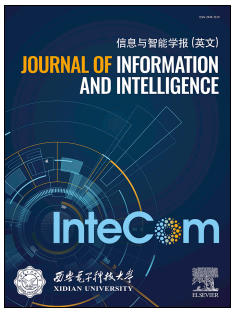
# Journal Pre-proof




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

Tinshu Sasi, Arash Habibi Lashkari, Rongxing Lu, Pulei Xiong, Shahrear Iqbal

PII: S2949-7159(24)00076-3

DOI: https://doi.org/10.1016/j.jiixd.2024.09.001

Reference: JIIXD 71

To appear in: *Journal of Information and Intelligence*

Received Date: 12 June 2024

Revised Date: 28 August 2024

Accepted Date: 9 September 2024

Please cite this article as: Sasi T., Lashkari A.H., Lu R., Xiong P. & Iqbal S., An Efficient Self Attention-Based 1D-CNN-LSTM Network for IoT Attack Detection and Identification Using Network Traffic, *Journal of Information and Intelligence*, https://doi.org/10.1016/j.jiixd.2024.09.001.

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

Tinshu Sasi[a], Arash Habibi Lashkari[a,b], Rongxing Lu[a,*], Pulei Xiong[c], Shahrear Iqbal[c]

[a]*Canadian Institute for Cybersecurity (CIC), Faculty of Computer Science, University of New Brunswick (UNB), Fredericton, Canada*
[b]*Behaviour-Centric Cybersecurity Center (BCCC), School of Information Technology, York University, Toronto, Ontario, Canada*
[c]*National Research Council (NRC), Canada*
*(tinshu.sasi@unb.ca, ahabibil@yorku.ca, rlu1@unb.ca, Pulei.Xiong@nrc-cnrc.gc.ca, Shahrear.Iqbal@nrc-cnrc.gc.ca)*

[*]Corresponding author: rlu1@unb.ca

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

**Abstract**

In the last ten years, the IoT has played a crucial role in society's digital transformation. However, because of the wide range of devices it encompasses, it is also facing increased security vulnerabilities. This research presents a novel mechanism called the Self-attention-based 1D-CNN-LSTM, which uses Convolutional Neural Networks (CNNs) combined with a Long Short-Term Memory (LSTM) model enhanced with a Self-Attention mechanism for detecting IoT attacks. The proposed mechanism achieves high accuracy and efficiently differentiates malicious and benign network traffic. By employing Shapley Additive Explanations (SHAP), we identified 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 nine augmented IoT tabular datasets named CIC-BCCC-NRC_TabularIoTAttack-2024 from accessible IoT datasets to evaluate the model's robustness and showcase its efficacy in IoT security.

*Keywords:* IoT Attacks, Machine Learning, Artificial Intelligence, IoT Attack Detection

## 1. Introduction

The digital revolution has substantially transformed our lives, with the Internet of Things (IoT) playing a prominent role. However, the rapid development of IoT in various aspects of life has led to numerous emerging cybersecurity threats. Consequently, detecting and preventing potential attacks in IoT networks has recently attracted significant interest from academia and industry. Among the various attack detection approaches, machine learning-based methods, especially deep learning, have demonstrated outstanding potential due to their early detection capabilities [1].

Over the past several years, there has been a substantial increase in the number of attacks specifically targeting IoT devices. These attacks encompass various types of cyber threats, 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, posing multiple safety risks; and manipulating the Controller Area Network (CAN) bus of vehicles to potentially alter speed and direction, thereby posing a threat to public safety. An investigation into smart home hacking revealed that, over one week, fraudsters and unidentified groups launched over 12,000 attacks against various smart home devices, including TVs, thermostats, smart kettles, and security systems [2, 3].

### *1.1. What are IoT attacks?*

IoT attacks constitute cyber-attacks leveraging IoT devices to access consumers' sensitive data. Typically, attackers deploy malware on these devices, causing damage or infiltrating additional organizational 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 cyber-attacks. Hackers and organizations can exploit common flaws and "zero-day exploits" to attack IoT devices in various ways [4].

### *1.2. Motivation*

The primary motivation for addressing the IoT attack detection problem lies in the critical need to protect IoT systems and networks from potential security vulnerabilities. As IoT devices become increasingly pervasive across various aspects of daily life, they are increasingly targeted by cyber attacks. The capability to accurately identify and differentiate between malicious and benign instances within IoT network traffic is vital for maintaining the integrity and reliability of these interconnected devices and services. Given the escalating complexity and volume of IoT network data, traditional security measures are often inadequate. Therefore, developing robust detection methodologies is essential for enhancing the security of IoT systems, preventing potential breaches, and ensuring the continued trust and functionality of IoT technologies in our daily lives.

The problem of IoT attack detection by identifying and differentiating between malicious and benign instances is multifaceted and encompasses several key challenges:

- **Feature Selection:** Identifying the optimal features that can accurately distinguish between malicious and benign instances is critical. This involves analyzing a wide range of features and determining those most indicative of malicious behavior.
- **Efficacy and Reliability:** Establishing the efficacy and reliability of these features for detection purposes is a significant challenge. This requires rigorous testing and validation to ensure that the selected features consistently and accurately identify malicious activity, minimizing false positives and negatives.
- **Methodology Development:** Developing robust detection methodologies capable of handling the escalating complexity and volume of IoT network data is imperative. These methodologies must be adept at processing large amounts of data in real-time and adapting to new and evolving threats.
- **System Integration:** Ensuring the seamless integration of the developed methodologies into existing IoT systems and networks without causing disruptions or performance issues is also a crucial consideration.

By addressing these challenges, we aim to enhance the security and reliability of IoT systems, making them more resilient to potential cyber threats. This will safeguard the interconnected devices that play an increasingly central role in our lives.

To address the multifaceted problem of identifying and differentiating between malicious and benign instances in IoT network traffic, specifically focusing on feature selection, efficacy and reliability, and methodology development, we propose a Self-attention-based 1D-CNN-LSTM Network. This network leverages Convolutional Neural Networks (CNNs), Residual Networks, and Long Short-Term Memory (LSTM) networks, in conjunction with attention mechanisms, to enhance the detection of IoT attacks in TCP network flows. The proposed approach capitalizes on the robust feature extraction capabilities of CNNs, the effective sequence learning abilities of LSTMs, and the precise contextual insights provided by self-attention mechanisms, resulting in a comprehensive and effective framework.

In summary, this paper makes the following contributions:

- We introduce a novel Self-attention-based 1D-CNN-LSTM network to detect IoT attacks by analyzing time-related features extracted from TCP flow data.
- To ensure the robustness of our model, we evaluate its performance on nine publicly available external datasets, which have been enhanced and pre-processed using CICFlowmeter.
- We aim to identify the optimal combination of hyperparameters that yield the highest performance metrics for our model.
- We employ Shapley Additive Explanations (SHAP) to calculate feature importance scores, which are then used to identify the most significant features from the extracted feature set.
- By retraining the model on the most significant features, we reduce the model size while maintaining performance comparable to the original model.
- We release the CIC-BCCC-NRC_TabularIoTAttack-2024 dataset, which includes over 80 extracted features from nine IoT datasets.

The subsequent sections of this paper are structured as follows. In Section 2, foundational insights into the subject matter are provided, elucidating key technical terminologies pertinent to IoT attacks. This section also encapsulates an overview of the IoT attacks referenced in the training datasets. Section 3 delves into a comprehensive review of prior scholarly works on IoT attacks, exploring the methodologies employed for detection utilizing machine learning and deep learning techniques. Furthermore, it presents an exhaustive classification of various types of IoT attacks. The rationale behind the proposed methodology is expounded upon in Section 4, elucidating the preprocessing steps and detailing the architectural framework, with a comprehensive breakdown of each constituent component. Section 5 offers an in-depth analysis of the research outcomes. This segment encompasses discussions on the dataset, experimental setup, features utilized, evaluation metrics employed, and the results garnered from the proposed methodology. Finally, Section 6 serves as the concluding segment of the paper, encapsulating a summary of the contributions made, addressing encountered challenges, and providing insights into potential avenues for future research endeavors.

## 2. Preliminaries

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 [4].

### *2.1. IoT architecture*

The IoT architecture refers to the organization and setup of IoT devices to fulfil 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 IoT presents further difficulties regarding interoperability, security, and several other issues. The IoT architecture can include up to seven levels. [5]. Figure 1 provides a detailed illustration of the 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 [6].

- *Transport Layer/ Network Layer*: The Transport/Network layer is responsible for transferring data to the processing systems of the subsequent layer [6]. 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-Power Wireless Personal Area Networks), MQTT (Message Queue Telemetry Transport), CoAP (Constrained Application Protocol), DDS (Data Distribution Service), AMQP (Advanced Message Queuing Protocol) [5].

- *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 [7].

- *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 [6]. 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 [8].

Architecture Levels.pdf

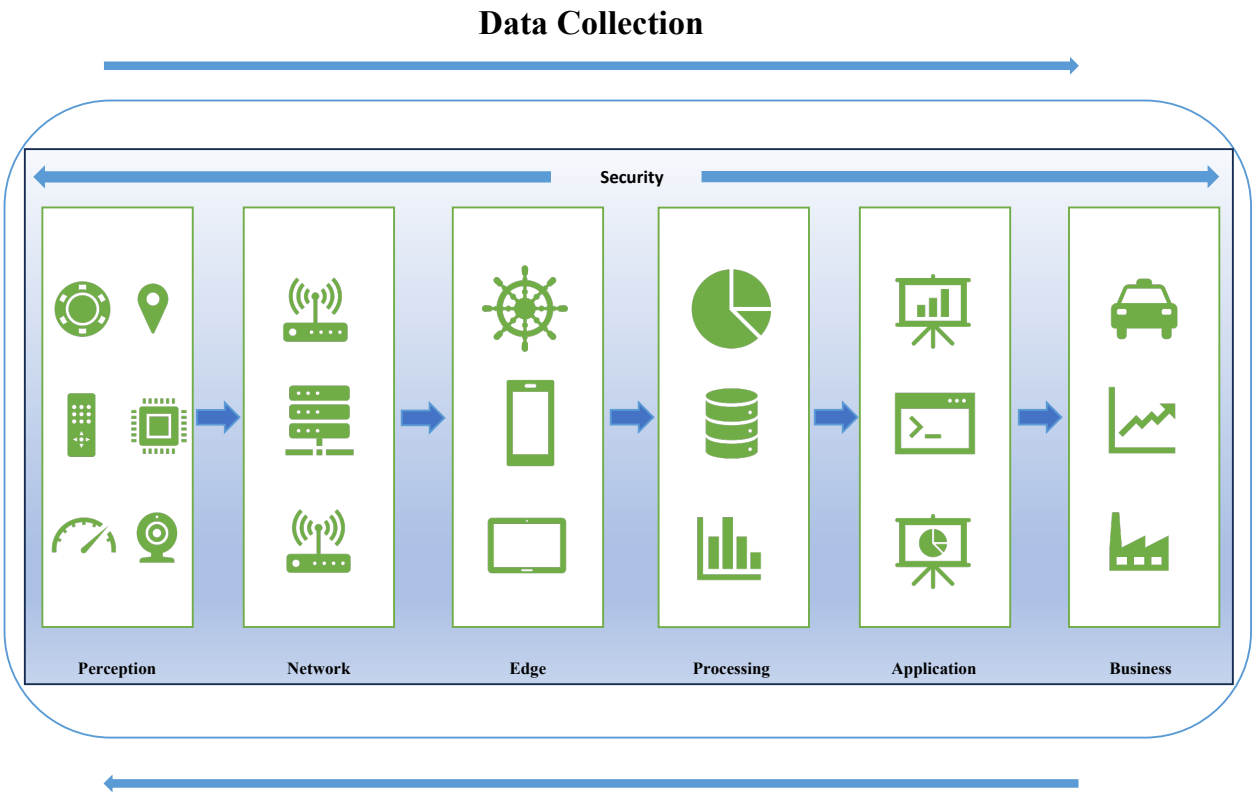



Figure 1: IoT Architecture Levels

- *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 [9].
- *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 [5] [8] [9].
- *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 IoT devices often deal with confidential information, it is crucial to implement strong security measures [5] [9].

- *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 [4].
- *Exposure*: It refers to a problem or mistake within the system configuration that allows an unauthorized person to undertake actions to obtain information [4].
- *Threats*: It refers to an intentional or unintentional action that exploits vulnerabilities within a system[4].
- *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 [4].

### *2.2. Common IoT Vulnerabilities*

There are several vulnerabilities in IoT networks, but according to OWASP, ten significant vulnerabilities make IoT devices susceptible to attacks are (Table 2) [10]:

- *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. 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. IoT or IT attacks*

Table 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, 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. 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. [4].
- *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. [4].
- *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. [4].
- *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. [4].

Table 2: Top 10 OWASP IoT Vulnerabilities

| **Rank** | **Vulnerability** | **Description** |
|---|---|---|
| 1 | Weak, Guessable, or Hardcoded 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, backend 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. |
| 10 | Lack of Physical Hardening | Absence of Physical Security Measures, enabling attackers to access critical information or assume local control. |

## 3. Related Works

Although there has been a lot of research on attacks targeting the 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 [4]. This section first provides an overview of previous research on IoT attacks and then discusses the methods utilized to carry out IoT attack detections, including Machine and Deep Learning.

Many academic publications thoroughly analyze several aspects of risks and attacks in the IoT field. The mentioned literary works [11] [12] [13] [14] [15] extensively examine the classification of risks and intrusions linked to the IoT. These papers primarily focus on two main categories: the architectural features of the 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.

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: [16], [11], [17], [18], [19], and [20].

The authors of the study [21] did thorough research on security vulnerabilities in the 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 [22] in 2018 to explore several attack strategies that target the 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 [23].

### *3.1. 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 2 provides a thorough overview of several techniques for detecting attacks in the 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 IoT is employed in the betting and gambling sector to discover insider trading cases by analyzing trade activity patterns [24].

- *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 [25].

  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 [25].

- *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 [26].

- *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 [27].

- *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 [4].

- *Deep Learning*: Deep learning (DL), a branch of machine learning (ML), provides sophisticated capabilities for identifying attacks in the IoT by automatically extracting complex features from raw data without requiring manual feature engineering. CNNs and recurrent neural networks (RNNs) are frequently used in deep learning-based intrusion detection systems to secure the 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, 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 IoT contexts. This makes them highly suitable for identifying familiar and new IoT attacks [4].

To summarize, 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 IoT.

## 4. Proposed Model

This section presents the motivation for designing a model architecture called Self-attention-based 1D-CNN-LSTM, using a CNN combined with an LSTM model enhanced with a Self-Attention mechanism to identify IoT attacks in TCP network traffic flow data. A one-dimensional convolutional neural network (1D CNN), compared with 2D-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.

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.

The proposed approach consists of two distinct steps. The initial phase involves converting data, extracting features, generating datasets for training and testing the model, and developing the model itself. The second module involves training the model using the training dataset, incorporating all features, and determining feature importance by utilizing Shapley Additive explanations (SHAP) values. Subsequently, the model is retrained using a decreased set of features while maintaining comparable performance metrics to the original model.

### *4.1. Pre-processing - Stage I*

This section will provide a detailed analysis of the components involved in the pre-processing stage technique, and Figure 3 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 [28].

- *Data Conversation and Feature Extraction*: As previously outlined in Section 4.1.1, 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 [29]. In order to handle the missing data in the ”Flow Bytes/s” and ”Flow Packets/s” columns of our dataset, we choose to utilize KNN Imputation to fill in these missing values. After combining these CSV files, we created the required dataset.

- *Model Creation*: Described in detail in section 4.1.3

#### *4.1.1. Feature Extraction using 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 [30].

*4.1.2. Data Pre-Processing*

During the pre-processing stage, we identify a dataset that consists of raw benign and malicious pcap files in order to extract features. The CICflowmeter can be executed either through the command line or within Java Integrated Development Environments (IDEs) such as IntelliJ or Eclipse. When we start the application, we have the option to select either offline or online mode. The offline option allows us to input raw Pcap data into the application for analysis. During this process, we extract features from the Pcap files and save them as CSV files. From the Pcap files, we can extract 83 temporal statistical features, which is a significantly larger number compared to the features provided in the pre-processed CSV files by the dataset authors.

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 [29]. 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 [31].

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 [32].

*4.1.3. Model Structure*

In this section, we examine the components utilized in the model's structure. 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:

- *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 and Figure 5 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.

*4.1.4. Training and Tuning - Stage II*

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.

- *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 [33].

  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 3 of hyperparameters utilized for training the model.

Table 3: Hyperparameters List

| No | Activations | 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 [34].

- *SHAP Feature Importance*: 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 10 provides a visual representation of these insights.

  SHAP 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 [35] [36].

  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 [36].

  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.

- *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.

## 5. Experiments & Results

This section provides 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 Section 4, this section also demonstrates the experimentation and results of the proposed Self-attention-based 1D-CNN-LSTM deep learning technique.

Further in the section, 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.

### *5.1. 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 public libraries.

### *5.2. Datasets*

Nine augmented datasets have been generated using the publicly available IoT datasets to train, test, and evaluate the proposed model to maximize the variety of attacks and benign activities. Below is a summary of datasets augmented during preprocessing for training, validation, and model testing, summarised in table Table 4.

- *ACI IoT Network Traffic Dataset 2023*: This study aims to introduce a novel dataset tailored explicitly for machine learning (ML) applications in the domain of 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 [37].

  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 [37].

- *CIC IoT dataset 2022*: This study endeavors to produce an advanced dataset to profile, analyze behavior, 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 cardsone 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 [38].

  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 [38].

- *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 [39].

- *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[40].

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 [40].

- *IoT Network Intrusion Dataset*: This dataset features various network attacks within an 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 [41].

- *MQTT-IoT-IDS 2020*: This study aims to develop a dataset pertaining to the 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 [42].

  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 [42].

- *TON IoT*: The TON_IoT datasets represent a new iteration of Industry 4.0/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 [**?** ].

- *UQ IoT IDS 2021*: The UQ-IoT-IDS-2021 dataset, provided by the University of Queensland (UQ), Australia, primarily focuses on cybersecurity within the 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 [43].

Table 4: 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 | 46.3 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 | 696.2 MB |
| 2022 | CIC-BCCC-NRC_UQ-IoT-2022 | UQ IoT 2022 | Yes | Real | 2.31 GB |

### *5.3. Summary of IoT Attacks in Training dataset*

The selected training dataset includes 14 common IoT attacks as follows, and Appendix AppendixA lists the malicious activities in all selected datasets.

1. *ACK Fragmentation*: Fragmented ACK attacks, a variant of ACK & PSH-ACK Flood, use 1500-byte packets to dominate network bandwidth at a low packet rate. These packets pass through routers, ACLs, and firewalls undetected, exhausting resources as they require reassembly. Containing irrelevant data, they aim to crash the target, disrupting services similar to other DDoS attacks [44].
2. *ICMP Flood*: An ICMP flood is a DoS attack that uses pings to analyze a device's operation. A "ping flood attack" occurs when attackers overload a network router or IP address with ICMP packets, preventing it from handling legitimate traffic. This exhausts the device's resources (memory, processing power, and interface rate), hindering service to actual requests [45].
3. *ICMP Fragmentation*: IP/ICMP fragmentation volumetric DoS attacks flood networks by breaking IP datagrams into smaller packets, which are reassembled during communication. This method meets network size limits, governed by the maximum transmission unit (MTU). Packets exceeding this limit must be fragmented, with only one packet containing essential information. Attackers use IP fragmentation to target communication and security systems, often sending forged fragments that cannot be reassembled, depleting memory resources.

These attacks use fake UDP or ICMP packets designed to exceed the MTU, quickly exhausting server resources and making it unavailable for legitimate traffic [46].

4. *PSH ACK Flood*: ACK or PUSH ACK packets facilitate bidirectional data transfer between host and client until the session ends. ACK flood attacks target vulnerable servers with bogus ACK packets from non-existent sessions. This forces the server to use resources to verify the packets, leading to decreased performance and service availability [47].
5. *DDoS RST FIN Flood*: Clients and hosts use RST or FIN packets to end TCP SYN sessions. In an RST or FIN flood, the targeted server receives numerous counterfeit packets unrelated to its connections. The server must use significant resources to match these packets, reducing performance and causing partial unavailability [48].
6. *SYN Flood*: SYN floods, or half-open attacks, are DDoS attacks that deplete a server's resources by flooding it with SYN packets, causing slow or no response to legitimate traffic [49].
7. *Synonymous IP Flood*: This attack floods the target server with fake TCP SYN packets using the victim's source and destination addresses, forcing the server to use system resources to process each packet [50].
8. *DDoS TCP Flood*: TCP SYN Flood attacks exploit the TCP three-way handshake by sending numerous SYN requests without completing the final ACK, causing the server to waste resources on partially formed connections [51].
9. *UDP Flood*: A UDP Flood attack uses the User Datagram Protocol to launch a volumetric DoS attack by flooding the victim system with UDP packets to random ports. This forces the host to scan for active programs and respond with ICMP packets, exhausting resources and preventing access for authorized users. Attackers may spoof IP addresses to conceal their location [52].
10. *UDP Fragmentation*: A variation of UDP flood, this attack uses the largest allowable packets to overload the channel with fewer packets. These counterfeit packet fragments force the server to waste resources trying to reassemble non-existent packets, leading to resource depletion, server crashes, or channel overflow. This attack is hard to filter and poses a high risk of channel overflow [53].
11. *DNS Spoofing*: A hacker uses DNS spoofing to trick a machine or network into communicating with a fake website or server, leading to potential identity theft, financial fraud, malware, and other online issues [54].
12. *HTTP Flood*: An HTTP flood is a DDoS attack that uses legitimate HTTP GET or POST requests to overwhelm a web server or application. These volumetric attacks often involve a botnet and do not rely on defective packets, spoofing, or reflection. They require less bandwidth to take down a target and must be carefully adapted to the specific site or application, making detection and prevention difficult [55].
13. *Mirai*: The Mirai botnet software turns IoT devices into "bots" for launching powerful volumetric DDoS attacks. It targets devices with open ports or default credentials, infecting them with malware. Infected devices join the Mirai botnet, allowing attackers to issue commands from a central "command & control" server. This enables simultaneous DDoS attacks on websites, networks, and digital infrastructure using the collective power of all Mirai bots [56].

    - *Mirai GRE-ETH Flood*: GRE creates IP-based virtual point-to-point connections and can encapsulate many network layer protocols. DDoS scrubbing services use GRE for mitigation. The GRE-ETH Flood attacks routers and switches with excessive GRE and ETH frames, draining CPU, memory, and bandwidth. The goal is to interrupt network services, cause DoS or DDoS, and exploit weaknesses in the target device's traffic management [57].
    - *Mirai GRE-IP Flood*: A GRE-IP Flood is a network attack that overwhelms a network with GRE and IP packets. The attacker sends numerous GRE-encapsulated IP packets to exhaust network resources, causing congestion, slow speeds, or service disruptions. Goals include interrupting network operations, causing DoS, or exploiting infrastructure weaknesses [57].
    - *Mirai UDP Plain*: UDP Plain attacks, or UDP flood attacks, are network-based DoS attacks that flood a server or network with UDP packets. Since UDP is connectionless and does not require a handshake, attackers can quickly produce and transmit packets. This overwhelms the target, causing network delays, degraded services, or restricted access. The attack exploits UDP's statelessness, potentially causing packet loss or misorder, further disrupting communication [52].

14. *MITM ARP Spoofing*: Attackers intercept network traffic via ARP spoofing (ARP poisoning). After accessing the network, they identify the IP addresses of devices like a workstation and a router. The attacker then sends false ARP responses, making the devices update their ARP cache to reroute traffic through the attacker's device. This allows the attacker to eavesdrop on conversations, hijack sessions, tamper with communication, and initiate DDoS attacks [58].

### *5.4. Features*

In this research, we used CICFlowMeter to extract the network-based features from benign and malicious PCAP files for only TCP network flow data, which have been listed in Appendix AppendixB.

### *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. Subsequently, a Self-attention-based 1D-CNN-LSTM mechanism was constructed, and hyperparameter tuning was performed using the Grid Search technique. This process identified the top 5 hyperparameters indicated in Table 3.

We evaluated five models across nine 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 13, the results indicated only a minor performance decrement between the initial model and those with the reduced feature set.

### *5.6. Experimental Results*

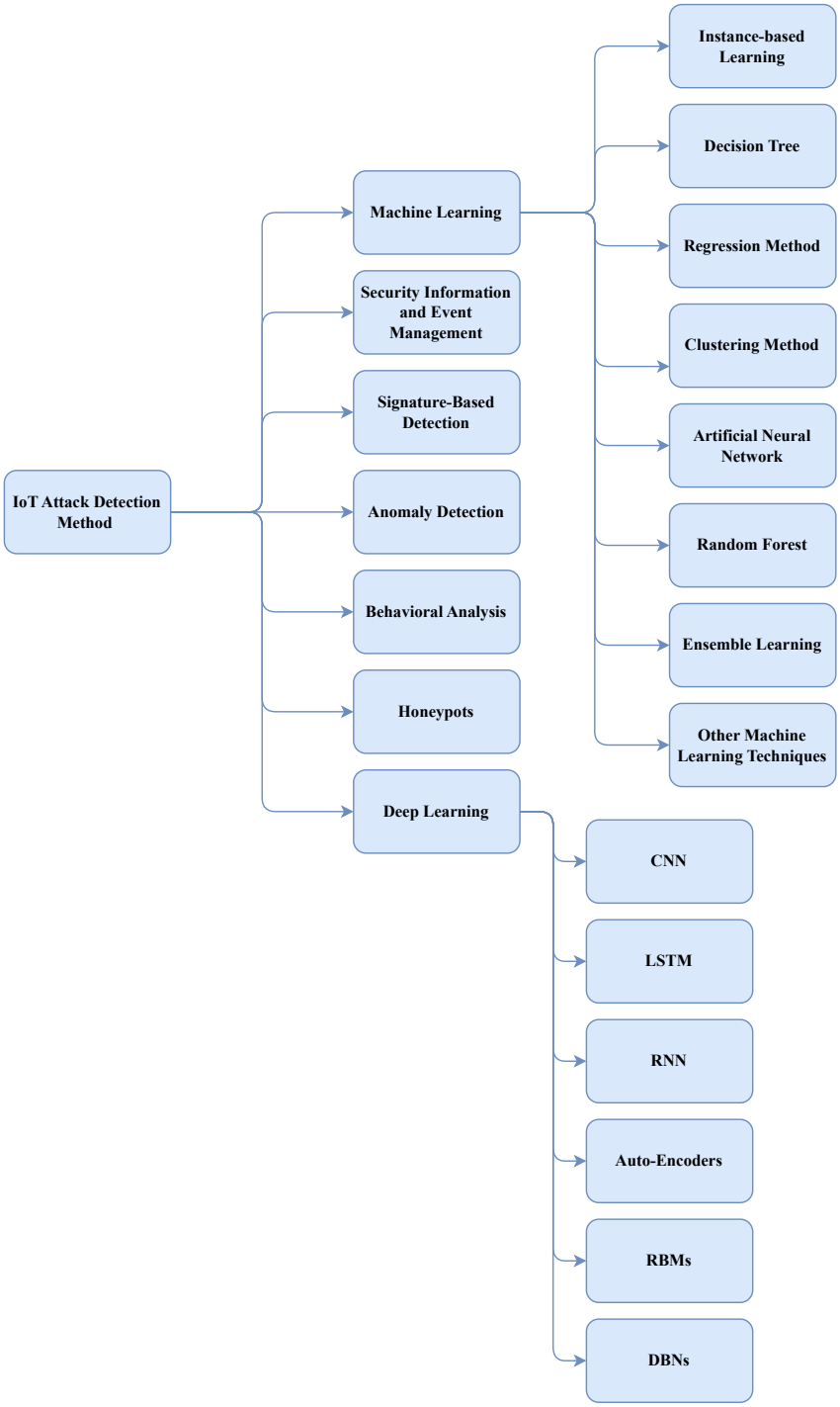
This section presents our experiment findings regarding identifying malicious activities within network TCP flow data. Due to the predominance of malicious activities over benign ones, a data balancing procedure was necessitated prior to model training. Specifically, we retained all rows corresponding 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 3 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 between our model and baseline models across all datasets mentioned in Table 4. The outcomes of these analyses are displayed in Table 5 and Figure 6.

Table 5: Baseline Models Evaluation Results based on Accuracy

| **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 |
| **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 |

| Dataset | CNN | CNN-LSTM | CNN-LSTM-ResNet | CNN-LSTM-ResNet-Additive Attention | CNN-LSTM-ResNet-Self Attention |
|---|---|---|---|---|---|
| **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 |
| **CIC-BCCC-NRC_IoMT-2024** | 0.9863 | 0.9899 | 0.5045 | 0.5873 | 0.9988 |




and Machine Learning$_1$.$pdf$

Figure 2: Overview of Available IoT Attack Detection and Identification Methods

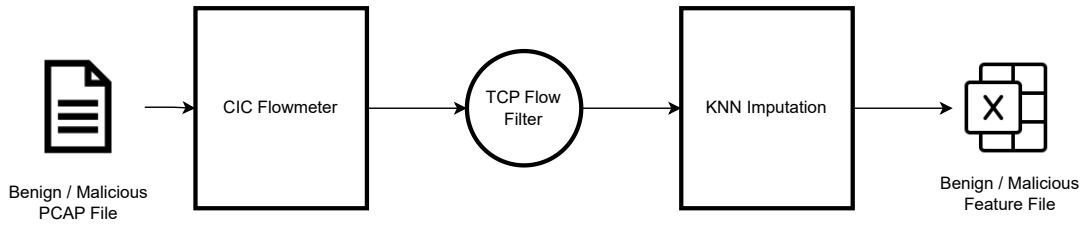



Figure 3: Data Pre-processing

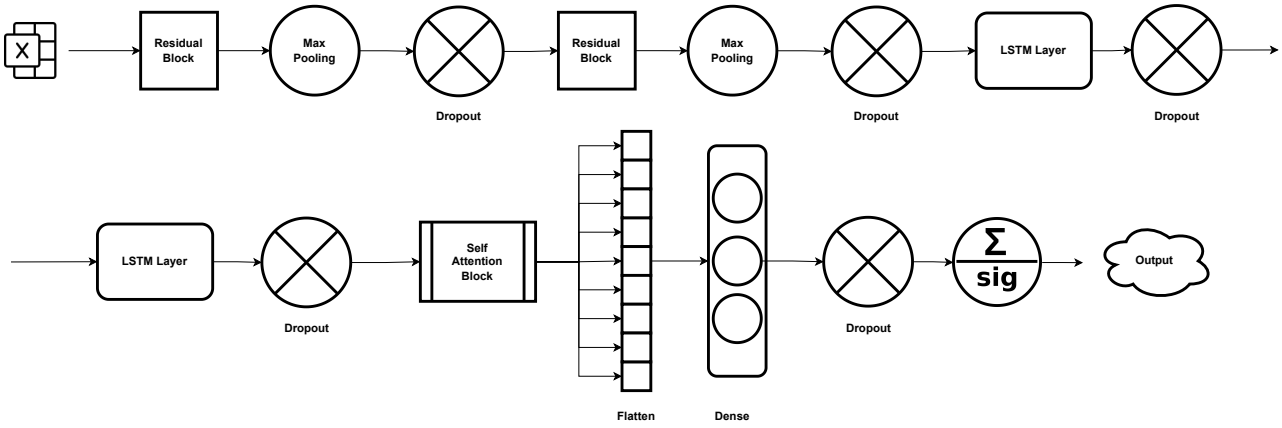



Figure 4: Model Architecture

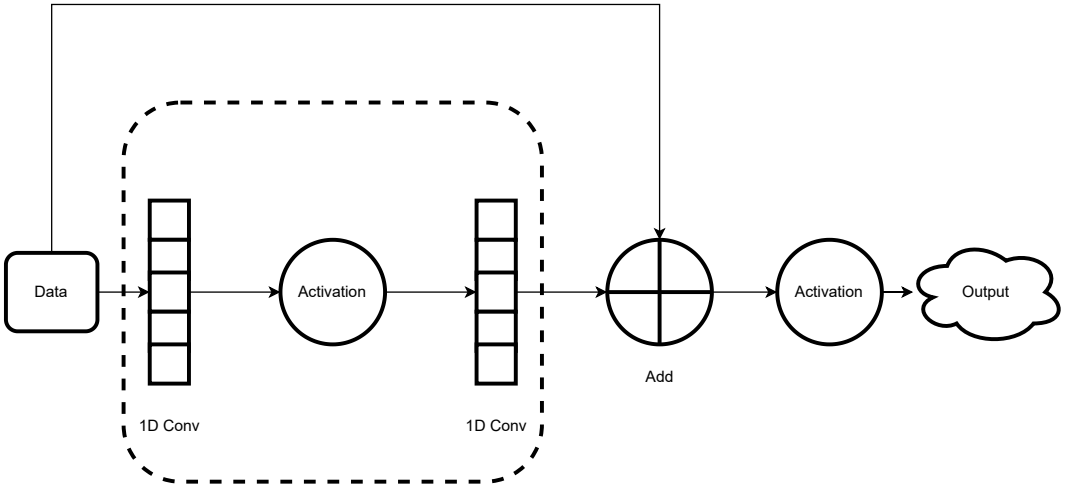



Figure 5: Residual Block

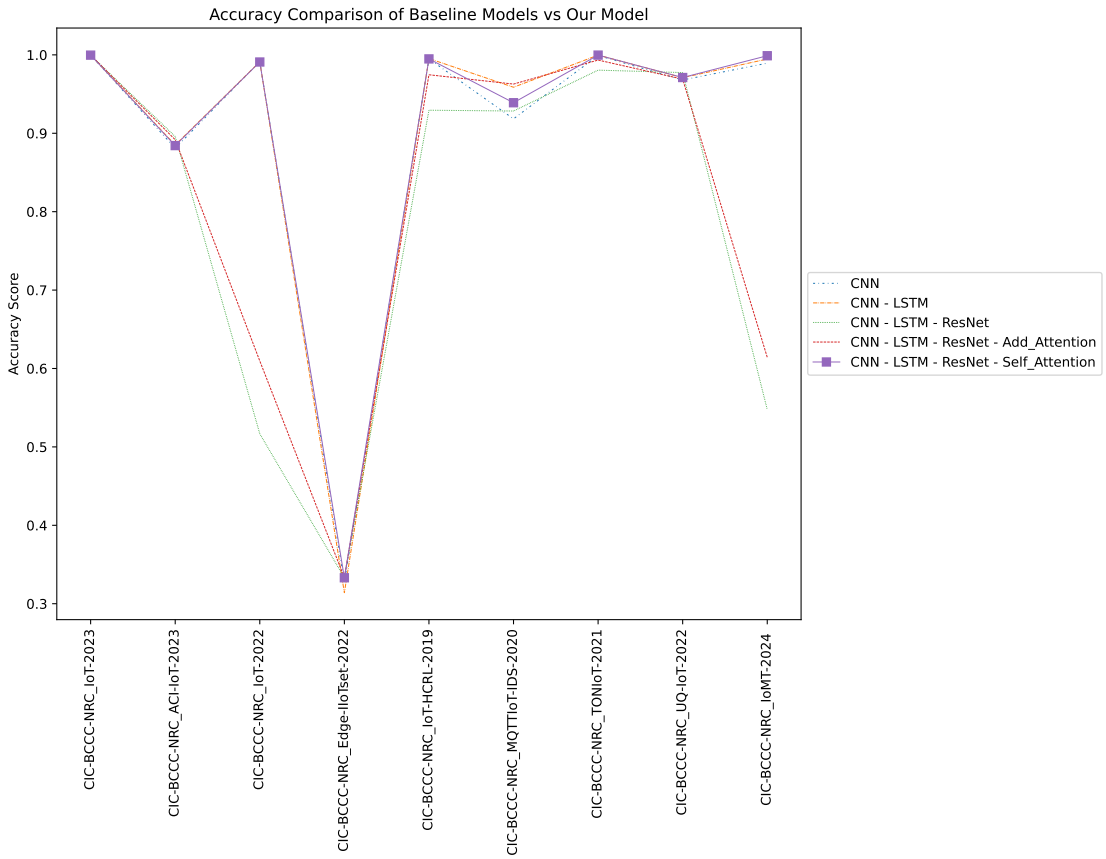



Figure 6: Accuracy Comparison between Baseline Models and Our Model

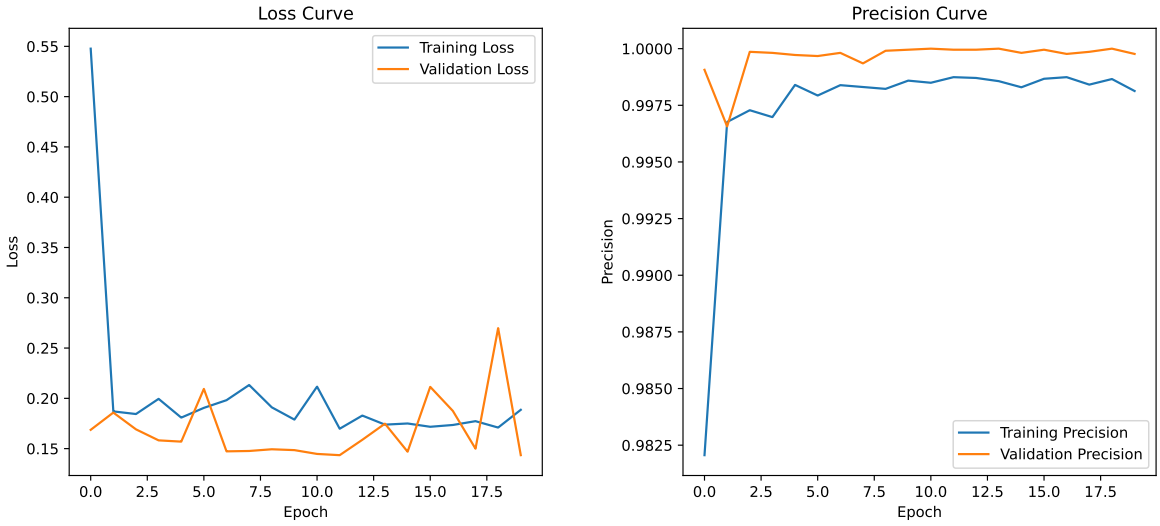



Figure 7: Our Model's Training History Results

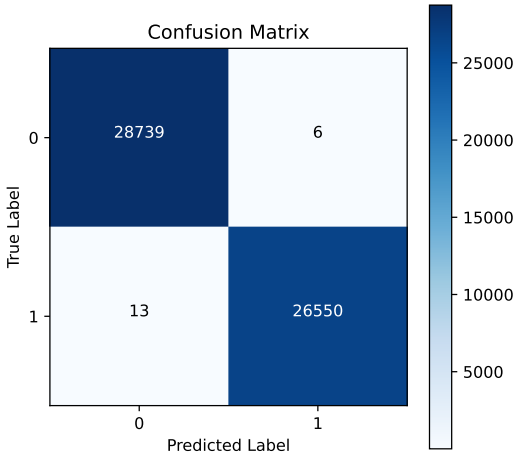



Figure 8: Our Model's Confusion Matrix Results

The training history of our main model and the confusion matrix of the main model are described in ‘Figure 7 and Figure 8, respectively. After evaluating our model using the CIC-BCCC-NRC_IoT-2023 test data and the dataset mentioned in Table 4, we obtained the results summarized in Table 6. The confusion matrix for the initial model across all datasets is provided in Table 7.

Table 6: 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 |

| **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 |

Table 7: Confusion Metrics For Initial Model Across All Datasets

After completing the evaluation process, SHAP analysis is employed to compute feature importance, facilitating the identification of key features that contribute to the model’s predictive outcomes. Subsequently, Figure 9 and Figure 10 illustrate the summary plot and feature importance scores derived from analyzing a randomly selected subset of 100 test data samples.

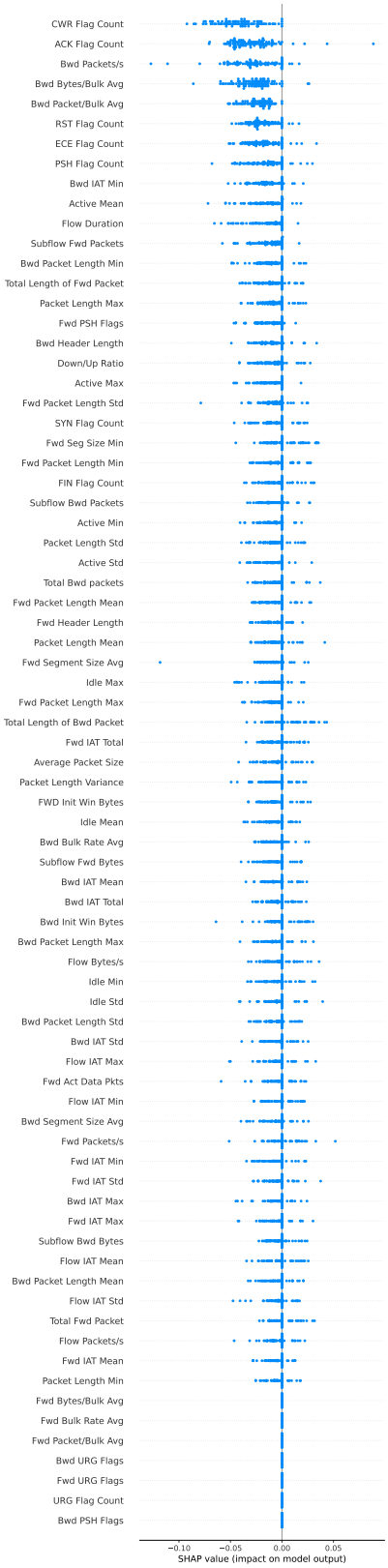
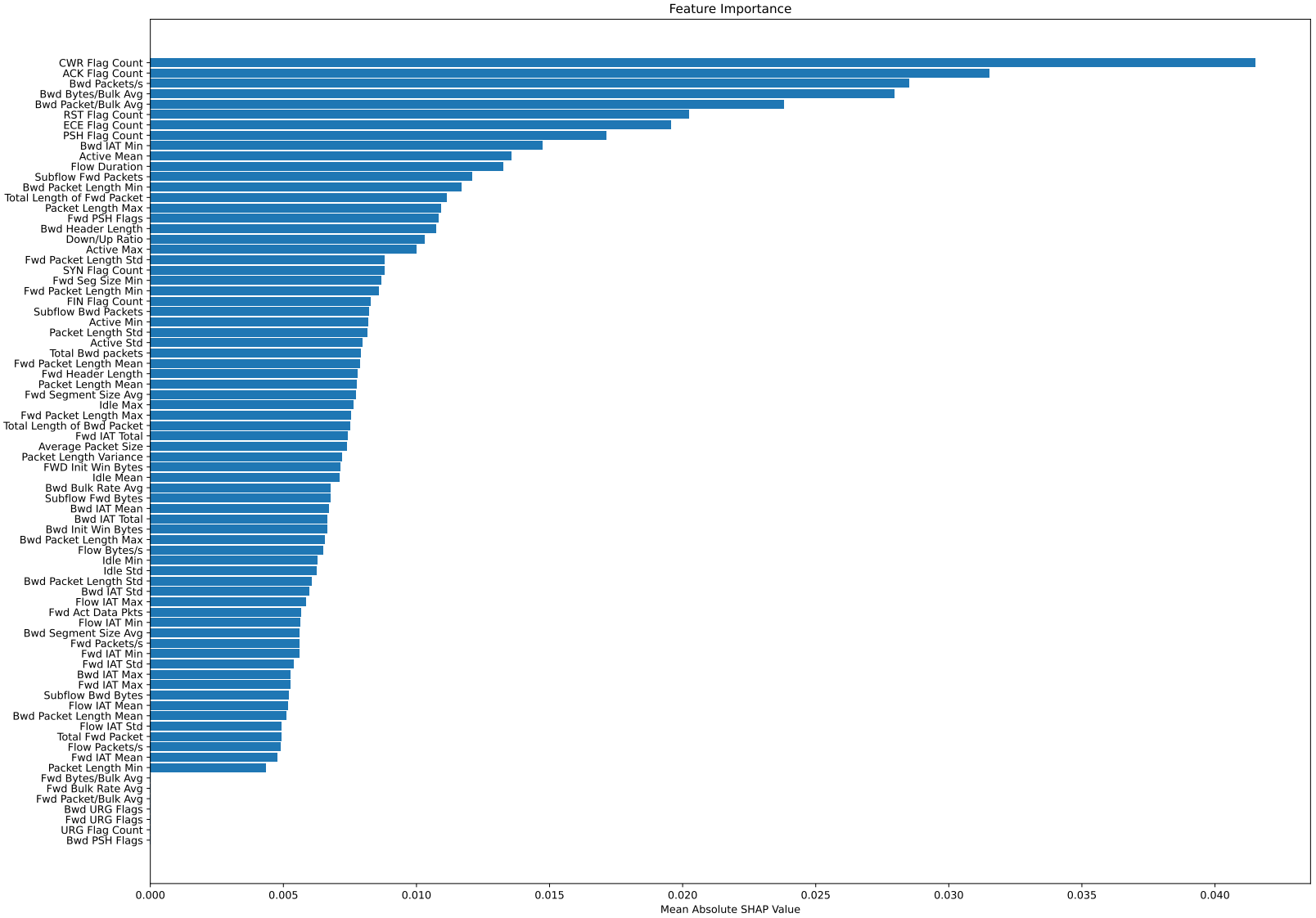



Figure 9: SHAP Summary Plot for Our Model (Trained on CIC-BCCC-NRC_IoT-2023 Dataset)

Figure 10: Feature Importance Graph



Due to the standard scaling and balancing of the training dataset, the SHAP values may appear smaller, indicated by the blue colour in Figure 9, as the features are normalized to have a mean of 0 and a standard deviation of 1. This normalization prevents any single feature from dominating the prediction, resulting in a more distributed contribution across multiple features. In addition, we incorporated regularization approaches, such as dropout, which could potentially diminish the influence of particular features, resulting in decreased SHAP values. Models often perform better with scaled features because scaling can lead to faster convergence and better numerical stability during training. Consequently, SHAP values derived from models trained on scaled data might be more consistent and reliable [59] [60] .

Nevertheless, when analyzing SHAP values derived from scaled features, it is crucial to bear in mind that they represent the influence of variations in the scaled (standardized) feature values, rather than the original feature values. The direction of the impact (positive or negative) remains consistent, but the magnitude needs to be interpreted in the context of the scaled features [61].

In the context of our research problem, this Figure 9 aids in comprehending the features that influence the model's prediction towards the positive class (such as identifying a malicious event in IoT traffic) and the features that influence the prediction towards the negative class (such as benign traffic). Positive SHAP values, located on the right side of the x-axis, enhance the likelihood of the positive class, whereas negative SHAP values, found on the left side, diminish it. This interpretation facilitates the diagnosis and validation of the model's behaviour, ensuring that the model's decisions are made based on relevant features.

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 10 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 4. Through this evaluation process, we ascertain that a minimum of 7 features is necessary to achieve performance akin to the initial model's, as shown in Table 8.

Table 8: Reduced Set's Features Accuracy List

| Dataset | 4 Features | 5 Features | 7 Features | 10 Features | 11 Features |
|---|---|---|---|---|---|
| **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 |
| **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_TONIoT-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 11) and confusion matrix (Figure 12) for the CIC-BCCC-NRC_IoT-2023 dataset, along with confusion matrices for the datasets referenced in Table 4 are presented in Table 9.

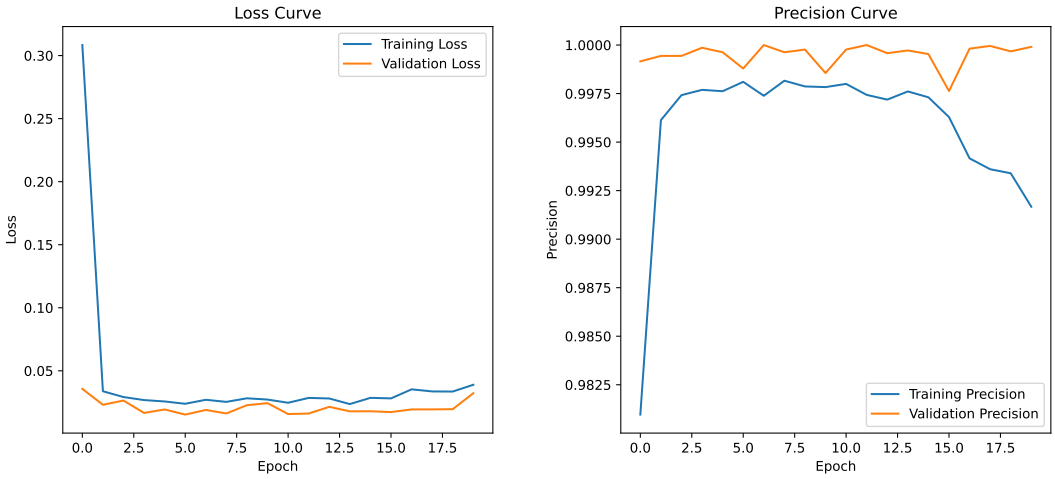



Figure 11: Reduced Set Feature's Training History Results

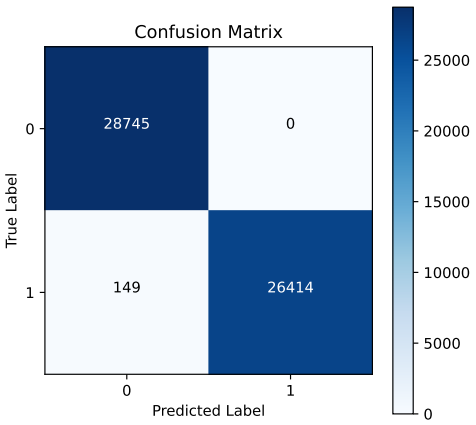



Figure 12: Reduced Set Feature's Confusion Matrix Results

| 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-IIoTset-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 9: Confusion Metrics For 7 Feature Reduced Model Across All Datasets

Table 10 provides a detailed description of the top 7 features identified, which exhibit similar performance to the original model.

Table 10: 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 Bytes/Bulk Avg | 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 Packet/Bulk Avg | 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. |

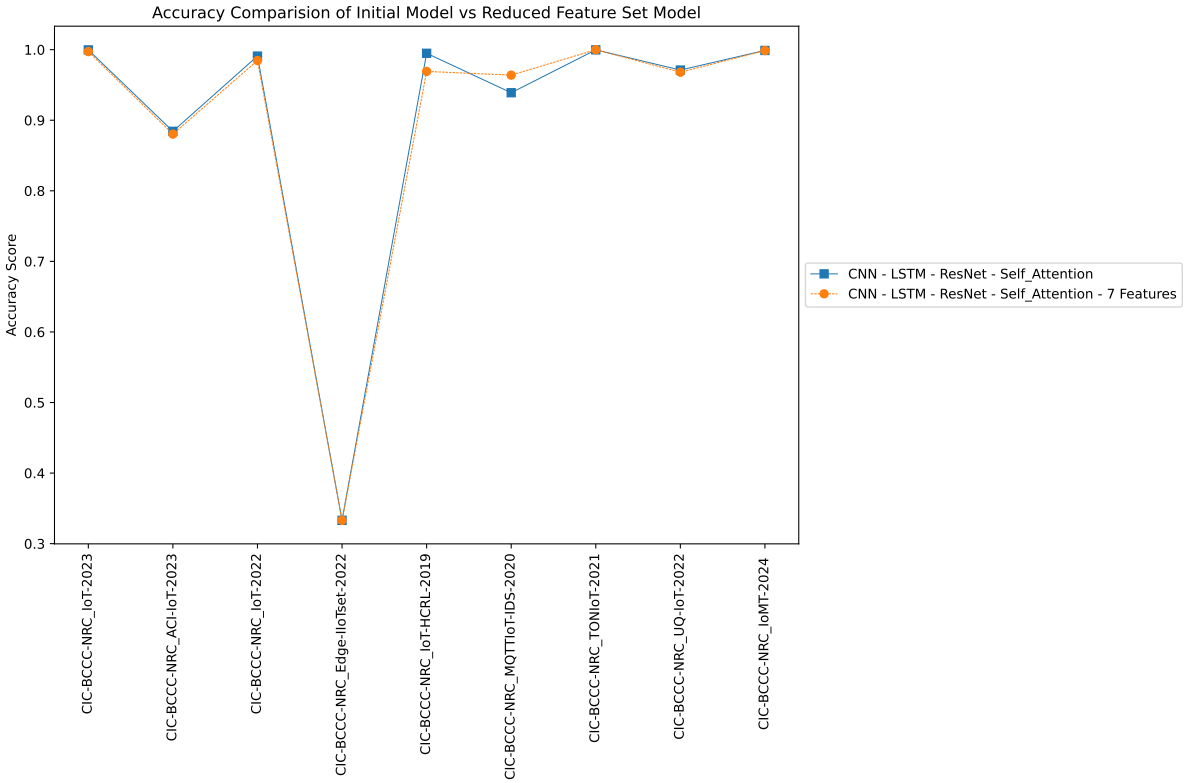



Figure 13: Accuracy Comparison between Initial Model vs Reduced Feature Set Model

| | **Total Parameters** | **Trainable Parameters** | **Non-trainable Parameters** | **Training Time (Sec)** |
|---|---|---|---|---|
| **Initial Model** | 251661 (983.05 KB) | 251661 (983.05 KB) | 0 (0.00 Byte) | 1198 s |
| **Reduced 7 Feature Model** | 136461 (533.05 KB) | 136461 (533.05 KB) | 0 (0.00 Byte) | 1067 s |

Table 11: Comparison of Initial and Reduced Feature Model Sizes and Training Time

Finally, the performance comparison between the initial model and the reduced 7-feature set model is presented in Figure 13. Additionally, Table 11 illustrates the model size and training time for both the initial model and the reduced 7-feature set model. It is observed that there is a 45.77% decrease in model size, accompanied by a slight reduction in training time.

Due to the significant imbalance present in all datasets, characterized by a higher number of attack instances compared to benign ones, and considering that these datasets were used as test datasets without balancing, it is likely that this imbalance contributed to the high recall and precision observed in our model, as evidenced by the confusion matrices in Table 7 and Table 9. Nevertheless, given that the training was conducted on the balanced CIC-BCCC-NRC_IoT-2023 dataset, we are confident in the model's capability 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.

In summary, the better performance of the proposed Self-attention-based 1D-CNN-LSTM approach for IoT attack detection can be attributed mostly to its robust feature extraction capabilities, efficient handling of high-dimensional data, and its capacity to highlight important data points via the self-attention mechanism. The integration of convolutional Neural Networks (CNNs) with Long Short-Term Memory (LSTM) networks allows the model to effectively capture spatial and temporal patterns in network data, hence facilitating accurate distinction between benign

and malicious activities. Applying Shapley Additive Explanations (SHAP) to the model optimizes its performance by emphasizing the most significant characteristics, therefore simplifying the model and increasing computing efficiency without compromising accuracy. Moreover, the model's performance consistency across different enhanced IoT datasets emphasizes its resilience and flexibility, establishing it as a reliable and flexible solution for IoT security and intrusion detection.

## 6. Conclusion & Future Works

In this paper, a novel method called the Self-attention-based 1D-CNN-LSTM, integrating CNNs, Long Short-Term Memory (LSTMs), and self-attention mechanisms, was developed to enhance the detection of IoT attacks within TCP network flows [62]. 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.

Throughout this research, significant emphasis has been placed on the efficiency of the data preprocessing stage. We employ the CICFlowmeter tool for precise feature extraction from network traffic data and implement 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 research tested the initial and reduced models on nine augmented IoT datasets called CIC-BCCC-NRC_TabularIoTAttack-2024. 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.

### *6.1. Future Works*

1. *Model Architecture Optimization*: Continuously refining the architecture of the Self-attention-based 1D-CNN-LSTM mechanism 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, multi-head, 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*: Developing 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.

### *6.2. Availability*

After publishing this paper, the augmented datasets, namely CIC-BCCC-NRC_TabularIoTAttack-2024, are available on the collaborator universities' websites.

*6.3. Acknowledgement*

This project was supported in part by collaborative research funding from the National Research Council of Canadas Artificial Intelligence for Logistics Program.

## AppendixA. List Of All Malicious Activities Present In All Augmented Datasets

This appendix provides a comprehensive list of all malicious activities identified in the datasets referenced in Table 4. Each activity is accompanied by a brief description to aid in understanding the various threats documented.

Table A.12: List Of All Malicious Activities Present In All Datasets in Table 4 [4] [63]

| **Malicious Activity** | **Description** |
|---|---|
| Benign Traffic | Normal network traffic with no malicious intent. |
| DDoS ACK Fragmentation | 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 Fragmentation | 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. |
| 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. |
| 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. |

| Malicious Activity | Description |
|---|---|
| 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. |
| 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. |
| 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. |
| 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. |

## AppendixB. CICFlowmeter Extracted Features List

This appendix presents a comprehensive list of the features extracted by the CICFlowmeter tool. Each feature is described briefly to help understand the various aspects of network traffic that were captured and analyzed.

Table B.13: 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 |
| 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 |
| 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 |
| 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) |

Table B.13 – continued from previous page

| **Feature Name** | **Description** |
|---|---|
| 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 |
| 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 |
| 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 |

Table B.13 – continued from previous page

| Feature Name | Description |
|---|---|
| Active Max | Maximum time a flow was active before becoming idle |
| 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 |


[1] L. Vu, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, E. Dutkiewicz, Deep transfer learning for iot attack detection, IEEE Access 8 (2020) 107335–107344. doi:10.1109/ACCESS.2020.3000476.

[2] Z. Wang, J. Li, S. Yang, X. Luo, D. Li, S. Mahmoodi, A lightweight iot intrusion detection model based on improved bert-of-theseus, Expert Systems with Applications 238 (2024) 122045. doi:https://doi.org/10.1016/j.eswa.2023.122045.
URL https://www.sciencedirect.com/science/article/pii/S0957417423025472

[3] Z. Wang, D. Liu, Y. Sun, X. Pang, P. Sun, F. Lin, J. C. S. Lui, K. Ren, A survey on iot-enabled home automation systems: Attacks and defenses, IEEE Communications Surveys & Tutorials 24 (4) (2022) 2292–2328. doi:10.1109/COMST.2022.3201557.

[4] T. Sasi, A. H. Lashkari, R. Lu, P. Xiong, S. Iqbal, A comprehensive survey on iot attacks: Taxonomy, detection mechanisms and challenges, Journal of Information and Intelligence (2023). doi:https://doi.org/10.1016/j.jiixd.2023.12.001.
URL https://www.sciencedirect.com/science/article/pii/S2949715923000793

[5] A. Simmons, Internet of things (iot) architecture: Layers explained, (Accessed on 06/04/2023) (Nov 2022).
URL https://dgtlinfra.com/internet-of-things-iot-architecture/

[6] J. Sengupta, S. Ruj, S. D. Bit, A comprehensive survey on attacks, security issues and blockchain solutions for iot and iiot, Journal of Network and Computer Applications 149 (2020) 102481. doi:https://doi.org/10.1016/j.jnca.2019.102481.
URL https://www.sciencedirect.com/science/article/pii/S1084804519303418

[7] B. F. C. W. P. F. H. R. R. D. Olawale Oladehin, Dan Griffin Catalin Vieru, Iot lens: Edge layer, (Accessed on 06/04/2023) (Mar 2023).
URL https://docs.aws.amazon.com/wellarchitected/latest/iot-lens/edge-layer.html

[8] S. Shock, The 5 layers of iot architecture that give it super powers, (Accessed on 06/04/2023) (Jan 2023).
URL https://www.techtarget.com/iotagenda/blog/IoT-Agenda/The-importance-of-the-edge-layer-in-an-IoT-ecosystem

[9] R. Agar, Iot architecture guide. major and additional layers of iot system, (Accessed on 06/04/2023) (Nov 2022).
URL https://www.helpwire.app/blog/iot-architecture/

[10] A. Arampatzis, Top 10 vulnerabilities that make iot devices insecure, (Accessed on 06/04/2023) (Oct 2022).
URL https://venafi.com/blog/top-10-vulnerabilities-make-iot-devices-insecure/

[11] L. Sun, Q. Du, A review of physical layer security techniques for internet of things: Challenges and solutions, Entropy 20 (10) (2018). doi:10.3390/e20100730.
URL https://www.mdpi.com/1099-4300/20/10/730

[12] M. M. Ogonji, G. Okeyo, J. M. Wafula, A survey on privacy and security of internet of things, Computer Science Review 38 (2020) 100312. doi:https://doi.org/10.1016/j.cosrev.2020.100312.
URL https://www.sciencedirect.com/science/article/pii/S1574013720304123

[13] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, S. C. de Alvarenga, A survey of intrusion detection in internet of things, Journal of Network and Computer Applications 84 (2017) 25–37. doi:https://doi.org/10.1016/j.jnca.2017.02.009.
URL https://www.sciencedirect.com/science/article/pii/S1084804517300802

[14] S. Hajiheidari, K. Wakil, M. Badri, N. J. Navimipour, Intrusion detection systems in the internet of things: A comprehensive investigation, Computer Networks 160 (2019) 165–191. doi:https://doi.org/10.1016/j.comnet.2019.05.014.
URL https://www.sciencedirect.com/science/article/pii/S1389128619306267

[15] D. E. Kouicem, A. Bouabdallah, H. Lakhlef, Internet of things security: A top-down survey, Computer Networks 141 (2018) 199–221. doi:https://doi.org/10.1016/j.comnet.2018.03.012.
URL https://www.sciencedirect.com/science/article/pii/S1389128618301208

[16] G. Fersi, Fog computing and internet of things in one building block: a survey and an overview of interacting technologies, Cluster Computing 24 (4) (2021) 2757–2787. doi:10.1007/s10586-021-03286-4.
URL https://doi.org/10.1007/s10586-021-03286-4

[17] N. K. Tran, M. Ali Babar, J. Boan, Integrating blockchain and internet of things systems: A systematic review on objectives and designs, Journal of Network and Computer Applications 173 (2021) 102844. doi:https://doi.org/10.1016/j.jnca.2020.102844.
URL https://www.sciencedirect.com/science/article/pii/S1084804520303118

[18] R. A. Memon, J. P. Li, J. Ahmed, M. I. Nazeer, M. Ismail, K. Ali, Cloud-based vs. blockchain-based iot: a comparative survey and way forward, Frontiers of Information Technology & Electronic Engineering 21 (4) (2020) 563–586. doi:10.1631/FITEE.1800343.
URL https://doi.org/10.1631/FITEE.1800343

[19] P. J. Taylor, T. Dargahi, A. Dehghantanha, R. M. Parizi, K.-K. R. Choo, A systematic literature review of blockchain cyber security, Digital Communications and Networks 6 (2) (2020) 147–156. doi:https://doi.org/10.1016/j.dcan.2019.01.005.
URL https://www.sciencedirect.com/science/article/pii/S2352864818301536

[20] R. R. Krishna, A. Priyadarshini, A. V. Jha, B. Appasani, A. Srinivasulu, N. Bizon, State-of-the-art review on iot threats and attacks: Taxonomy, challenges and solutions, Sustainability 13 (16) (2021). doi:10.3390/su13169463.
URL https://www.mdpi.com/2071-1050/13/16/9463

[21] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, J. Qin, A survey on application of machine learning for internet of things, International Journal of

Machine Learning and Cybernetics 9 (8) (2018) 1399–1417. doi:10.1007/s13042-018-0834-5.
URL https://doi.org/10.1007/s13042-018-0834-5
[22] L. Xiao, Y. Li, G. Han, G. Liu, W. Zhuang, Phy-layer spoofing detection with reinforcement learning in wireless networks, IEEE Transactions on Vehicular Technology 65 (12) (2016) 10037–10047. doi:10.1109/TVT.2016.2524258.
[23] S. M. Tahsien, H. Karimipour, P. Spachos, Machine learning based solutions for security of internet of things (iot): A survey, Journal of Network and Computer Applications 161 (2020) 102630. doi:https://doi.org/10.1016/j.jnca.2020.102630.
URL https://www.sciencedirect.com/science/article/pii/S1084804520301041
[24] A. Chatterjee, B. S. Ahmed, Iot anomaly detection methods and applications: A survey, Internet of Things 19 (2022) 100568. doi:https://doi.org/10.1016/j.iot.2022.100568.
URL https://www.sciencedirect.com/science/article/pii/S2542660522000622
[25] P. Maniriho, A. N. Mahmood, M. J. M. Chowdhury, A study on malicious software behaviour analysis and detection techniques: Taxonomy, current trends and challenges, Future Generation Computer Systems 130 (2022) 1–18. doi:https://doi.org/10.1016/j.future.2021.11.030.
URL https://www.sciencedirect.com/science/article/pii/S0167739X21004751
[26] M. F. Razali, M. N. Razali, F. Z. Mansor, G. Muruti, N. Jamil, Iot honeypot: A review from researcher's perspective, in: 2018 IEEE Conference on Application, Information and Network Security (AINS), 2018, pp. 93–98. doi:10.1109/AINS.2018.8631494.
[27] What is siem, (Accessed on 06/04/2023) (Oct 2023).
URL https://www.ibm.com/topics/siem
[28] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, A. A. Ghorbani, Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment, Sensors 23 (13) (2023). doi:10.3390/s23135941.
URL https://www.mdpi.com/1424-8220/23/13/5941
[29] I. A. N. Authority, Protocol numbers, (Accessed on 06/04/2023).
URL https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml
[30] A. H. Lashkari, Cicflowmeter (formerly iscxflowmeter), (Accessed on 06/04/2023) (Jan 2022).
URL https://www.unb.ca/cic/research/applications.html
[31] A. Abulkhair, Data imputation demystified time series data, (Accessed on 06/04/2023) (Jun 2023).
URL https://medium.com/@aaabulkhair/data-imputation-demystified-time-series-data-69bc9c798cb7
[32] B. singh, Handling missing data with knn imputer, (Accessed on 06/04/2023) (Aug 2023).
URL https://medium.com/@bhanupsingh484/handling-missing-data-with-knn-imputer-927d49b09015
[33] A. W. Services, What is hyperparameter tuning?, (Accessed on 06/04/2023) (Aug 2023).
URL https://aws.amazon.com/what-is/hyperparameter-tuning/
[34] scikit learn.org, train_test_split, (Accessed on 06/04/2023).
URL https://scikit-learn.org/stable/modules/generated/sklearn.model$_s$*election.train*$_t$*est*$_s$*plit.html*
[35] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, pp. 4768–4777.
[36] C. Molnar, Interpretable machine learning: A guide for making black box models explainable, (Accessed on 06/04/2023) (Aug 2023).
URL https://christophm.github.io/interpretable-ml-book/shap.htmlshap-feature-importance
[37] N. Bastian, D. Bierbrauer, M. McKenzie, E. Nack, Aci iot network traffic dataset 2023 (2023). doi:10.21227/qacj-3x32.
URL https://dx.doi.org/10.21227/qacj-3x32
[38] S. Dadkhah, H. Mahdikhani, P. K. Danso, A. Zohourian, K. A. Truong, A. A. Ghorbani, Towards the development of a realistic multidimensional iot profiling dataset, in: 2022 19th Annual International Conference on Privacy, Security Trust (PST), 2022, pp. 1–11. doi:10.1109/PST55820.2022.9851966.
[39] S. Dadkhah, E. C. P. Neto, R. Ferreira, R. C. Molokwu, S. Sadeghi, A. Ghorbani, Ciciomt2024: Attack vectors in healthcare devices-a multi-protocol dataset for assessing iomt device security, Preprints (February 2024). doi:10.20944/preprints202402.0898.v1.
URL https://doi.org/10.20944/preprints202402.0898.v1
[40] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, H. Janicke, Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications: Centralized and federated learning (2022). doi:10.21227/mbc1-1h68.
URL https://dx.doi.org/10.21227/mbc1-1h68
[41] H. Kang, D. H. Ahn, G. M. Lee, J. D. Yoo, K. H. Park, H. K. Kim, Iot network intrusion dataset (2019). doi:10.21227/q70p-q449.
URL https://dx.doi.org/10.21227/q70p-q449
[42] H. Hindy, C. Tachtatzis, R. Atkinson, E. Bayne, X. Bellekens, Mqtt-iot-ids2020: Mqtt internet of things intrusion detection dataset (2020). doi:10.21227/bhxy-ep04.
URL https://dx.doi.org/10.21227/bhxy-ep04
[43] K. D. Z. Z. G. M. L. U. He, Ke, J. Yu, Uq iot ids dataset 2021 (2021). doi:https://doi.org/10.48610/17b44bb.
URL https://espace.library.uq.edu.au/view/UQ:17b44bb
[44] Radware, Fragmented ack attack, (Accessed on 15/04/2024).
URL https://www.radware.com/security/ddos-knowledge-center/ddospedia/fragmented-ack-attack/
[45] Akamai, What is an icmp flood ddos attack?, (Accessed on 15/04/2024).
URL https://www.akamai.com/glossary/what-is-icmp-flood-ddos-attack
[46] Netscout, Ip/icmp fragmentation attacks.
URL https://www.netscout.com/what-is-ddos/ip-icmp-fragmentation
[47] D. Guard, Ack and push ack flood, (Accessed on 15/04/2024).
URL https://ddos-guard.net/en/terms/ddos-attack-types/ack-push-ack-flood
[48] D. Guard, Rst and fin flood, (Accessed on 15/04/2024).
URL https://ddos-guard.net/en/terms/ddos-attack-types/rst-fin-flood
[49] Cloudflare, Syn flood attack, (Accessed on 15/04/2024).

URL https://www.cloudflare.com/learning/ddos/syn-flood-ddos-attack/
[50] D. Guard, Synonymous ip attack, (Accessed on 15/04/2024).
URL https://ddos-guard.net/en/terms/ddos-attack-types/rst-fin-flood
[51] Imperva, Tcp syn flood, (Accessed on 15/04/2024).
URL https://www.imperva.com/learn/ddos/syn-flood/
[52] Imperva, Udp flood, (Accessed on 15/04/2024).
URL https://www.imperva.com/learn/ddos/udp-flood/
[53] D. Guard, Fragmented udp flood, (Accessed on 15/04/2024).
URL https://ddos-guard.net/en/terms/ddos-attack-types/udp-fragmentation-flood
[54] P. Security, What is dns spoofing? + 5 tips to prevent it, (Accessed on 15/04/2024) (Mar 2024).
URL https://www.pandasecurity.com/en/mediacenter/dns-spoofing/
[55] Imperva, Http flood, (Accessed on 15/04/2024).
URL https://www.imperva.com/learn/ddos/http-flood/
[56] Radware, What is the mirai botnet ?, (Accessed on 15/04/2024).
URL https://www.radware.com/security/ddos-knowledge-center/ddospedia/mirai/
[57] R. Winward, Iot attack handbook, (Accessed on 15/04/2024).
URL https://www.radware.com/getattachment/402db7f3-0467-4fa3-bb9a-ae88b728e91b/MiraiHandbookEbookFinal$_0$4.*pdf.aspx*
[58] Imperva, Arp spoofing, (Accessed on 15/04/2024).
URL https://www.imperva.com/learn/application-security/arp-spoofing/
[59] A. Bhandari, Feature scaling: Engineering, normalization, and standardization, (Accessed on 07/12/2023) (Jun 2024).
URL https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/
[60] H. Wang, Q. Liang, J. T. Hancock, T. M. Khoshgoftaar, Feature selection strategies: a comparative analysis of shap-value and importance-based methods, Journal of Big Data 11 (1) (2024) 44. doi:10.1186/s40537-024-00905-w.
URL https://doi.org/10.1186/s40537-024-00905-w
[61] K. Aas, M. Jullum, A. Løland, Explaining individual predictions when features are dependent: More accurate approximations to shapley values, Artificial Intelligence 298 (2021) 103502. doi:https://doi.org/10.1016/j.artint.2021.103502.
URL https://www.sciencedirect.com/science/article/pii/S0004370221000539
[62] S. Yaras, M. Dener, Iot-based intrusion detection system using new hybrid deep learning algorithm, Electronics 13 (6) (2024). doi:10.3390/electronics13061053.
URL https://www.mdpi.com/2079-9292/13/6/1053
[63] P. Victor, A. H. Lashkari, R. Lu, T. Sasi, P. Xiong, S. Iqbal, Iot malware: An attribute-based taxonomy, detection mechanisms and challenges, Peer-to-Peer Networking and Applications 16 (3) (2023) 1380–1431. doi:10.1007/s12083-023-01478-w.
URL https://doi.org/10.1007/s12083-023-01478-w

**Declaration of interests**

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: