computer Applications **173** (2021), 102844.
- [76] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, *Attention is all you need*, 2023.
- [77] Princy Victor, Arash Habibi Lashkari, Rongxing Lu, Tinshu Sasi, Pulei Xiong, and Shahrear Iqbal, *Iot malware: An attribute-based taxonomy, detection mech-*

- anisms and challenges*, Peer-to-Peer Networking and Applications **16** (2023), no. 3, 1380–1431.
- [78] Ly Vu, Quang Uy Nguyen, Diep N. Nguyen, Dinh Thai Hoang, and Eryk Dutkiewicz, *Deep transfer learning for iot attack detection*, IEEE Access **8** (2020), 107335–107344.
- [79] Zhendong Wang, Jingfei Li, Shuxin Yang, Xiao Luo, Dahai Li, and Soroosh Mahmoodi, *A lightweight iot intrusion detection model based on improved bert-of-theseus*, Expert Systems with Applications **238** (2024), 122045.
- [80] Zhibo Wang, Defang Liu, Yunan Sun, Xiaoyi Pang, Peng Sun, Feng Lin, John C. S. Lui, and Kui Ren, *A survey on iot-enabled home automation systems: Attacks and defenses*, IEEE Communications Surveys & Tutorials **24** (2022), no. 4, 2292–2328.
- [81] Ron Winward, *Iot attack handbook*, (Accessed on 15/04/2024).
- [82] Liang Xiao, Yan Li, Guoan Han, Guolong Liu, and Weihua Zhuang, *Phy-layer spoofing detection with reinforcement learning in wireless networks*, IEEE Transactions on Vehicular Technology **65** (2016), no. 12, 10037–10047.
- [83] Sami Yaras and Murat Dener, *Iot-based intrusion detection system using new hybrid deep learning algorithm*, Electronics **13** (2024), no. 6.
- [84] Shiyan Yi, Haichun Liu, Tao Chen, Jianwen Zhang, and Yibo Fan, *A deep lstm-cnn based on self-attention mechanism with input data reduction for short-term load forecasting*, IET Generation, Transmission & Distribution **17** (2023), no. 7, 1538–1552.

- [85] Bruno Bogaz Zarpelão, Rodrigo Sanches Miani, Cláudio Toshio Kawakani, and Sean Carlisto de Alvarenga, *A survey of intrusion detection in internet of things*, Journal of Network and Computer Applications **84** (2017), 25–37.

Vita

Candidate's full name: Tinshu Sasi

University attended (with dates and degrees obtained):

Bachelor of Technology (Computer Science & Engineering) Manav Rachna International University 2010 - 2014

Publications:

Tinshu Sasi, Arash Habibi Lashkari, Rongxing Lu, Pulei Xiong and Shahrear Iqbal, **"A comprehensive survey on IoT attacks: Taxonomy, detection mechanisms and challenges"**, submitted to Journal of Information and Intelligence (ScienceDirect).

Princy Victor, Arash Habibi Lashkari, Rongxing Lu, Tinshu Sasi, Pulei Xiong and Shahrear Iqbal, **"IoT malware: An attribute-based taxonomy, detection mechanisms and challenges"**, submitted to Peer-to-Peer Networking and Applications (SpringerLink).