UNIVERSITY OF RWANDA
AFRICAN CENTRE OF EXCELLENCE IN INTERNET OF THINGS
KIGALI-RWANDA

A SELF-LEARNING LIGHTWEIGHT INTRUSION DETECTION SYSTEM FOR INTERNET OF THINGS

PhD Thesis submitted in the fulfillment of award of the Degree

of

Doctor of Philosophy in Internet of Things - Embedded Computing Systems

By

Promise Ricardo AGBEDANU

221030392

UNIVERSITY OF RWANDA
AFRICAN CENTRE OF EXCELLENCE IN INTERNET OF THINGS

A SELF-LEARNING LIGHTWEIGHT INTRUSION DETECTION SYSTEM FOR INTERNET OF THINGS

PhD THESIS

PhD Thesis submitted in the fulfillment of award of the Degree

of

Doctor of Philosophy in Internet of Things - Embedded Computing Systems

By
Promise Ricardo AGBEDANU
(221030392)

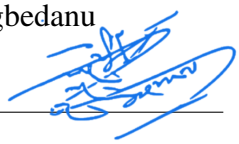
Supervisor: Prof Richard MUSABE
Co-Supervisor: Dr. James RWIGEMA
Co-Supervisor: Prof Ignace GATARE
Co-Supervisor: Dr. Shanchieh Jay Yang

September 2025

Declaration

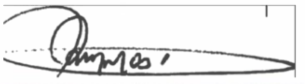
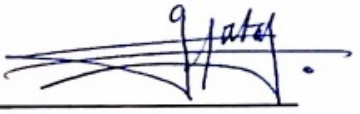
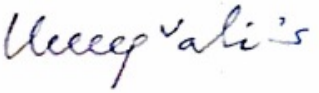
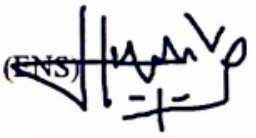
I hereby declare that the dissertation entitled ***"A self-learning lightweight intrusion detection system for internet of things"*** to be submitted for the Degree of Doctor of Philosophy is my original work and the dissertation has not formed the basis for the award of any degree, diploma, associateship, or fellowship of similar titles. It has not been submitted to any other University or Institution for the award of any degree or diploma.

Promise R. Agbedanu

Signature: 

Date: 3rd September 2025

Promise R. Agbedanu, a PhD student of UR-ACEIoT registration ID **221030392**, successfully defended the PhD thesis/dissertation entitled **"A Self-Learning Lightweight Intrusion Detection System for Internet of Things"**, which he prepared after fulfilling the requirements specified in the associated legislation, before the thesis examination members whose signatures are below.

Thesis Supervisor:	**Prof. Richard Musabe** University of Rwanda	
Co-Supervisor:	**Dr. James Rwigema** University of Rwanda	
Resident Co-Supervisor:	**Prof. Ignace Gatare** University of Rwanda	
Viva Voce Members:	**Prof. Wali U. Garba (Chair)** University of Rwanda	
	Prof. Baraka Maiseli University of Dar es Salaam	
	Prof. Vincent Hayarimana Burundi Higher Institute of Education (~~ENS~~)	
	Dr. Mwitende Gervais Rwanda Polytechnic	
Date of Defense:	**03/09/2025**	

Dedication

First of all, I would like to express my sincere gratitude to Yahweh, the creator of the universe and all that is in it, from the depths of the wells of my heart. I am forever grateful. This thesis is dedicated to my wife (Mpho Mary Agbedanu), my son (Jefferey Awuku Agbedanu), my dad (Emmanuel Agbedanu), my mum (Joyce Biose), my uncle (William Agbedanu), my aunt (Dzigbordzi Tegbe-Agbedanu), and the entire Agbedanu family.

Acknowledgement

To begin with, I would like to extend my heartfelt gratitude to my advisors, Prof Richard Musabe, Prof Shanchieh Jay Yang (Gonzaga University), Prof Ignace Gatare, and Dr James Rwigema. To Prof Damien Hanyurwimfura, Dr Omar Gatera, and all the African Centre of Excellence in Internet of Things (ACEIoT) staff, I am forever grateful for your continued support and encouragement throughout my PhD journey. I also want to express my appreciation to all the Regional Scholarship and Innovation Fund (RSIF) regional coordination unit staff, PASET-RSIF, Google PhD Fellowship, and the Carnegie Foundation of New York. To my Google PhD Fellowship mentor, Yanis Pavlidis, I am most grateful for your encouragement and guidance. To the entire staff of RIT Global of the Rochester Institute of Technology (RIT), especially Lyndsey McGrath, I am grateful for all the support you provided when I was in the United States. You made me feel at home. I also want to say a big thank you to Megan Kless, the director of the International Student Services of RIT. To my colleagues, Destiny Kwabla Amenyedzi, Dr. Theofirda Julius Maginga, Francisco Pinto, Chiedza Hwata, and Micheline Kazeneza, I am very grateful for being there for me in both the good and the bad times. I also want to thank Dr. Nana Adwoa Nkuma Johnson and Dr. Farian Severine Ishengoma for proofreading this work. A special thank you also goes to the following: Prof Benjamin Asubam Weyori (Department of Computer and Electrical Engineering, University of Energy and Natural Resources, Ghana), Mr Nicodemus Awarayi (Department of Computer Science, University of Energy and Natural Resources, Ghana), Pastor Seyram Agbenyegah, Mr Christian Djin Fiagbedzi, and Mr Kwashie Efui Kpobi

Preface

This thesis is a summary of my work on developing ***a self-learning lightweight intrusion detection system of Internet of Things.*** This work was carried out at the African Centre of Excellence in Internet of Things, College of Science and Technology, University of Rwanda. The first part of this thesis consists of two chapters. The first chapter focuses on the introduction, and the second chapter looks at related works. The second part is the actual research work done, which comprises the following:

Chapter 4
A Real-Time Deep Learning IDS for IoT with Dynamic Quantization and Adaptive Online Learning

Chapter 5
ALMANET: A Hybrid Online Learning IDS for Real-Time IoT Security

Chapter 6
Adaptive Multi-Label Intrusion Detection in IoT Networks Using ALMA with Stochastic Weight Averaging

Chapter 7
A Scalable Approach to IoT and IIoT Security: Evaluating Adaptive SAMKNN for Zero-Day Attack Detection

Chapter 8
An Online Adaptive Approach to Detecting Zero-day Attacks in IoT and IIoT Systems

Chapter 9
IPCA-SAMKNN: A Novel Network IDS for Resource-Constrained Devices

Chapter 10
Using Incremental Ensemble Learning Techniques to Design Portable Intrusion Detection for Computationally Constrained Systems

Abstract

The past decade has seen an increase in the adoption of the Internet of Things (IoT) ecosystem. The number of IoT devices is estimated to be around 30.9 billion by the end of 2025, as reported by Statista. This number is almost four times the current population of the world. The massive adoption of the IoT system in domains such as healthcare, energy, transportation, home, and industry, coupled with the heterogeneity, computational constraints, and the insecure nature of some of these IoT devices, has made it attractive to cyber-attacks. Over the years, various techniques have been explored either to mitigate or reduce the number of attacks against the IoT system. Some of these techniques are encryption, access control, secure architecture, and intrusion detection systems (IDS). IDSs have proven very effective in detecting attacks in traditional computing systems. Over the past ten years, a lot of work has been done focusing on the use of IDS to detect attacks within the IoT ecosystem. However, IDSs for traditional computing systems do not meet the requirements of the IoT ecosystem due to the heterogeneous nature of the IoT ecosystem, the protocol-specific nature of the IoT ecosystem, the dynamic nature of the IoT ecosystem, and the limited computational capacity of IoT devices. This has led to the IoT security research community working to develop IoT-specific IDS that can overcome the earlier challenges mentioned. Most of the works done in the quest to design these IoT-specific IDSs use machine learning (ML) based techniques, with the majority of these approaches using offline ML techniques. Using offline ML algorithms to design IDS for the IoT ecosystem leads to problems such as the IDS becoming obsolete when there is a change in the data that was used to train the model, computational complexity, and inability to adapt to real-time network traffic. Online ML algorithms have shown the ability to produce lightweight models, adapt to real-time environments, and handle drifts in domains such as recommended systems.

In this thesis, we designed a lightweight IDS using online ML techniques that can run on the edge of an IoT network, such as a gateway device. In addition, the proposed IDS should be able to adapt to changes in network traffic in real-time.

The study proposed an IoT-based IDS using an online ML algorithm to build an IDS that is lightweight and self-learning by dynamically adapting to changes in traffic.

The study is divided into five stages. The first stage involved investigating various ML algorithms to identify which techniques are most suitable for developing a self-learning IDS for the IoT ecosystem. Our investigation revealed that online ML algorithms can be used to design IDS models that are lightweight in nature and can adapt to network traffic in real time without having to retrain the model. To validate this, we used an ensemble of Gaussian Naïve Bayes and Hoeffding Tree to design an online ensemble model. The results show that the proposed IDS recorded an average accuracy of 99.98% with a memory usage between 122.38 KB and 650.11 KB.

During the second stage, we focused on using a lightweight data preprocessing technique to reduce the memory and computational requirements of the proposed intrusion detection. We used an incremental principal component analysis (IPCA) as the data preprocessing technique and used the Self-Adjusting Memory k-Nearest Neighbour (SAMKNN) to model our IDS. We used an on-device (Raspberry Pi Model B) training approach to build the proposed IDS. The results show that the proposed model could record an accuracy as high as 98.91%, using a memory of 1.4% of the total memory allocated on the device, 1.6% of the CPU, and an average energy usage of 2%.

In the third stage, we developed an adaptive version of the SAMKNN algorithm that dynamically updates and reacts to drifts. Our proposed Adaptive SAMKNN dynamically adjusts its memory allocation. The approach not only allows the IDS to detect real-time threats but also uses minimal memory. The results show that the proposed IDS outperforms the non-adaptive version of SAMKNN in terms of memory efficiency.

In the fourth stage of the study, we explored the ability of the proposed IDS to detect zero-day attacks deployed by adversaries as adversarial attacks. The results show that the IDS built using adaptive SAMKNN can detect synthetic day attacks injected into the various datasets used for the experimental validation.

Finally, we developed an online deep learning based IDS with dynamic quantization. The results show that the proposed system achieved high performance and used minimal computational resources after quantization.

Key words: Intrusion Detection System, Internet of Things, Industrial Internet of Things, Online Machine Learning, Zero-day attack, Online Deep Learning, Adversarial Machine Learning, Generative Adversarial Network, Dynamic Quantization.

Publications

Chapter 4 is based on the following article:
P. R. Agbedanu, R. Musabe, I. Gatare, J. Rwigema, "A Real-Time Deep Learning IDS for IoT with Dynamic Quantization and Adaptive Online Learning," Submitted for publication (under review).

Chapter 5 is is published as:
Promise Ricardo Agbedanu, Shanchieh (Jay) Yang, Richard Musabe, Ignace Gatare, James Rwigema, "ALMANET: A hybrid online learning IDS for real-time IoT security," Egyptian Informatics Journal, Volume 31, 2025, 100764, ISSN 1110-8665, https://doi.org/10.1016/j.eij.2025.100764.

Chapter 6 is based on the following article:
P. R. Agbedanu, S. J. Yang, R. Musabe, I. Gatare, J. Rwigema, "Adaptive Multi-Label Intrusion Detection in IoT Networks Using ALMA with Stochastic Weight Averaging," Submitted for publication (under review).

Chapter 7 is published as:
P. R. Agbedanu, S. J. Yang, R. Musabe, I. Gatare, J. Rwigema, "A Scalable Approach to IoT and IIoT Security: Evaluating Adaptive SAMKNN for Zero-Day Attack Detection," *Sensors*, 25(1), 216.

Chapter 8 is published as:
P. R. Agbedanu, S. Jay Yang, R. Musabe, I. Gatare and J. Rwigema, "An Online Adaptive Approach to Detecting Zero-day Attacks in IoT and IIoT Systems," *2024 IEEE Globecom Workshops (GC Wkshps)*, Cape Town, South Africa, 2024, pp. 1-7, doi: 10.1109/GCWkshp64532.2024.11101063.

Chapter 9 is published as:
P. R. Agbedanu, R. Musabe, J. Rwigema, I. Gatare, Y. Pavlidis, "IPCA-SAMKNN: A novel network IDS for resource-constrained devices," In *2022 2nd International Seminar on Machine Learning, Optimization, and Data Science (ISMODE)* (pp. 540-545). IEEE.

Chapter 10 is published as
P. R. Agbedanu, R. Musabe, J. Rwigema, I. Gatare, Using Incremental Ensemble Learning Techniques to Design Portable Intrusion Detection for Computationally Constrained Systems. *International Journal of Advanced Computer Science and Applications,* vol. 13, no. 11, pp. 33-45, 2022.

List of Figures

List of Tables

Acronyms

6LoWPAN - IPv6 Over Low-power Wireless Personal Area Networks
AI - Artificial Intelligence
ALMA - Approximate Large Margin Algorithm
BLE - Bluetooth Low Energy
CNN - Convolutional Neural Network
CoAP - Constrained Application Protocol
DDoS - Distributed Denial of Service
DNN - Deep Neural Network
DoS - Denial of Service
GAN - Generative Adversarial Networks
HIDS - Host-based Intrusion Detection System
IDS - Intrusion Detection System
IIoT - Industrial Internet of Things
IPCA - Incremental Principal Component Analysis
IoT - Internet of Things
ML - Machine Learning
MQTT - Message Queuing Telemetry Transport
NIDS - Network Intrusion Detection System
RNN - Recurrent Neural Network
ROCAUC - Receiving Operating Characteristic Area Under the Curve
SAMKNN - Self Adjusting Memory k-Nearest Neighbour
SVM - Support Vector Machine
SWA - Stochastic Weight Averaging

Table of Contents

Chapter 1

1. Introduction

1.1 Motivation

Over the years, technology has evolved from a stand-alone computer to the interconnection of these computers to form the internet. As the evolution continues, we are in the Internet of Things (IoT) era. This new paradigm has become a game-changer in the technological world. IoT has been applied in various domains such as healthcare, manufacturing industries, transportation, and home automation. This paradigm has provided significant benefits to these and other domains and continues to expand its impact.

The integration of real-world objects with the internet brings numerous advantages and cyber-security threats to our lives through our interaction with these devices [1]. It is, therefore, not surprising that the security of IoT, ranging from the physical security of the devices to the security of their architecture, has become a key area of research for many researchers.

Currently, several works are being done to ensure data confidentiality, access control, authentication, privacy, and trust in IoT environments. Although a lot of progress has been made in the security of IoT using the parameters mentioned above, attackers still find ways to exploit the vulnerabilities that exist in IoT systems. A good mitigation strategy is to detect these vulnerabilities during exploitation. This is where the Intrusion Detection System (IDS) comes into play. IDS in IoT is a research area that still has many unexplored areas [1]. Although much work has been done in traditional networks with reference to IDS, these solutions cannot be applied to IoT systems. According to [2], there are limitations when applying traditional security solutions in IoT environments. These limitations are the limited computing power of IoT devices, the vast number of devices, and data sharing between objects and users. It is, therefore, expedient to develop a lightweight IDS that can be supported by the computational resources provided by IoT systems. Moreover, most IDS proposed for IoT systems are passive in nature. They only raise alerts or can detect intrusion, but rely mostly on humans to determine what actions to take after detecting the intrusion.

As the evolution of computing technology continues, the ability of things such as fridges, air-conditioners, medical equipment, and meters to communicate has become a reality due to fast communication technologies. A paradigm popularly known as IoT has become a household term with smart homes and has numerous uses in energy, agriculture, manufacturing, healthcare, and transportation. There is no doubt that the IoT has many benefits, which is why the number of IoT devices is growing at an exponential rate. The number of IoT devices is estimated to reach 30.9 billion by 2025, according to [3]. The numerous benefits of the IoT ecosystem make it

attractive to cyber-attacks. An attack statistic presented by SAM Seamless Network shows that over 1 billion IoT-based attacks happened in 2021 [4]. Although methodologies such as encryption and secure architecture are progressively being deployed to ensure that IoT devices are secured, the computational constraint of these devices makes it difficult to implement these security measures to their fullest potential. Another approach to securing these devices from cyber-attacks is to detect these attacks before an attacker exploits them. Intrusion Detection Systems have been around for more than four decades, with the development of these IDSs focused on traditional computing systems [5]. They have been among the primary methodologies used to protect computer networks. [6] defines intrusion detection as detecting activities perpetrated against computer systems by intruders.

Over the past forty years, a lot of breakthroughs have been made in the area of intrusion detection. One of the biggest breakthroughs in this area is using ML to detect intrusions. However, with all these breakthroughs, deploying traditional computing-based IDS methods in IoT is challenging. This situation has been created because of the computational constraints posed by IoT devices. This has led to several studies being carried out to design IDSs that can be deployed in IoT systems without significantly affecting the computational resources of these devices. Several approaches have been proposed in designing lightweight IDSs for IoT environments, but these studies fail to either report how these lightweight IDSs are achieved or how much computational resources these proposed approaches consume. For example, [7]–[11] proposed various ML-based techniques that are supposed to translate into lightweight IDSs, but these works either failed to report how these techniques translated into lightweight IDS or how much computational resources these proposed methods consume.

Most traditional IDSs are built for traditional networks, where these IDSs are not constrained by processing power, memory, and energy. These IDSs often depend on batch learning methods and complex anomaly detection algorithms that need access to high computing hardware. In the world of IoT, these requirements are not feasible because IoT devices usually have a limited CPU, small memory, and restricted battery life. Additionally, they often communicate over low-bandwidth and low-energy protocols. Running traditional IDS in such environments can cause slowdowns, increase false alarms, drain battery life, and fail to deliver timely protection. Above these shortfalls, most conventional IDSs are not designed to handle concept drift, the natural evolution of attack methods and normal network behavior, without undergoing frequent, resource-intensive retraining.

Additionally, there is a range of communication protocols in the IoT ecosystem, specifically designed to accommodate the resource-constrained nature of the devices in this ecosystem [12]. Comparing the IoT network to traditional computer networks, where as the traditional computer networks use protocols such as TCP/IP and HTTP, IoT systems often use specialized protocols such as message queuing telemetry transport (MQTT), constrained application protocol (CoAP), IPv6 over Low-power Wireless Personal Area Networks (6LoWPAN), Zigbee, LoRaWAN, and Bluetooth Low Energy (BLE). These protocols are designed to provide low overhead and stable connections, conditions that are crucial for low-powered devices. However, the efficiency produced by these protocols comes at a cost of threat detection since existing IDS traditionally can not recognize these IoT protocols [13], [14]. This leads to a situation where malicious activity that operates at the level of these protocols might go undetected. The lack of building IDS solutions that

are not peculiar IoT protocols has been a challenge faced by researchers as far as 2012 [15]. With emphasis on how traditional IDS solutions do not focus on the behaviors of these IoT protocols [15]. IoT network traffic differs not only in terms of packet size but also in the distinct patterns of this traffic, a situation that makes it difficult for traditional IDS to perform deep packet inspections on this traffic.

Designing a lightweight IDS aims to address the above-mentioned problems by combining designs that are efficient in resource usage and can learn and adapt on the fly. Lightweight IDSs are built to use minimal computational resources while still achieving high detection performance, making them ideal for low-power devices such as IoT systems, Raspberry Pi boards, and microcontroller-based gateways. Unlike traditional IDSs that are computationally expensive, lightweight IDSs that can learn from new data instances on the fly can update their parameters incrementally, avoiding the cost of downtime of retraining from scratch. Using this approach makes these IDSs faster to adapt to emerging threats and more feasible for IoT networks.

The IoT ecosystem has limited resources, yet it needs reliable and real-time protection. Lightweight IDS provides a realistic, scalable, and adaptive answer that bridges the gap between high-performance IDS and the practical needs of IoT devices.

This thesis focuses on creating and testing a self-learning, lightweight IDS designed specifically for IoT and edge computing environments. The system is built to keep up with changing cyber threats and adapt to new attack patterns in real time, even when resources are limited. It combines adaptive online learning methods with efficient data processing and optimization techniques, such as dynamic quantization, to deliver high detection accuracy while keeping computing and memory use low. This makes it practical for deployment on small, low-power devices. The approach is tested on a variety of real-world IoT and industrial IoT datasets, as well as on actual embedded devices, showing that it is not only effective and adaptable but also scalable and ready for real-world use in securing today's IoT networks.

1.2 Problem Statement

Detecting intrusions in the IoT comes with challenges because of the computational limitations, specific protocols, and standards of IoT devices. Additionally, IoT environments are characterized by dynamic and real-time data. Due to the above-mentioned challenges, deploying traditional IDS in IoT environments is difficult, and even in situations where traditional IDS is deployed in IoT environments, it will be ineffective. Because of these challenges, it is important to develop IDSs for the IoT ecosystem designed using low-resource utilization techniques. These IDSs should be capable of dynamically adapting to changes in the IoT ecosystem, learning and adapting to new intrusion patterns.

1.2.1 Research Questions

The research seeks to answer the following research questions.

1. Given the dynamic and heterogeneous nature of IoT environments, which machine learning algorithms are most applicable for achieving effective intrusion detection?

2. How can intrusion detection models be designed to detect intrusions in real-time, adapt to evolving attack patterns, and adapt to concept drift in IoT network traffic?

3. What strategies can be applied to reduce computational and memory overhead in IDS models, enabling their deployment on resource-constrained IoT and edge devices without significantly compromising detection accuracy?

4. To what extent do the proposed IDS approaches maintain effectiveness when validated on real-world IoT datasets and deployed in embedded platform scenarios?

1.2.2 Overall Objective

The main objective of this study is to advance the progress made in designing a self-learning and lightweight IDS for the IoT ecosystem. The study will focus on proposing a self-learning lightweight IDS for IoT systems that can dynamically respond to changes when faced with new attacks or unseen threats. The specific objectives of this study are as follows;

1.2.2.1 Specific Objectives

1. Investigate the applicability of ML algorithms for intrusion detection in IoT environments.

2. Develop and evaluate models capable of online, incremental learning to adapt to evolving attack patterns and concept drift.

3. Reduce computational and memory overhead, making it possible for the proposed system to be deployed on resource-constrained IoT and edge devices.

4. Validate the proposed approaches through real-world datasets and deployment scenarios on embedded platforms.

1.3 Contributions of the study

Although the initial part of this study investigated existing ML algorithms, the novelty of the study does not lie in the selection of the best-performing algorithm. The novelty lies in developing new adaptive and resource-efficient variants of online algorithms, such as Adaptive SAMKNN, hybrid ensemble algorithms, and online deep learning, designed for the computational and energy constraints of IoT and the edge environments. The models incorporate dynamic memory management, updates based on prediction confidence, and incremental feature reduction. These modifications enable the models to adapt to evolving attack patterns and concept drift in real time without retraining them from scratch. These enhancements are not in the baseline algorithms and address challenges like the detection of zero-day attacks.

Another contribution is the integration of lightweight design approaches, combining the adaptiveness of these algorithms with optimization techniques such as dynamic quantization for deep models and on-device incremental training. The above approach ensures that the proposed IDS not only has a high detection rate but is also deployable on low-power devices like Raspberry Pi. The

deployment experimental validation shows the feasibility of the proposed IDS in real-world IoT environments, bridging the gap between the theoretical performance of the proposed IDS and the practical security of resource-constrained devices.

1.4 Thesis Overview

The aim of this thesis is to design a self-learning, lightweight intrusion detection system for the IoT ecosystem.

The first part of this thesis is made up of two chapters. Chapter 1 focuses on the motivation of this study, the problem statement, research questions, and research objectives.

Chapter 2 is a comprehensive review of the literature related to this work.

Chapter 3 presents a general methodology for this thesis.

The second part of the thesis comprises the works we have done that are under review and those that have been published.

Chapter 4 of the study presents an online deep learning algorithm with dynamic quantization and confidence-aware learning control, which is designed to detect intrusions within the IoT ecosystem in real-time.

Chapter 5 is a study that looks at the use of a hybrid online learning based IDS for IoT systems. The work focuses on designing a hybrid IDS framework that combines an Approximate Large Margin Algorithm (ALMA) with Stochastic Weight Averaging (SWA) and a simple online neural network to detect intrusions within the IoT ecosystem.

Chapter 6 is an adaptive multi-label IDS for IoT systems. In this work, we explored the ability of ALMA with SWA to detect simultaneous attacks. Most existing IDS for IoT systems rely on either detecting attacks that appear in a binary class or a multi-class. We pushed the limit further to explore if ALMA with SWA can detect multi-label attacks and how the model behaves when certain attack classes are introduced to it before others.

Furthermore, in Chapter 7, we developed a scalable approach to detect zero-day attacks within the IoT and industrial internet of things (IIoT) environment. We design a novel algorithm called an adaptive self-adjusting memory k-nearest neighbor that is able to handle drifts and detect zero-day attacks.

In Chapter 8, we used an online adaptive approach to detect zero-day attacks. In addition to the efficiency and the detection rate of this approach, the proposed technique demonstrates the feasibility of being used in resource-constrained devices due to the lightweight nature of the model.

Chapter 9 explores the use of a feature reduction technique and its impact on building a lightweight IDS for IoT systems. The approach measured parameters such as memory and energy usage.

Finally, in Chapter 10, we used an online ensemble technique to design an IDS suitable for computationally constrained systems. We used Gaussian Naïve Bayes and Hoeffding tree to build an IDS that is not just efficient in terms of detection accuracy but also computationally inexpensive.

A conceptual flow diagram of the thesis is shown in Figure 1.1.

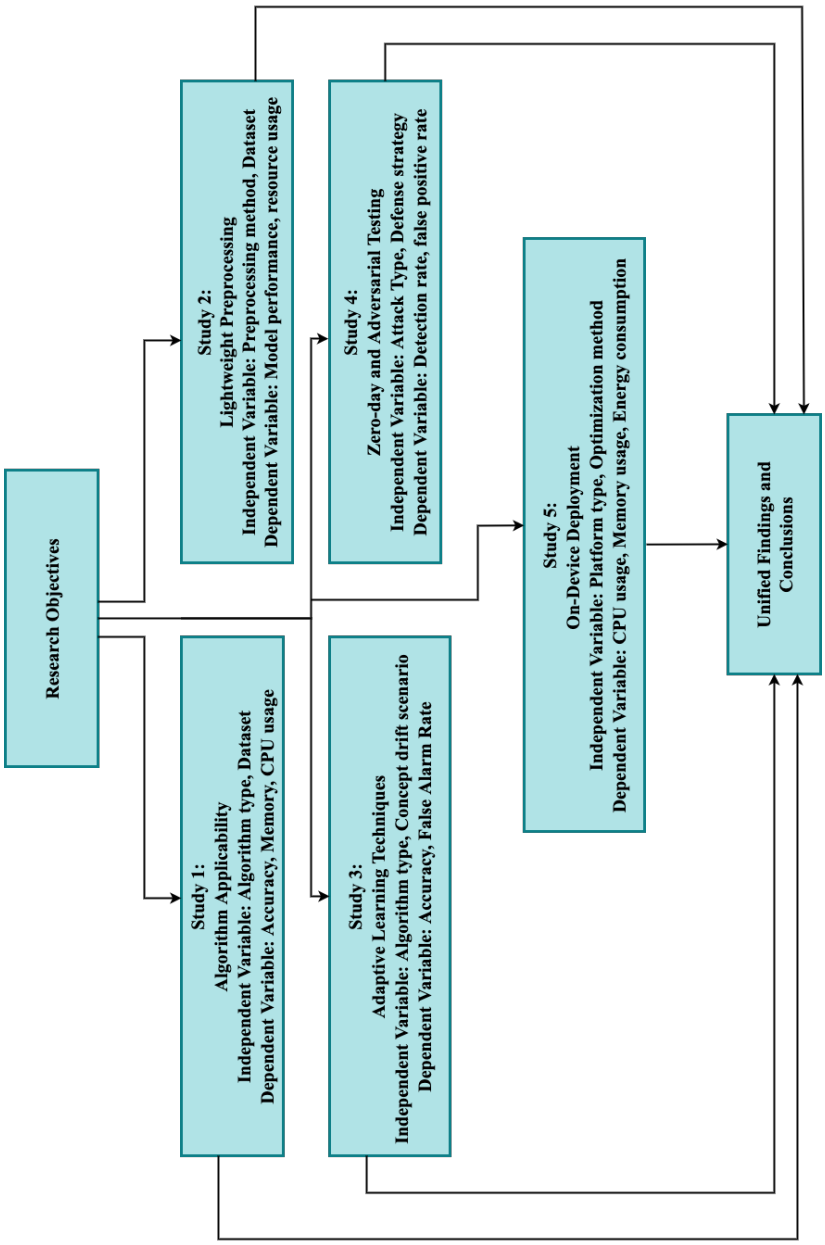


Figure 1.1: A conceptual flow diagram of the thesis

Chapter 2

2. Literature Review

2.1 Overview of IDS

There are various security solutions that have been developed over the years to identify malicious activities within a computer network. One of these security solutions is an IDS. According to [16], any unauthorized activity that aims to threaten the confidentiality, integrity, or availability of a computing system is an intrusion. To protect computing systems from these adversarial activities, IDSs in the form of software or hardware are implemented to monitor these computing systems and generate alerts when the activities deviate from normal activities.

The work of an IDS is to continuously monitor inputs such as network packets, system logs, or the behavior of applications and compare them against a model that can differentiate malicious actions from normal ones. Early detections of intrusions by IDSs can help reduce the damage these adversarial activities may cause to computing systems. Some modern IDS solutions are often integrated into broader security frameworks that have the ability not only to detect these malicious activities but also prevent them [16]–[18]

The last ten years have seen a continuous evolution in IDS technology and the adoption of this technology to address new challenges in emerging technologies such as cloud computing, industrial control systems, and the Internet of Things (IoT). According to [19], the number of connected IoT devices will reach over 75 billion by 2025. Additionally, the growing sophistication of cyber attacks and the role IDSs have played in detecting most of these attacks show that IDSs are essential tools for protecting computing networks [20]. Most modern IDSs are designed to adapt to high-volume data streams, work on resource-constrained devices, and detect novel attack patterns using advanced ML techniques.

2.2 Types of IDS

The classification of IDSs can be done using multiple categories. One of these categories is using the detection technique and the scope of monitoring. In this section, we will review the two major types of IDS, namely, signature-based versus anomaly-based detection, and host-based versus network-based deployment. We will also consider the hybrid IDS.

2.2.1 Signature-Based vs. Anomaly-Based IDS

A signature-based IDS uses a database of known attack patterns and flags any activity that matches a stored signature. This type of IDS operates like antivirus software. It relies on matching a pattern to identify previously seen attacks [16]. Open-source network IDS tools like Snort and Suricata use this approach, scanning network traffic for signatures of malware or intrusion behaviors. The major limitation of signature-based IDS is its inability to detect attacks that the IDS has not previously seen. This type of IDS can not detect a polymorphic variation of malware or a zero-day attack. Because cyberspace keeps evolving, deploying a signature-based IDS is less effective in detecting new attacks [21].

On the other hand, an anomaly-based IDS builds a model of the normal behavior of a computing system and flags a profile or usage that significantly deviates from the normal usage as an intrusion [22]. Anomaly detection systems are built on the assumption that attacks are patterns that differ from normal behaviour. Anomaly detectors are designed using statistical models, ML, or rules that characterize normal operations from abnormal ones. During detection, any activity that falls outside a predefined accepted threshold of the normal activity is classified as suspicious [16]. The primary advantage of anomaly-based IDS is its ability to detect previously unseen attacks, including zero-day exploits. However, the use of anomaly-based IDS comes at the cost of producing higher false positives since not every anomaly is malicious [1]. Tuning and training can be used to minimize the high false-positive rate. Most modern anomaly-based IDSs leverage techniques ranging from statistical models and advanced ML algorithms.

This study uses an anomaly-based IDS approach to develop the IDS we proposed in this work using online learning and streaming-based anomaly detection. The choice of an anomaly-based detection aligns with the objective of this thesis, which is to design a self-learning, lightweight, and adaptive IDS for dynamic and resource-constrained environments. Additionally, we selected anomaly-based detection due to its ability to detect unknown attacks, which signature-based IDS cannot inherently detect.

2.2.2 Host-Based vs. Network-Based IDS

The method of deployment of an IDS is another way used to categorize IDSs. An IDS can be a Host-Based IDS (HIDS) or a Network-Based IDS (NIDS). A HIDS is designed to run and detect intrusions on a single host, such as unauthorized changes to system files, anomalies in user logins, or suspicious process behavior. The ability of HIDS to have detailed visibility into a single host enables it to detect insider attacks, such as malware, because insider attacks do not generate network traffic. However, one limitation of HIDS is that it only protects the host on which it resides. This limitation makes it impractical to deploy them across a large number of IoT devices, given the resource-constrained nature of IoT devices [16], [23]. On the other hand, NIDS is designed to monitor network traffic on network endpoints such as routers, gateways, or switches, to detect adversarial traffic in transit between nodes. A typical NIDS examines network packet details such as headers, flow patterns, and payloads to identify malicious activity on the network. The beauty of NIDS is that one NIDS can monitor multiple devices on a network. Since NIDS can monitor network traffic, they are highly effective at identifying several attacks, including scanning services,

denial-of-service attacks, and infiltration, even before these attacks reach the end hosts. Some of the challenges faced by NIDS are encrypted traffic and high throughput of networks, which can lead to malicious content being hidden or the IDS missing a packet due to the high volume of data.

Most modern security solutions often use both HIDS and NIDS in a complementary fashion to build their security architectures. Combining data from host and network IDS can improve detection coverage and provide defense in depth. Recent works on IDS also discuss distributed IDS, where detection responsibilities are shared among multiple nodes or platforms [16], [24]. The choice of an IDS usually depends on the objective to be achieved and deciding on the trade-offs between resource usage and ease of management.

2.2.3 Hybrid and Specification-Based IDS

Considering the strengths of both signature and anomaly-based IDSs, many recent works propose a hybrid of signature and anomaly-based techniques to enhance the detection capabilities of IDS. For example, a hybrid IDS might run a signature-based engine in parallel with an anomaly detector [25]. According to [25], the most common hybrid technique is to use a signature-based method to detect known attacks and use an anomaly-based method to detect unknown attacks. Literature shows that hybrid IDSs can attain broader detection coverage while recording lower false positives than either method alone [26]. However, one of the downsides of hybrid IDS is that it usually requires more resources, making it challenging to deploy on constrained devices. A related concept is specification-based IDS, which is sometimes considered a subset of anomaly detection. In specification-based detection, the system is provided with a set of manually crafted rules or invariants that describe correct operation of a protocol or software; any deviation from these pre-defined normal behavior specifications is flagged as an intrusion [27], [28]. A classical example is in industrial control networks, where a legal sequence of messages can be specified for a given protocol and trigger an alert if an observed event violates those rules. Although specification-based IDS can be effective when it comes to detecting attacks such as protocol abuse or logic attacks and tends to have low positive rates, especially when the specifications are accurate, this comes with a drawback. The drawback is that these specifications need to be manually developed. Additionally, rules for each application or protocol should be maintained, which can be labor-intensive. Notwithstanding, recent research in IoT security has used specification-based methods to detect attacks on contained network protocols like RPL in low-power IoT networks that are not easily covered by generic signatures or learned anomaly models [27], [29].

2.3 Application of IDS in IoT

A system of interconnected embedded devices and sensors that communicate and exchange data over the internet is known as IoT. The general IoT ecosystem includes smart home devices, medical wearables, industrial sensors, and smart cities. The use of intrusion detection in IoT environments has become increasingly crucial in the last ten years, especially when the number of security incidents against IoT devices has grown. IoT devices are often deployed at scale and can be relatively easy targets for attackers due to limited built-in security. Additionally, most IoT devices

have significant resource constraints and cannot run traditional heavy security software, leaving them vulnerable to attacks. This serves as a perfect motivation for designing a specialized IDS that takes into consideration the resource constraints and distributed nature of IoT, especially when these IoT devices lack the processing power to perform such complex intrusion analysis on device [30].

An IDS for traditional computing is difficult to deploy directly on IoT nodes that are computationally constrained. To address this problem, researchers have considered using a distributed architecture for modeling intrusions in IoT systems. This tends to move the heavy computation to more capable nodes at the edge of the IoT network, while they continue to monitor other devices in the ecosystem. One distributed architecture approach that has been introduced is Multi-access Edge Computing (MEC). It involves placing computing resources near the edge of the network, which can run the IDS without incurring the latency of cloud offloading. Leveraging edge servers allows the IoT-based IDs to collect data from various devices and perform detection at the edge that may involve algorithms that use complex detection techniques, providing a balance between timely detection and resource preservation [31], [32].

2.4 Machine Learning-based IDS for IoT

ML has emerged as a fundamental component of modern IDS research, particularly in the realm of anomaly detection, owing to its ability to differentiate between normal and abnormal patterns. In the context of IoT security, characterized by the rapid emergence of novel devices and diverse attack variants, ML-based IDS presents a compelling option due to its inherent adaptability. Over the preceding five years, a significant array of supervised, unsupervised, and deep learning methodologies has been introduced to enhance intrusion detection capabilities within IoT networks [33].

2.4.1 Supervised Learning Approaches

In supervised ML techniques, the algorithm relies on labeled training data to learn. In the context of IDS, a supervised algorithm will have to be trained on labeled network traffic to differentiate between normal and malicious samples [34]. Some of the popular supervised ML algorithms are Decision Trees, Support Vector Machines, Naïve Bayes, k-Nearest Neighbors, Random Forests, and neural network-based classifiers. After the model is trained using a mixture of historical normal and attack data, it can predict labels for unseen data. Supervised ML has been extensively used to build IDS models in the IoT ecosystem. For instance, researchers have trained Random Forest classifiers on IoT network traffic features to detect botnet attacks, achieving high accuracy for the attacks present in the training set [34]. Supervised ML models usually exhibit high detection accuracy on attacks similar to those in the data that was used to train the model. On the other hand, to train the model, you need a comprehensive and labeled dataset from various attack scenarios. This is not just a drawback for supervised ML models but also a limitation in the IoT ecosystem, where obtaining labeled data can be difficult. Additionally, supervised models usually struggle with attacks that were not represented in the training data.

2.4.2 Unsupervised Learning

Unlike supervised ML algorithms, unsupervised ML algorithms do not require labeled data; rather, they seek inherent patterns in the data without prior knowledge of the attack signature [34]. This approach makes unsupervised algorithms extremely useful in environments like the IoT ecosystem. Some of the common unsupervised learning algorithms include clustering algorithms such as k-means, autoencoders, one-class SVM, and isolation forest. One of the drawbacks of using an unsupervised algorithm to build an IDS is that it can produce more false positives compared to supervised learning. This is because not every anomaly is an attack. The inclusion of domain knowledge or feedback from experts is one of the techniques that can be used with an unsupervised algorithm to deal with the massive false positives.

2.4.3 Semi-supervised learning

Semi-supervised learning leverages both supervised and unsupervised algorithms to train a model. In building IDS, a common semi-supervised approach is to train the model on only normal data, such that the model can detect anything outside the normal data as malicious. This approach is essentially an anomaly detection problem, with the only difference being that the normal data is labeled [34]. In some cases, a small amount of labeled attack data is used alongside a large pool of unlabeled data to improve the detection rate. Although semi-supervised methods can leverage abundant unlabeled data and require fewer attack examples, their accuracy may still lag behind compared to fully supervised methods, especially if the unlabeled data contains complex attack behaviors that are hard to discern. Notwithstanding, they provide a practical middle ground, especially in IoT contexts where labeling extensive datasets is not feasible.

2.5 Online ML Techniques

The goal of online ML algorithms is to continuously learn from the incoming data stream. In online ML, instead of a one-time training phase, the models update as new data sample arrives, a condition that allows them to adjust to changes in the data stream. According to [35], online learning is a family of ML algorithms that attempts to tackle predictive tasks by learning from a sequence of data instances one by one at each time. Online ML is able to overcome the shortfalls of offline ML, such as low efficiency in terms of space and time, and poor scalability when it comes to large-scale applications. Online learning algorithms can handle both supervised and unsupervised tasks. Some of the online ML algorithms that have been used in designing IDS include Adaptive Random Forest [36], Support Vector Regression [36], Auto-Associative Deep Random Neural Network [37], online deep neural network [38], Hoeffding Tree [39], and Hoeffding Adaptive Tree [39].

2.6 Commonly Used Datasets for IoT IDS Evaluation

A good IDS needs to be rigorously evaluated using a representative dataset. Creating a real-world dataset in the IoT ecosystem is usually challenging due to the heterogeneity and privacy of the

ecosystem. In spite of these, several datasets have been adopted from traditional IDS research as the standard benchmark datasets for evaluating IDS built for the IoT ecosystem. In this section, we present a brief overview of some of the common datasets used in evaluating IDS for IoT systems. Table 2.1 presents an overview of some of the common IoT-based datasets used in building and evaluating IDS.

Table 2.1: Commonly Used Datasets for IoT IDS Evaluation

S/N	Dataset	Source	Domain	Type	# Features	# Classes	# Benign	# Malicious	# Total
1	IoT-23	[40]	Home	Real	20	16	30,858,735	294,449,255	325,307,990
2	NF-ToN-IoT-v2	[41]	Home	Simulated	43	10	16,792,214	10,728,046	27,520,260
3	Edge-IIoT	[42]	Industry	Simulated	61	15	9,728,708	11,223,940	20,952,648
4	LATAM-DoS-IoT	[43]	Home	Real	18	4	799,187	30,662,911	31,462,098
5	LATAM-DDoS-IoT	[43]	Home	Real	18	4	799,187	49,666,991	50,466,178
6	X-IIOTID	[44]	Industry	Simulated	65	10	421,417	399,417	820,834
7	WUSTL-IIOT-2021	[45]	Industry	Simulated	41	6	87,016	1,107,444	1,194,464
8	MQTT-IOT-IDS2020	[46]	Home & Industry	Simulated	43	5	2,284,994	19,792,003	22,076,997
9	BOT-IoT	[47]	Home	Simulated	32	7	9,543	73,360,900	73,370,443
10	IoTNIDS	[48]	Home	Real	N/A	5	1,756,276	1,229,718	2,985,994
11	IoTNIDS	[48]	Home	Real	N/A	4	347,443	1,152,772	1,500,215
12	MedBIoT	[49]	Home	Simulated	100	2	12,540,478	5,305,089	17,845,567
13	IoTIDS20	[50]	Home	Real	80	5	40,073	585,710	625,783
14	MQTT	[51]	Home	Simulated	33	6	11,915,716	165,463	12,081,179
15	IoT Healthcare Security	[52]	Healthcare	Simulated	50	2	108,568	80,126	188,694
16	WUSTL EHMS 2020	[53]	Healthcare	Simulated	43	2	14,272	2,046	16,318
17	WUSTL-IIoT-2018	[54]	Industry	Simulated	6	2	427,938	6,622,051	7,049,989
18	N-BaIoT	[55]	Industry	Real	114	11	502,595	6,506,674	7,062,606
19		[56]	Home	Real	47	34	1,098,195	92,274,963	93,373,158
20	NF-BoT-IoT-v2	[41]	Home	Simulated	43	5	51,989	16,881,819	16,993,808

2.7 Research Gap

Since one of the characteristics of IoT is its heterogeneity, it is important to design IDS that are able to satisfy this characteristic. IoT networks often include a variety of device types, which may include but are not limited to sensors, actuators, cameras, medical devices, and industrial control systems, with each of these devices using different communication protocols across multiple layers. This heterogeneity means that an IDS meant for the IoT ecosystem must understand various protocols and contexts. Recent research highlights attacks on IoT-specific protocols such as Routing Protocol for Low-Power and Lossy Networks (RPL) [27], [28], [57].

Traditional IDSs face challenges when applied to the IoT ecosystem. One major gap is the lack of adaptivity of these IDSs. They are trained using static data in an offline approach, making it difficult for them to detect evolving attacks. For example, an IDS built to detect attack signatures cannot detect zero-day exploits, and even an anomaly-based IDS model may become obsolete when there are drifts in the IoT network. Several studies have emphasized the need to design IDSs that can self-update by learning from new data in real-time for IoT systems [1], [17], [58]. Another limitation of most current IDSs built for the IoT ecosystem is the high computing resources they use. Whereas these IDSs achieve high accuracy, they are usually built from complex models that make them impractical for the deployment of IoT devices. Most IoT-based IDS research often focuses on detection performance without reporting memory, CPU, or energy usage. This makes it challenging to ascertain the lightweight nature of these solutions and whether they can be deployed in IoT environments.

2.8 Conclusion

In this chapter, we reviewed the state-of-the-art in IDS with a focus on its application to IoT environments. We began with an overview of IDS fundamentals and types. Traditional IDS approaches are categorized into signature-based detection and anomaly-based detection. We also discussed how these approaches are categorized as host-based or network-based, depending on data sources, and how hybrid systems and specification-based methods combine multiple techniques to improve effectiveness. Applying IDS to IoT introduces unique challenges due to resource-constrained devices, heterogeneous protocols, and distributed deployments. The review highlighted that IoT IDS solutions increasingly leverage distributed architectures to offload heavy computations from devices. Techniques like multi-access edge computing (MEC) are used to relocate intrusion analytics to edge servers, enabling complex detection without overwhelming IoT nodes. We saw that IoT-specific IDS often includes protocol-aware or specification-based checks to catch attacks that general-purpose IDS might miss. This addresses IoT-tailored threats such as routing attacks in low-power networks or abuse of IoT device interfaces. A key point is that security for IoT can rarely rely on a single monolithic IDS; instead, multi-layered and cooperative frameworks are recommended for robust defense

ML has become a dominant technique in IDS research for IoT, this is evidenced by numerous studies employing supervised, unsupervised, and deep learning models. Supervised ML algorithms, such as SVM, Random Forest, and neural networks, have achieved high detection rates on known

attack classes in IoT datasets. Unsupervised methods such as clustering and autoencoders have also shown good progress in detecting novel intrusions without requiring labeled attack data. Deep learning, in particular, is a driving force behind recent advances: researchers have applied DNNs, CNNs, RNNs, and LSTMs, and hybrid deep models to automatically learn complex features from IoT data and improve accuracy.

An emerging gap in IoT-based IDS is the need to evaluate models on broader criteria such as energy, memory usage, adaptability to new attacks, and resilience against adversarial evasion. For instance, a deep learning model might score >99% on a test set, but if its memory is too large to run on an edge device or consumes a lot of energy, then its usability is limited.

We also surveyed commonly used datasets, noting that the IDS research community has made progress in developing IoT-specific datasets like BoT-IoT, TON-IoT, among others. These datasets have been instrumental in shifting evaluations towards more IoT-realistic scenarios. The creation of new datasets that include emerging attack vectors will be important for future IDS research. Currently, many studies use a custom mix of datasets, making it difficult to assess which technique is truly superior.

References

[1] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in internet of things", *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[2] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in internet of things: The road ahead", *Computer networks*, vol. 76, pp. 146–164, 2015.

[3] Statista, • *Global IoT and non-IoT connections 2010-2025 | Statista*. [Online]. Available: `https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/` (visited on 07/05/2022).

[4] SAM Seamless Network, *2021 IoT Security Landscape - SAM Seamless Network*. [Online]. Available: `https://securingsam.com/2021-iot-security-landscape/` (visited on 07/13/2022).

[5] J. P. Anderson, "Computer security threat monitoring and surveillance", *Technical Report, James P. Anderson Company*, 1980.

[6] J. R. Vacca, *Computer and information security handbook*. Newnes, 2012.

[7] S. U. Jan, S. Ahmed, V. Shakhov, and I. Koo, "Toward a lightweight intrusion detection system for the internet of things", *IEEE Access*, vol. 7, pp. 42 450–42 471, 2019.

[8] S. Latif, Z. e Huma, S. S. Jamal, *et al.*, "Intrusion detection framework for the internet of things using a dense random neural network", *IEEE Transactions on Industrial Informatics*, 2021.

[9] S. Roy, J. Li, B.-J. Choi, and Y. Bai, "A lightweight supervised intrusion detection mechanism for iot networks", *Future Generation Computer Systems*, vol. 127, pp. 276–285, 2022.

[10] R. Zhao, G. Gui, Z. Xue, *et al.*, "A novel intrusion detection method based on lightweight neural network for internet of things", *IEEE Internet of Things Journal*, 2021.

[11] T. D. Diwan, S. Choubey, H. Hota, *et al.*, "Feature entropy estimation (fee) for malicious iot traffic and detection using machine learning", *Mobile Information Systems*, vol. 2021, 2021.

[12] J. Granjal, J. M. Silva, and N. Lourenço, "Intrusion detection and prevention in coap wireless sensor networks using anomaly detection", *Sensors*, vol. 18, no. 8, p. 2445, 2018.

[13] A. Palmieri, P. Prem, S. Ranise, U. Morelli, and T. Ahmad, "Mqttsa: A tool for automatically assisting the secure deployments of mqtt brokers", in *2019 IEEE World Congress on Services (SERVICES)*, IEEE, vol. 2642, 2019, pp. 47–53.

[14] L. G. Araujo Rodriguez and D. Macêdo Batista, "Program-aware fuzzing for mqtt applications", in *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2020, pp. 582–586.

[15] C. Bormann, A. P. Castellani, and Z. Shelby, "Coap: An application protocol for billions of tiny internet nodes", *IEEE Internet Computing*, vol. 16, no. 2, pp. 62–67, 2012.

[16] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges", *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[17] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets", *Computers & security*, vol. 86, pp. 147–167, 2019.

[18] A. Thakkar and R. Lohiya, "A review of the advancement in intrusion detection datasets", *Procedia Computer Science*, vol. 167, pp. 636–645, 2020.

[19] A. Yastrebova, R. Kirichek, Y. Koucheryavy, A. Borodin, and A. Koucheryavy, "Future networks 2030: Architecture & requirements", in *2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, IEEE, 2018, pp. 1–8.

[20] M. M. Rahman, S. Al Shakil, and M. R. Mustakim, "A survey on intrusion detection system in iot networks", *Cyber Security and Applications*, vol. 3, p. 100 082, 2025.

[21] P.-C. Lin, Y.-D. Lin, and Y.-C. Lai, "A hybrid algorithm of backward hashing and automaton tracking for virus scanning", *IEEE transactions on computers*, vol. 60, no. 4, pp. 594–601, 2010.

[22] M. Ozkan-Okay, R. Samet, Ö. Aslan, and D. Gupta, "A comprehensive systematic literature review on intrusion detection systems", *IEEE Access*, vol. 9, pp. 157 727–157 760, 2021.

[23] O. H. Abdulganiyu, T. Ait Tchakoucht, and Y. K. Saheed, "A systematic literature review for network intrusion detection system (ids)", *International journal of information security*, vol. 22, no. 5, pp. 1125–1162, 2023.

[24] R. A. Bridges, T. R. Glass-Vanderlan, M. D. Iannacone, M. S. Vincent, and Q. Chen, "A survey of intrusion detection systems leveraging host data", *ACM computing surveys (CSUR)*, vol. 52, no. 6, pp. 1–35, 2019.

[25] E. Gyamfi and A. Jurcut, "Intrusion detection in internet of things systems: A review on design approaches leveraging multi-access edge computing, machine learning, and datasets", *Sensors*, vol. 22, no. 10, p. 3744, 2022.

[26] A. S. Dina and D. Manivannan, "Intrusion detection based on machine learning techniques in computer networks", *Internet of Things*, vol. 16, p. 100 462, 2021.

[27] M. A. Kareem and S. Tayeb, "Ml-based nids to secure rpl from routing attacks", in *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, 2021, pp. 1000–1006.

[28] V. Sharma, I. You, K. Yim, R. Chen, and J.-H. Cho, "Briot: Behavior rule specification-based misbehavior detection for iot-embedded cyber-physical systems", *IEEE Access*, vol. 7, pp. 118 556–118 580, 2019.

[29] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions", *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2359–2391, 2017.

[30] P. Ioulianou, V. Vasilakis, I. Moscholios, and M. Logothetis, "A signature-based intrusion detection system for the internet of things", *Information and Communication Technology Form*, 2018.

[31] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Privacy-preserved task offloading in mobile blockchain with deep reinforcement learning", *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2536–2549, 2020.

[32] H. Wu, K. Wolter, P. Jiao, Y. Deng, Y. Zhao, and M. Xu, "Eedto: An energy-efficient dynamic task offloading algorithm for blockchain-enabled iot-edge-cloud orchestrated computing", *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2163–2176, 2020.

[33] K. Albulayhi, A. A. Smadi, F. T. Sheldon, and R. K. Abercrombie, "Iot intrusion detection taxonomy, reference architecture, and analyses", *Sensors*, vol. 21, no. 19, p. 6432, 2021.

[34] B. R. Kikissagbe and M. Adda, "Machine learning-based intrusion detection methods in iot systems: A comprehensive review", *Electronics*, vol. 13, no. 18, p. 3601, 2024.

[35] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey", *Neurocomputing*, vol. 459, pp. 249–289, 2021.

[36] I. Chouchen and F. Jemili, "Intrusion detection based on incremental learning", in *2023 International Conference on Cyberworlds (CW)*, IEEE, 2023, pp. 448–455.

[37] M. Nakıp and E. Gelenbe, "Online self-supervised deep learning for intrusion detection systems", *IEEE Transactions on Information Forensics and Security*, 2024.

[38] O. A. Wahab, "Intrusion detection in the iot under data and concept drifts: Online deep learning approach", *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 19 706–19 716, 2022.

[39] N. Martindale, M. Ismail, and D. A. Talbert, "Ensemble-based online machine learning algorithms for network intrusion detection systems using streaming data", *Information*, vol. 11, no. 6, p. 315, 2020.

[40] A. Parmisano, S. Garcia, and M. J. Erquiaga, "A labeled dataset with malicious and benign iot network traffic", *Stratosphere Laboratory: Praha, Czech Republic*, 2020.

[41] M. Luay, S. Layeghy, S. Hosseininoorbin, M. Sarhan, N. Moustafa, and M. Portmann, *Temporal analysis of netflow datasets for network intrusion detection systems*, 2025. arXiv: 2503.04404 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2503.04404.

[42] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning", *IEEE Access*, vol. 10, pp. 40 281–40 306, 2022.

[43] J. G. Almaraz-Rivera, J. A. Perez-Diaz, J. A. Cantoral-Ceballos, J. F. Botero, and L. A. Trejo, "Toward the protection of iot networks: Introducing the latam-ddos-iot dataset", *IEEe Access*, vol. 10, pp. 106 909–106 920, 2022.

[44] M. Al-Hawawreh, E. Sitnikova, and N. Aboutorab, *X-iiotid: A connectivity- and device-agnostic intrusion dataset for industrial internet of things*, 2021. DOI: 10.21227/mpb6-py55. [Online]. Available: https://dx.doi.org/10.21227/mpb6-py55.

[45] M. Zolanvari, *Wustl-iiot-2021*, 2021. DOI: 10.21227/yftq-n229. [Online]. Available: https://dx.doi.org/10.21227/yftq-n229.

[46] H. Hindy, C. Tachtatzis, R. Atkinson, E. Bayne, and X. Bellekens, *Mqtt-iot-ids2020: Mqtt internet of things intrusion detection dataset*, 2020. DOI: 10.21227/bhxy-ep04. [Online]. Available: https://dx.doi.org/10.21227/bhxy-ep04.

[47] N. Moustafa, *The bot-iot dataset*, 2019. DOI: 10.21227/r7v2-x988. [Online]. Available: https://dx.doi.org/10.21227/r7v2-x988.

[48] H. Kang, D. H. Ahn, G. M. Lee, J. D. Yoo, K. H. Park, and H. K. Kim, *Iot network intrusion dataset*, 2019. DOI: 10.21227/q70p-q449. [Online]. Available: https://dx.doi.org/10.21227/q70p-q449.

[49] A. Guerra-Manzanares, J. Medina-Galindo, H. Bahsi, and S. Nõmm, "Medbiot: Generation of an iot botnet dataset in a medium-sized iot network", English (US), in *ICISSP 2020 - Proceedings of the 6th International Conference on Information Systems Security and Privacy*, ser. ICISSP 2020 - Proceedings of the 6th International Conference on Information Systems Security and Privacy, SciTePress, 2020, pp. 207–218. DOI: 10.5220/0009187802070218.

[50] I. Ullah and Q. H. Mahmoud, "A scheme for generating a dataset for anomalous activity detection in iot networks", in *Advances in Artificial Intelligence: 33rd Canadian Conference on Artificial Intelligence, Canadian AI 2020, Ottawa, ON, Canada, May 13–15, 2020, Proceedings*, Ottawa, ON, Canada: Springer-Verlag, 2020, pp. 508–520, ISBN: 978-3-030-47357-0. DOI: 10.1007/978-3-030-47358-7_52. [Online]. Available: https://doi.org/10.1007/978-3-030-47358-7_52.

[51] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli, and E. Cambiaso, "Mqttset, a new dataset for machine learning techniques on mqtt", *Sensors*, vol. 20, no. 22, p. 6578, 2020.

[52] F. Hussain, S. G. Abbas, G. A. Shah, *et al.*, *Iot healthcare security dataset*, 2021. DOI: 10.21227/9w13-2t13. [Online]. Available: https://dx.doi.org/10.21227/9w13-2t13.

[53] A. A. Hady, A. Ghubaish, T. Salman, D. Unal, and R. Jain, "Intrusion detection system for healthcare systems using medical and network data: A comparison study", *IEEE Access*, vol. 8, pp. 106 576–106 584, 2020. DOI: 10.1109/ACCESS.2020.3000421.

[54] M. Teixeira, M. Zolanvari, and R. Jain, *Wustl-iiot-2018*, 2020. DOI: 10.21227/kzgp-7t84. [Online]. Available: https://dx.doi.org/10.21227/kzgp-7t84.

[55] Y. Meidan, M. Bohadana, Y. Mathov, *et al.*, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders", *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[56] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment", *Sensors*, vol. 23, no. 13, p. 5941, 2023.

[57] K. Yun, P. V. Astillo, S. Lee, J. Kim, B. Kim, and I. You, "Behavior-rule specification-based ids for safety-related embedded devices in smart home", in *2021 World Automation Congress (WAC)*, IEEE, 2021, pp. 65–70.

[58] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation", *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

This page has been intentionally left blank.

Chapter 3

General Methodology

3.1 Overview

The main objective of this work is to develop a self-learning, lightweight IDS for the IoT ecosystem. The proposed IDS can be deployed on resource-constrained devices with the ability to adapt to changes in network traffic and detect unknown attacks in real-time. Although this work has been presented across several chapters, with each chapter representing a publication that focuses on a particular objective, all the solutions proposed in these chapters were developed through a unified and interactive methodology. This chapter outlines the general methodological framework used in this study.

3.2 Research Design

The study adopted a design science research (DSR) methodology [1], [2], iteratively progressing through the following phases:

1. Problem Identification and Motivation
2. Defining Objectives of a Solution
3. Design and Development
4. Demonstration
5. Evaluation
6. Communication

Each of the independent studies (Chapters 4–10) represents a specific phase of refinement or expansion, but all share a common problem domain, objectives, and experimental design.

3.2.1 Problem Identification

The study begins by identifying the limitations of deploying traditional IDSs within the IoT ecosystem. The key limitations identified are;

- The resource-constrained nature of IoT devices.

- The inability of traditional IDS to adapt to real-time IoT network traffic and zero-day threats.
- Lack of lightweight and adaptive IDS models that suit heterogeneous environments like the IoT ecosystem.

The above limitations serve as the problems that need to be solved. This motivates the need for an IDS that fits into the IoT ecosystem, is lightweight in nature, self-learning, and capable of detecting novel threats in real-time.

3.2.2 Defining Objectives of a Solution

The problem identification led to the statement of objectives for the proposed solution. These objectives are;

- Investigate the applicability of ML algorithms for intrusion detection in IoT environments.
- Develop and evaluate models capable of online, incremental learning to adapt to evolving attack patterns and concept drift.
- Reduce computational and memory overhead, making it possible for the proposed system to be deployed on resource-constrained IoT and edge devices.
- Validate the proposed approaches through real-world datasets and deployment scenarios on embedded platforms.

3.2.3 Design and Development

The design and development phase used an iterative design approach to design models that solve the problem identified. The following key milestones were attained.

1. Online Deep Learning with Quantization (Chapter 4)
2. Hybrid Online Models (ALMANET) (Chapter 5)
3. Multi-label Intrusion Detection with an adaptive ALMA with SWA (Chapter 6)
4. Adaptive SAMKNN for Real-time and Zero-day Detection (Chapters 7 and 8)
5. Resource-optimized model using IPCA and SAMKNN (Chapter 9)
6. Incremental Ensemble Learning Techniques (Chapter 10)

Each proposed solution is supported by a clear architectural diagram, description of the algorithm used, and the implementation of techniques that are optimized for resource-constrained environments.

3.2.4 Demonstration

The developed IDS prototypes were deployed and demonstrated as follows;

- On-device training and evaluation using Raspberry Pi.
- Testing of the prototypes on popular benchmark datasets such as NF-BoT-IoT, ToN-IoT, Edge-IIoT, and SCADA.
- Simulation of adversarial attacks and zero-day exploits.

The demonstrations prove that the proposed systems meet the qualities of real-time deployment in resource-constrained environments while recording high detection accuracy and low false positives.

3.2.5 Evaluation

The proposed models were subjected to a common experimental validation to ensure comparability and reproducibility.

1. Datasets: The experimental validation used widely used public datasets such as NF-BoT-IoT, NF-ToN-IoT, CSE-CIC-IDS2018, Edge-IIoTset, and ToN-IoT datasets.
2. Hardware: The models were tested across both simulated platforms and physical edge devices such as Raspberry Pi.
3. Evaluation Metrics: Metrics such as accuracy, F1 score, ROC-AUC, memory usage, CPU utilization, energy consumption, and latency were used to evaluate the efficiency of the proposed system.

3.2.6 Communication

The research was effectively disseminated through;

- Multiple peer-reviewed journal and conference publications.
- Sharing the codes of the algorithms proposed in this work on GitHub.
- Sharing with researchers on areas that they can explore as future work.

3.2.7 Conclusion

Although each chapter focuses on tackling a sub-problem of the main problem, each technique used to solve a sub-problem is an improvement of the proposed system. The different techniques explored in this thesis represent a complementary approach toward solving a single core challenge: how to deploy a self-learning, real-time IDS at the edge of IoT networks.

References

[1] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research", *MIS quarterly*, pp. 75–105, 2004.

[2] K. Pfeffers, T. Tuunanen, C. E. Gengler, *et al.*, "The design science research process: A model for producing and presenting information systems research", in *Proceedings of the First International Conference on Design Science Research in Information Systems and Technology (DESRIST 2006), Claremont, CA, USA*, 2006, pp. 83–106.

This page has been intentionally left blank.

Chapter 4

Chapter 4 is based on the following article:

P. R. Agbedanu, R. Musabe, I. Gatare, J. Rwigema, "A Real-Time Deep Learning IDS for IoT with Dynamic Quantization and Adaptive Online Learning," Submitted for publication (under review).

A Real-Time Deep Learning IDS for IoT with Dynamic Quantization and Adaptive Online Learning

Authors: Promise Ricardo Agbedanu, Richard Musabe, Ignace Gatare, James Rwigema

Abstract: With the number of Internet of Things (IoT) devices estimated to be around 75 billion by the end of 2025, as reported by IHS Markit, maintaining the security of the IoT ecosystem has become a daunting task. This calls for the development of efficient, adaptive, and lightweight intrusion detection systems (IDS) to protect the IoT ecosystem. This chapter presents a novel Deep Online algorithm with Dynamic Quantization and Confidence-Aware Learning Control designed for real-time IoT threat detection. The proposed system integrates a deep feature extractor and a linear classifier head, optimized using a threshold-based mechanism that dynamically pauses and resumes learning based on prediction confidence. Furthermore, dynamic quantization significantly compresses the model, enabling efficient deployment on resource-constrained devices. The system was evaluated on four benchmark IoT security datasets, achieving over 96% reduction in model size without compromising performance. Across all datasets, the quantized model maintained an accuracy, precision, recall, and F1 score of up to 0.99. Additionally, the model recorded an ROCAUC above 0.84, with inference latencies under 1.5 ms. Concept and feature drift experiments demonstrated the model's rapid adaptability to evolving data distributions, while adversarial and noise evaluation confirmed the model's robustness against evasion and perturbation attacks. This study establishes an efficient memory IDS for real-time IoT systems that easily

adapts to drifts and is robust against adversarial perturbation.

4.1 Introduction

The last ten years have seen the adoption of technologies like smart grid, smart home, smart healthcare, smart transportation, and smart industry, which all form part of the Internet of Things (IoT) ecosystem. The IoT ecosystem makes it possible for devices from domains such as healthcare, home, energy, transportation, industry, and agriculture to communicate with each other via the internet. The connectivity and communication among these devices expand the network attack surface. A classical example is seen in IoT botnet attacks like Mirai. This calls for robust intrusion detection approaches to secure the ecosystem [1], [2]. Deep learning techniques have been extensively used in recent years to develop IDS for IoT-based systems. Literature shows that deep learning techniques achieve superior detection accuracy compared to traditional signature-based approaches [3].

According to [4], neural networks are capable of learning complex feature representations of network traffic automatically. They can even detect subtle attack patterns that rule-based systems might miss. However, deploying deep learning-based IDS, which are usually expensive in terms of computations on IoT edge devices, remains a challenging scenario due to the computational limitations of these devices. These computational limitations include CPU power, memory, and energy. Additionally, IDS models built with standard deep learning models, such as CNN or LSTM, incur high inference latency as well as consume a substantial amount of energy. This makes these models practically difficult to deploy on IoT edge nodes [5]. Similarly, Li et al. [6] stressed the need to push intelligence to the network edge for IoT, highlighting that deep models that are deployed for the IoT ecosystem must be optimized for edge computing environments. A practical and promising solution is to compress and quantize the model after it has been trained, an approach that can drastically reduce the memory and computation footprint of neural networks. Techniques such as weight pruning and low-bit quantization have been shown to shrink model size by an order of magnitude without significant loss in accuracy [7]. For example, Han et al. [7] achieved up to 49x compression of CNN models through a combination of pruning and quantization. An approach that enables this complex neural network to fit in the memory of an IoT device

without significantly affecting the accuracy. A recent work proposed by Wang et al. [4] adopted a dynamic quantization technique to develop an IDS for IoT systems. The results of the work demonstrate that 8-bit integer networks can maintain high attack detection accuracy while improving the inference efficiency. These advances indicate that quantization is a viable approach to implementing lightweight IDS models for IoT.

Another challenge confronting the monitoring of IoT security is the dynamic nature of data streams. IoT networks usually exhibit non-stationary behavior. Both normal traffic and attack patterns can change over time. For instance, a new device may join the network, a firmware may be updated, or a new attack may be introduced to the network [8]. This leads to a problem called concept drift, a situation where an IDS model trained on historical data gradually becomes obsolete as the data distribution changes. A static model may become stale and unable to recognize new attack patterns or adapt to changes in benign traffic, leading to degradation in the detection performance [9]. Earlier works, such as the work done by Yin et al. [10], emphasized that continuous learning and model adaptation are important when it comes to long-term deployment of IDSs in streaming environments. Techniques such as online learning, incremental model updates, and drift detection have been explored to address this challenge. Mirsky et al. [11] proposed an online IDS that uses an ensemble of autoencoders to continuously learn from real-time traffic. The proposed approach highlights the feasibility of real-time on-device learning using lightweight models. Compared to Mirsky et al. [11], a recent study by Tabassum et al. [12] proposed an incremental learning technique for a distributed IDS for the IoT ecosystem. In the proposed approach, the model updates itself with new data to improve the detection of evolving attacks.

These works above reveal a trend towards adaptive IDSs that can keep up with changing attack patterns. However, continuous retraining of deep models on streaming data can be computationally expensive and may itself strain IoT device resources or risk the model becoming unstable. A naive online learning strategy that updates on every new sample would negate the gains of model compression.

In this chapter, we address the two challenges of resource efficiency and adaptability by introducing a novel IDS framework that is both lightweight for edge deployment and capable of online self-learning. Our key contributions are as follows:

1. **Integration of Dynamic Quantization:** We incorporate post-training dynamic quantization into the IDS model pipeline to compress the online deep neural network. After initial training, weights are converted to 8-bit integers and inference is performed using integer arithmetic, significantly reducing memory usage and inference latency. Unlike static quantization, the dynamic approach quantizes weights on the fly and adjusts activation quantization ranges at runtime, preserving accuracy for streaming data. To our knowledge, this work is one of the first to apply dynamic quantization to an IoT IDS, demonstrating that even deep models can run efficiently on IoT hardware with minimal performance loss.

2. **Adaptive Online Learning with Confidence-Based Control:** We propose a dynamic threshold-controlled online learning algorithm that enables the IDS to adapt to concept drift in network traffic while minimizing unnecessary updates. The IDS monitors its own prediction confidence on incoming data; when the confidence drops below a defined threshold, the system automatically enters a learning mode to update the model on the new data. Once the model's confidence recovers above the threshold, learning is paused. This approach effectively acts as a built-in trigger to perform on-demand incremental learning, allowing the model to quickly learn emerging attack behaviors or accommodate benign changes. By pausing updates when the model is confident and network conditions are stable, we avoid continuous retraining overhead and preserve energy, an important consideration for battery-powered IoT devices. This adaptive training control is lightweight and data-driven, requiring only the computation of a confidence score, such as the softmax probability for the predicted class, and a simple comparison against a threshold.

3. **System Implementation and Evaluation:** We design and implement the IDS architecture in PyTorch, comprising a deep feature extractor and a binary classification head. The feature extractor can be realized by various deep neural networks. The classification head is a simple fully-connected layer producing a binary output. We detail how dynamic quantization is applied to this model using PyTorch's quantization toolkit, and how the online training mechanism is integrated such that the model can switch between inference and update modes seamlessly.

The rest of this chapter is organized as follows. Section 2 reviews related work. Section 3 describes the proposed IDS architecture and algorithms in detail, including the methodology used. Section 4 presents the experimental setup. We present the results of our experiment in Section 4. The discussions of the study are presented in Section 5. Finally, Section 6 concludes the chapter.

4.2 Related Work

Modern IoT-based IDS has increasingly explored deep learning techniques due to its ability to automatically derive meaningful features from complex data. Over the past ten years, a variety of deep neural network (DNN) architectures have been explored when it comes to detecting cyber-attacks in the IoT ecosystem [1], [13]. Soe et al. [14] proposed a sequential model combining recurrent neural networks (RNNs) and temporal convolution to detect IoT botnet traffic. In another study, Pektaş et al. [15] used a hybrid approach consisting of RNN and CNN to detect botnet attacks. Mirsky et al. [11] used an ensemble of autoencoders to detect anomalies in IoT traffic. The design approach adopted by [11] resulted in a lightweight IDS small enough to run on a Raspberry Pi. Researchers have also explored deep architectures that stack multiple types of layers to address IoT security problems. Manimurugan et al. [16] used a deep belief network (DBN) to develop an IDS to detect intrusions in the Internet of Medical Things environment. Alotaibi & Alotaibi [17] used a stack of deep learning models combining autoencoders and fully-connected networks to detect cyber-attacks in IoT networks. More complex pipelines have been proposed to extract different features within the IoT ecosystem. A recent work by Susilo et al. [1] described a multi-stage IDS based on a hybrid deep learning algorithm where an autoencoder is first used to reduce the dimensionality of the input features, and then a CNN classifier is used to make the prediction. The studies demonstrate that deep learning techniques have become one of the foundational approaches in modern IoT IDS research.

However, one of the common limitations of these studies is the assumption of the availability of powerful computational resources. In a cloud environment, large DNN algorithms can be trained and evaluated on IoT data, but directly deploying such heavy models on IoT edge devices is challenging due to the computational

constraints of these IoT edge devices. For instance, training an LSTM or CNN on-device is usually infeasible. Even running inference on a deep model may limit the computation of an IoT edge device. Some researchers try to mitigate this by offloading the IDS to the edge or cloud servers, as in Li et al. [6], where an edge computing platform is used to host deep learning based IDS for IoT systems. While this approach is effective, offloading the IDS to an edge platform introduces network latency and privacy concerns. Therefore, there is strong motivation for making IDS models themselves more efficient so that at least inference, if not training, can be done locally on IoT gateways or devices.

Model compression techniques have been widely studied to enable deployment of deep networks on resource-constrained hardware like microcontrollers, mobile phones, and IoT devices. Quantization is a technique that reduces the precision of neural network parameters and operations from floating-point to lower-bit representations. Using the lower precision leads to a reduction in model size, and inferences can be accelerated on CPUs that support integer arithmetic. If done properly, quantization yields minimal degradation in model accuracy [5]. In their work, Jacob et al. [5] introduced a quantization that allowed inference with integer-only arithmetic. Their approach, adopted in TensorFlow-Lite, demonstrated that even complex models (like CNNs on ImageNet) could be quantized to 8-bit weights and activations with little loss in accuracy by carefully calibrating scale factors and performing quantization-aware training. This enabled significant speed-ups and made on-device deep learning more feasible. This technique, adopted in TensorFlow Lite, has shown that even complex models can be quantized to 8-bit weights and activations with little loss in accuracy. Quantization enables on-device deep learning to be more feasible. Han et al. [7] also proposed a method called deep compression, a pipeline of pruning and quantization that achieved up to 49x reduction in model size without accuracy loss. The authors showed that a larger CNN model could be compressed from 240 MB to 6.9 MB while preserving the model's accuracy. They also showed that the compressed model was 3 to 7x more energy efficient during inference. Such approaches are particularly relevant for IoT, where devices often run on a battery or have energy limitations. Quantization can be applied at different stages and precisions. In this work, we focus on dynamic post-training quantization as provided by PyTorch, which primarily quantizes the weight matrices of linear layers and optionally LSTM layers to 8-bit. Dynamic

quantization means that while weights are stored and used in INT8, the activations are quantized dynamically at runtime using a scale and zero-point computed from their distribution on the fly just before the matrix multiplication, and de-quantized back to float after. This approach is simple because it does not require retraining the model with quantization awareness and is known to work well for models where the majority of the compute is in linear layers. This allows for a quick compression of a trained model and leverages efficient INT8 operations available in libraries like Facebook General Matrix Multiplication (FBGEMM). Wang et al. [4] leveraged PyTorch's dynamic quantization in their lightweight IoT IDS model. The authors reported that the approach streamlined the intrusion detection process by significantly cutting down model size and inference time while maintaining high accuracy on datasets like CIC-IDS2017 and N-BaIoT, which were used in validating their work. In our work, we similarly employ the dynamic quantization technique to compress the deep feature extractor and the classifier of our proposed IDS. We reduce the 32-bit weights to 8 bits, allowing us to achieve about 4x reduction in memory usage, which is a requirement for on-device training and model deployment in the IoT space. Additionally, integer arithmetic can produce an inference speedup.

Another advantage of quantization is energy efficiency. Embedded CPUs often have optimized instructions for fixed-point math, consuming less power per operation than floating-point. This can extend the battery life of IoT sensors that perform on-board ML inference.

However, it is important to note that extreme quantization or aggressive pruning can impair a model's ability to learn new information, which is a consideration for our online learning scenario. Therefore, we choose a balanced strategy, 8-bit quantization, which provides compression with minimal impact on model capacity. We also maintain the ability to switch the model back to higher precision for retraining when needed. Some latest research has explored mixed-precision or adaptive quantization, where portions of the network run in different precisions to balance accuracy and efficiency; this is outside our current scope.

Online ML learning algorithms learn and update their parameters as new data arrives. This technique enables IDs to update their model continuously as new samples arrive, rather than being fixed after initial training. This approach is important for the IoT ecosystem, where concept drift is common due to evolving

network conditions [18]–[20]. Several research works have used this technique to develop intrusion detection. Shyaa et al. [21] leveraged an incremental genetic programming ensemble that could adapt to drifts in network features over time, applied to streaming datasets. While drift adaptation is beneficial, it may lead to computational overhead due to frequent retraining and may also lead to catastrophic forgetting. Some researchers use window-based online learning, a technique where the model is retrained on a window of recent data while gradually forgetting older data. Although this approach can handle drifts, the IDS model may struggle with novel attacks [9], [22].

The solution proposed in our work uses a confidence-based trigger to decide when to update the model. The approach is conceptually related to drift detectors, which often observe the changes in the model's performance. Monitoring the model's prediction confidence on unlabeled data can serve as an implicit drift detector. If the model starts to become uncertain about data that it previously handled with high confidence, it may indicate that the incoming data distribution is shifting beyond the model's learned decision boundaries. A low-confidence prediction could flag either a potential new type of attack or a new benign pattern not seen in training.

In our proposed method, because the system can pause learning, we effectively allow the model to plateau on new knowledge and only trigger learning again when needed, providing some stability.

4.3 Methodology

This section presents the technical details of our proposed technique. The model is designed for online binary classification in streaming scenarios. It incorporates a mechanism to control learning activity based on prediction confidence and applies dynamic quantization to optimize memory usage during inference.

4.3.1 Problem Definition

Let $\mathcal{X} \subset \mathbb{R}^d$ denote the input feature space and $\mathcal{Y} = \{0, 1\}$ denote the binary label space. The model observes a stream of data instances (x_t, y_t) at each timestep t, where $x_t \in \mathcal{X}$ and $y_t \in \mathcal{Y}$. The goal is to incrementally update a classifier $f_\theta : \mathcal{X} \to [0, 1]$ to approximate the posterior probability $\mathbb{P}(y_t = 1 \mid x_t)$.

4.3.2 Model Architecture

The classifier is a composition of two neural modules:

$$f_\theta(x) = \sigma\left(h_\phi(g_\psi(x))\right) \tag{4.1}$$

- $g_\psi : \mathbb{R}^d \to \mathbb{R}^m$ is a deep feature extractor parameterized by ψ
- $h_\phi : \mathbb{R}^m \to \mathbb{R}$ is a linear classification head parameterized by ϕ
- $\sigma(z) = \frac{1}{1+e^{-z}}$ is the sigmoid activation function
- $\theta = \{\psi, \phi\}$ are the trainable parameters

The feature extractor consists of multiple fully connected layers with Layer Normalization, activation functions (e.g., GELU, ReLU), and dropout regularization.

4.3.3 Online Learning with Confidence-Gated Updates

Let $\hat{y}_t = f_\theta(x_t) \in [0, 1]$ denote the predicted probability for label 1 at time t. Define the *confidence score* c_t as:

$$c_t = \begin{cases} \hat{y}_t & \text{if } y_t = 1 \\ 1 - \hat{y}_t & \text{if } y_t = 0 \end{cases} \tag{4.2}$$

A sliding window $C_t = \{c_{t-w+1}, \ldots, c_t\}$ of size w is maintained to compute the mean confidence:

$$\bar{c}_t = \frac{1}{|C_t|} \sum_{i=t-w+1}^{t} c_i \tag{4.3}$$

The update to model parameters is controlled by two thresholds:

- Pause threshold τ_p (e.g., 0.95)
- Resume threshold τ_r (e.g., 0.90)

The model enters a *paused state* if $\bar{c}_t > \tau_p$ and resumes learning when $\bar{c}_t < \tau_r$. This leads to the conditional training rule:

$$\theta_{t+1} = \begin{cases} \theta_t & \text{if paused (no update)} \\ \theta_t - \eta \nabla_\theta \mathcal{L}(f_\theta(x_t), y_t) & \text{otherwise} \end{cases} \tag{4.4}$$

where η is the learning rate, and $\mathcal{L}$ is the *binary cross-entropy loss*:

$$\mathcal{L}(\hat{y}_t, y_t) = -y_t \log \hat{y}_t - (1 - y_t) \log(1 - \hat{y}_t) \tag{4.5}$$

The idea of using a confidence-gated update stems from [23], [24], although the approach used in our work is different from the approach used by [23], [24].

4.3.4 Dynamic Quantization

To optimize inference efficiency, the model supports *dynamic quantization* of the feature extractor and classifier head:

$$\hat{f}_\theta(x) = \sigma\left(\hat{h}_{\phi_q}\left(\hat{g}_{\psi_q}(x)\right)\right) \tag{4.6}$$

where $\hat{g}_{\psi_q}$ and $\hat{h}_{\phi_q}$ are quantized versions of g_ψ and h_ϕ respectively. Quantization is applied to all `Linear` layers using 8-bit integers via PyTorch's quantization engine:

$$\text{Quantize}(w) = \left\lfloor \frac{w - \mu_w}{s} \right\rceil, \quad \text{where } s = \frac{\max(w) - \min(w)}{2^b - 1} \tag{4.7}$$

with $b = 8$ bits and μ_w the mean of weights w.

4.3.5 Model Size Estimation

The total model size in kilobytes is computed as:

$$S = \frac{1}{1024} \sum_i n_i \cdot s_i \tag{4.8}$$

where n_i is the number of elements and s_i is the byte size of each element in the i-th tensor parameter.

The pseudocode of the proposed algorithm is shown in Algorithm 1.

4.3.6 The System Block Diagram of Our Proposed IDS

4.3.6.1 IoT Network Data

This module represents various IoT devices that generate raw network flow. The data include features such as packet size, packet count, byte count, flow duration, among others, that are extracted from NetFlow, SCADA protocols, or sensor communication.

Algorithm 1 Deep Online With Dynamic Quantization

Procedure INITIALIZE($\theta_f, \theta_c, \alpha, \tau_p, \tau_r, W$)
Init θ_f, θ_c, optimizer with LR α
Set thresholds τ_p, τ_r; window size W
$conf \leftarrow [\,]$, $paused \leftarrow$ **False**

Function PREDICTPROBA(x)
$x_t \leftarrow$ DICTTOTENSOR(x)
$p \leftarrow \sigma(c(f(x_t)))$
return $\{0 : 1-p,\ 1 : p\}$

Procedure LEARNONE(x, y)
$y_{lbl} \leftarrow 1.0$ **if** y=1 **else** 0.0
$x_t \leftarrow$ DICTTOTENSOR(x)
$p \leftarrow \sigma(c(f(x_t)))$ {no grad}
$conf_i \leftarrow p$ **if** y=1 **else** $1-p$
Append $conf_i$ to $conf$, truncate if $|conf| > W$
$avg \leftarrow$ mean($conf$)
if $paused$ **and** $avg < \tau_r$ **then**
  $paused \leftarrow$ **False**
end if
if $\neg paused$ **and** $avg > \tau_p$ **then**
  $paused \leftarrow$ **True**
  return
end if
$h \leftarrow f(x_t),\ logits \leftarrow c(h)$
$p \leftarrow \sigma(logits)$
$loss \leftarrow$ BCELoss(p, y_{lbl})
Update θ_f, θ_c via backprop

Procedure QUANTIZEMODEL()
Apply dynamic quantization to f, c
Store in $quantized\_model$

Function MODELSIZE(q)
$model \leftarrow$ quantized **if** q **else** full
$size \leftarrow 0$
for all param in model **do**
  $size \leftarrow size +$ `numel` $\times$ `elem_size`
end for
return $size/1024$

4.3.6.2 Preprocessing Module

Incoming dictionary-formatted feature vectors are converted into dense tensors using a feature index map. This standardized transformation enables the deep model to process structured input data efficiently.

4.3.6.3 Deep Feature Extraction Module

This component employs a multi-layer neural network to extract high-dimensional embeddings from the input tensors. These embeddings capture non-linear interactions among input features and provide a robust representation of the behavior of the network traffic.

4.3.6.4 Classifier Module

A lightweight classification head produces a binary decision on whether the traffic is either an attack or normal by applying a sigmoid activation to the extracted features. This module operates as the core inference engine for real-time threat detection.

4.3.6.5 Learning Control Logic Module

For our proposed system to handle concept drift, this module continuously monitors the model's confidence in its predictions. If the confidence drops below a predefined threshold, it triggers an update to fine-tune the model. When the confidence exceeds a second threshold, learning is paused to conserve resources.

4.3.6.6 Quantization Module

A post-training dynamic quantization to convert model weights from 32-bit floats to 8-bit integers is performed by this module. Quantization significantly reduces the memory footprint and computation time, enabling real-time inference on resource-constrained edge devices.

4.3.6.7 Deployment Module

Once trained and quantized, the model is deployed on a Raspberry Pi 5 model as an IoT gateway or edge node. This module manages the runtime environment and ensures timely prediction delivery with minimal latency.

4.3.6.8 System Output Module

The final output is a binary classification indicating whether the observed traffic is benign or malicious. This module enables the prediction output to be logged, visualized, or trigger automated alert mechanisms. The system block diagram of our proposed IDS is shown in Figure 5.1.

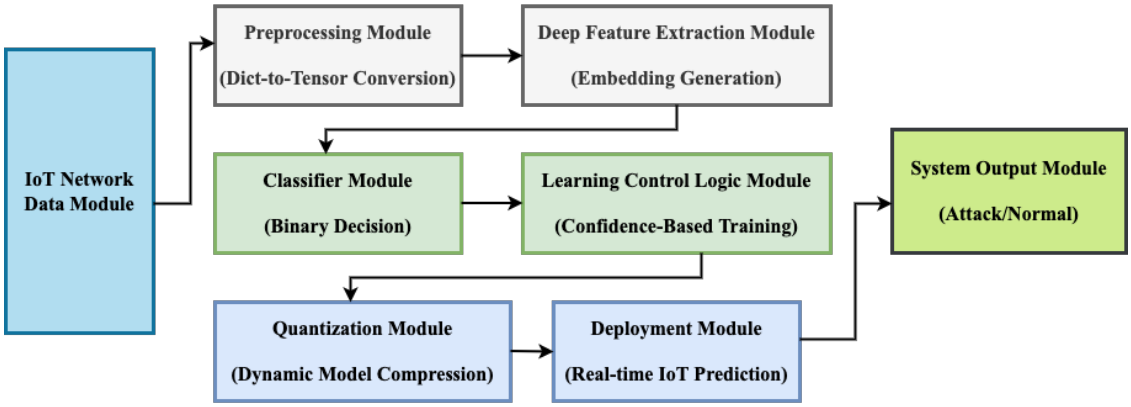


Figure 4.1: A system block diagram of our proposed deep online-based IDS for IoT systems with a dynamic quantization

4.4 Experimental Setup and Evaluation

We used an on-device training approach to build and validate our proposed system. The experiments were conducted on a Raspberry Pi 5 model running on 8 GB of RAM and an SD card storage space of 32 GB. Because our proposed system uses a streaming algorithm, we used the River streaming library that supports online ML proposed by [25]. This allows our proposed system to support real-time learning and detection.

4.4.1 Datasets

Four datasets were used to train and validate the proposed system. The datasets used are the NF-BoT dataset [26], NF-ToN dataset [26], the SCADA dataset [27], and the EdgeIIoT dataset [28]. The details of the datasets used are presented in the Table 4.1.

Table 4.1: A summary of the number of features, the number of samples, the number of attack samples, the number of benign samples, and the total number of samples of each of the four datasets that were used for our experimental validation.

#Features	#Attack	#Benign	Total Samples
NF BoT IoT			
44	150,514	540	151,054
NF ToN IoT			
44	43,496	24,266	67,762
EdgeIIoT			
43	133,499	24301	157,800
SCADA			
6	50,000	50,000	100,000

4.4.2 Evaluation Metrics

To determine the efficiency of our proposed system, we conducted an extensive evaluation and determined its usage and feasibility based on metrics such as accuracy, prediction consistency, precision, recall, F1, ROCAUC, model size, memory, and energy usage.

4.4.3 Experimental Validation

Five different experiments were conducted to validate the efficiency of the proposed system.

4.4.3.1 Baseline Performance

During the first experimental validation, we performed a baseline assessment of the proposed algorithm by comparing the performance before and after quantization. We compared the accuracy, precision, recall, F1, and ROCAUC of the model before and after quantization.

4.4.3.2 Resource Usage Analysis

In the second experiment, we measured the amount of computational resources used by the proposed IDS. The resources measured include model size, memory usage, the amount of energy used, and the inference time per sample. We used a tool called PowerJoular [29] to measure the energy usage of the proposed system.

4.4.3.3 Adaptation to Concept Drift

We also evaluated the performance of our proposed approach when it is confronted with drifts. The system was evaluated on two types of concept drifts. The first is

label drift, a situation where there is a change in the label of the underlying data, and the second is feature drift. In feature distribution drift, the change in the data pattern occurs within the input features.

4.4.3.4 Performance against Adversarial Perturbation and Noisy Data

In this experiment, we perturbed the data with some adversarial perturbations and noisy data, and then used it to test the model. In the case of the adversarial examples, we perturbed the input features with an epsilon value of 0.3. On the other hand, we used a Gaussian technique to add a noise level of 0.2 to the input features. First of all, the model is trained on the first 5000 samples. After training, predictions are made on the next 20,000 samples.

The equations of the two types of perturbations applied to clean input samples are as below:

Gaussian noise injection, as shown in equation 9, adds random noise sampled from a normal distribution to all input features.

$$x_i' = x_i + \mathcal{N}(0, \sigma^2) \tag{4.9}$$

The adversarial shift, as shown in equation 10, applies small but consistent shifts in alternating feature directions. This simulates evasion attacks, where adversaries craft inputs to mislead the classifier without changing the semantic class.

$$x_i' = \begin{cases} x_i + \epsilon & \text{if } i \text{ even} \\ x_i - \epsilon & \text{if } i \text{ odd} \end{cases} \tag{4.10}$$

Besides accuracy, we used prediction consistency to measure the efficiency of the proposed model. Prediction consistency, as shown in equation 11, measures the proportion of outputs from perturbed inputs that match the original (clean) predictions:

$$\text{Consistency} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(y_i = y_i') \tag{4.11}$$

Where y_i is the clean prediction, and y_i' is the perturbed prediction.

4.4.3.5 Scalability Analysis

Finally, we performed a scalability analysis of the proposed system. We tested the predictive capability of the proposed system by feeding it with the sample rates of 500, 1000, 5000, and 10000 while measuring the total time taken to process the sample rate, the actual sample processing rate per second, and the average inference and learning latency per sample.

4.5 Results

In the preceding subsections, we present the results after validating the proposed system through various experiments.

4.5.0.1 Baseline Performance

Table 4.2 presents a comparative analysis of the proposed model's classification performance before and after quantization across four benchmark datasets. The results show the model maintained comparable performance after quantization in terms of accuracy, F1-score, and ROCAUC. The model recorded an accuracy of 0.9992 and a ROCAUC of 0.9544 post-quantization when it was evaluated with the NF-BoT-IoT dataset, compared to 0.9984 and 0.8377, respectively, pre-quantization. Similar trends in terms of performance were observed when the other datasets were used to evaluate the proposed model. The NF-ToN-IoT dataset recorded a post-quantization accuracy of 0.9675 and ROCAUC of 0.9595. The EdgeIIoT and SCADA datasets recorded accuracies of 0.9318 and 0.9878, respectively, with corresponding ROCAUC values of 0.8484 and 0.9878 post-quantization.

Table 4.2: Comparing the accuracy, F1, and ROCAUC of the proposed model before and after quantization

Accuracy	**F1**	**ROCAUC**	**Accuracy**	**F1**	**ROCAUC**
Before Quantization			After Quantization		
NF BoT IoT					
0. 9984	0.9992	0.8377	0.9992	0.9996	0.9544
NF ToN IoT					
0.9478	0.9600	0.9370	0.9675	0.9750	0.9595
EdgeIIoT					
0.9194	0.9540	0.7640	0.9318	0.9600	0.8484
SCADA					
0.9878	0.9877	0.9878	0.9878	0.9877	0.9878

4.5.0.2 Resource Usage Analysis

Table 4.3 summarizes the resource efficiency of the quantized model. A significant model size reduction of over 96% was achieved across all datasets, with the quantized model occupying only 1.56 KB compared to the original 44–53 KB. Despite this compression, memory usage remained consistent at 1.4 MB, and the CPU utilization and energy consumption were steady at approximately 70.97% and 7.71 W, respectively. Inference time decreased modestly, with SCADA achieving the lowest latency at 1.0475 ms. These results confirm the feasibility of deploying the quantized model in real-time, IoT gateway environments without diminishing the model's predictive performance.

Table 4.3: The resource usage of the quantized model

Model Size 1 (KB)	**Model Size 2 (KB)**	**Reduction**	**Mem (%)**	CPU (%)	Energy (W)	Inference (ms)
NF BoT IoT						
52.75	1.56	97.05%	1.4	70.97	7.71	1.3112
NF ToN IoT						
52.75	1.56	97.05%	1.4	70.97	7.71	1.2755
EdgeIIoT						
53.00	1.56	97.06%	1.4	70.97	7.71	1.4195
SCADA						
44.00	1.56	96.46%	1.4	70.97	7.71	1.0475

4.5.0.3 Adaptation to Concept Drift

In this subsection, we evaluated the model's adaptation to concept drift. The results presented in Figures 4.2 to 4.9 show that the model can adapt to both label and feature drift across the four datasets.

Figures 4.2 and 4.3 show that the model maintained a consistently high accuracy on label drift, with the model quickly stabilizing after the drift when evaluated on the NF-BoT IoT dataset. On the other hand, the model displayed some fluctuations in terms of feature drift, but gradually converged back. This shows that the model is adaptive in both drift scenarios.

In Figures 4.4 and 4.5, on the NF-ToN IoT dataset, it can be observed that the model shows a drop in accuracy before the introduction of the drift point. However, the model shows recovery and stabilization after the drift. When it comes to feature drift, the model shows a steady increase in accuracy before the drift and continues to maintain stable performance after the drift. This is an indication that the model

effectively learns and generalizes over time, even in the midst of drifts.

On the EdgeIIoT dataset shown in Figures 4.6 and 4.7, the model recorded and sustained high performance both before and after concept and feature drift points. The accuracy was above 0.90 throughout, showing the model's adaptiveness to dynamic changes in both label and feature distributions.

On the SCADA dataset, as shown in Figures 4.8 and 4.9, it could be observed that the model showed a sharp recovery after an initial drop in accuracy before the point of concept drift while maintaining a stable accuracy after adapting to the drift. In the midst of feature drift, the accuracy remained consistently high, indicating that drift has minimal impact on the model's performance.

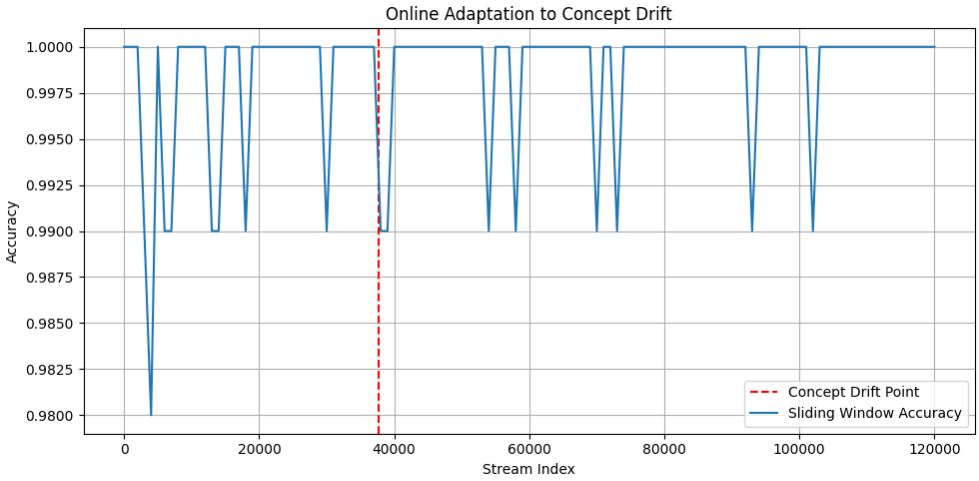


Figure 4.2: The performance of the model with reference to its adaptation to label drift on the NF-BoT IoT dataset.

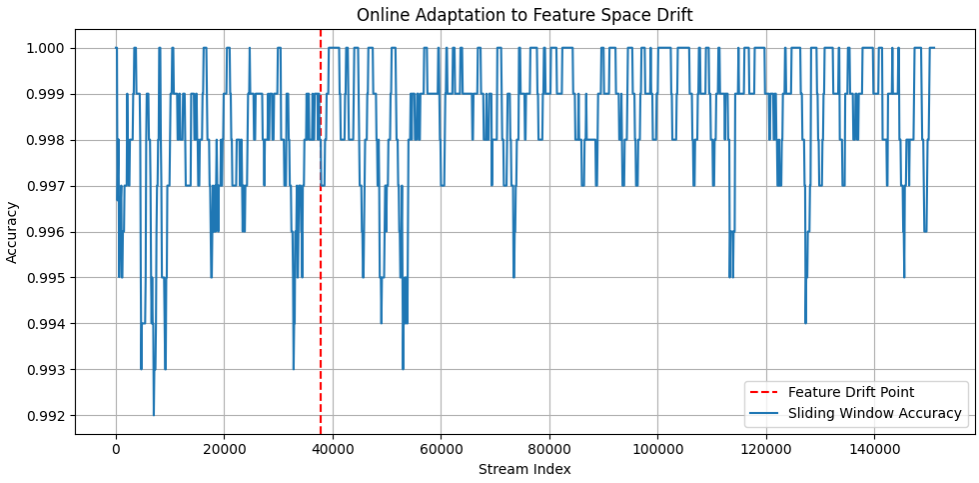


Figure 4.3: The performance of the model with reference to its adaptation to feature drift on the NF-BoT IoT dataset.

4.5.0.4 Performance against Adversarial Perturbation and Noisy Data

From Table 4.4, we can observe the level of robustness of the proposed IDS model against adversarial perturbations and noisy data inputs. Accuracy 1 from the table

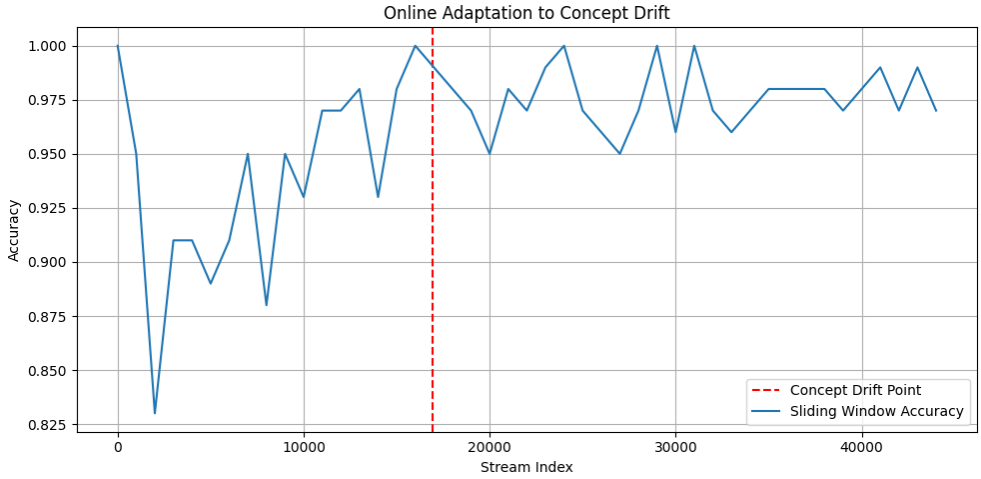


Figure 4.4: The performance of the model with reference to its adaptation to label drift on the NF-ToN IoT dataset.

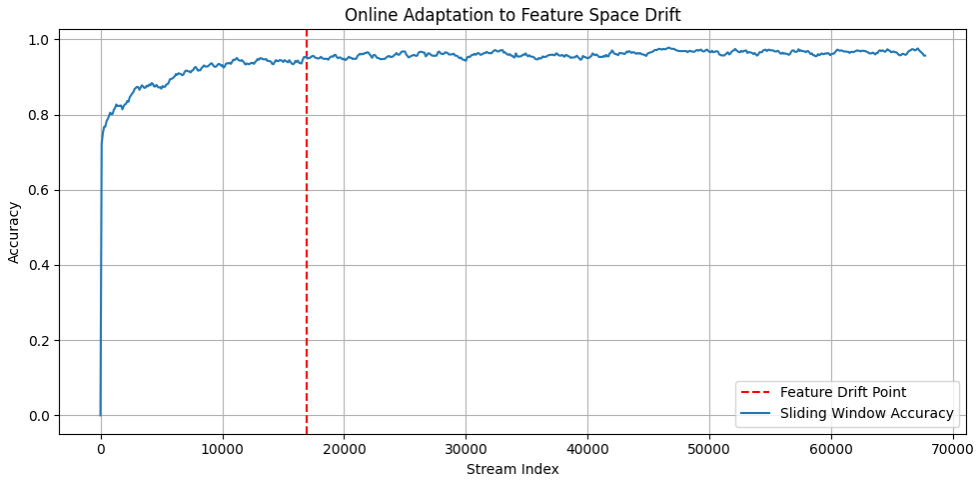


Figure 4.5: The performance of the model with reference to its adaptation to feature drift on the NF-ToN IoT dataset.

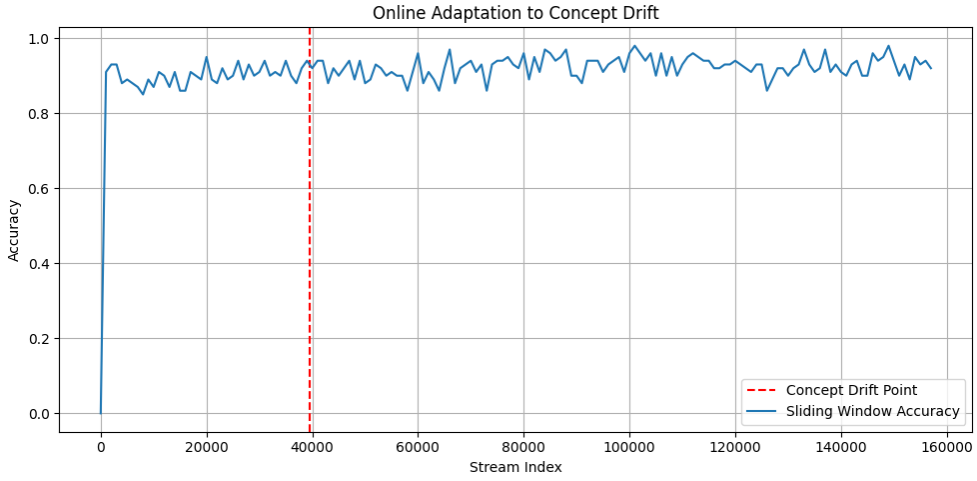


Figure 4.6: The performance of the model with reference to its adaptation to label drift on the EdgeIIoT dataset.

shows the model's performance on clean data, with accuracy 2 and 3 showing the model's performance on Gaussian noise and adversarial samples, respectively.

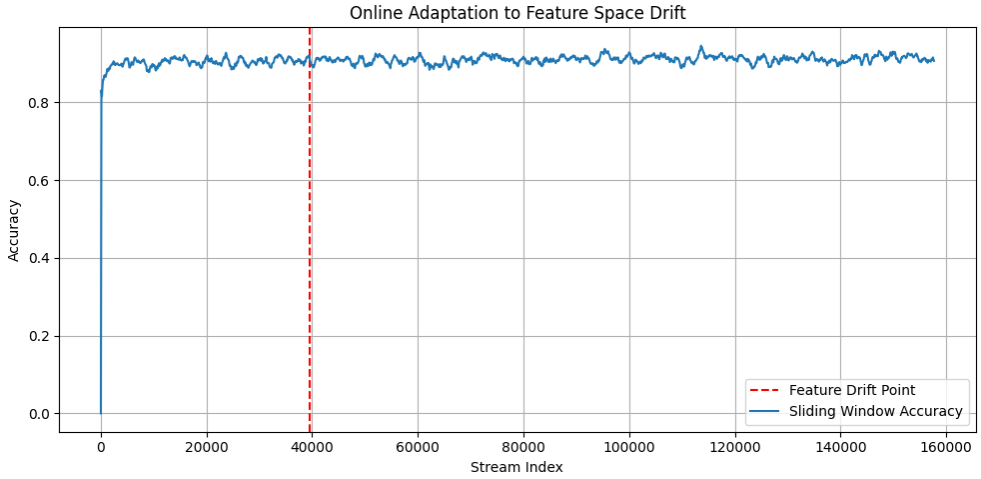


Figure 4.7: The performance of the model with reference to its adaptation to feature drift on the EdgeIIoT dataset.

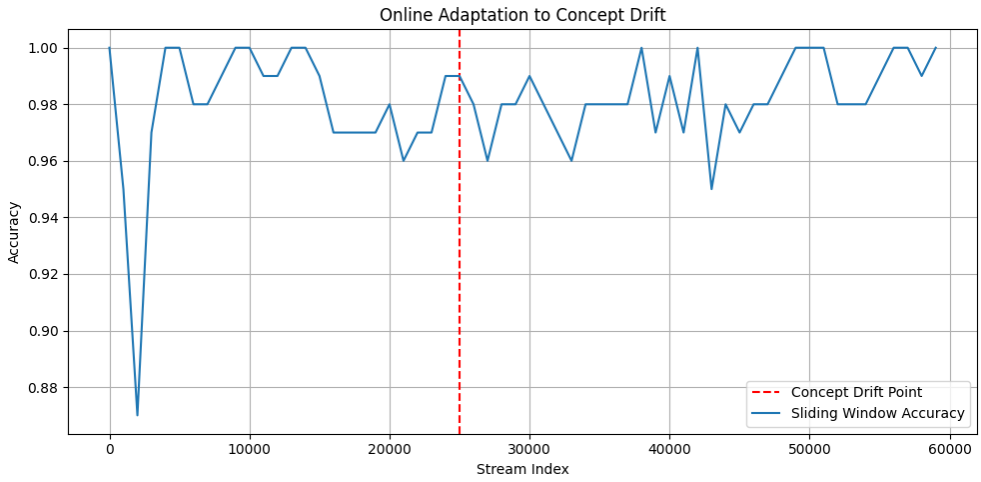


Figure 4.8: The performance of the model with reference to its adaptation to label drift on the SCADA dataset.

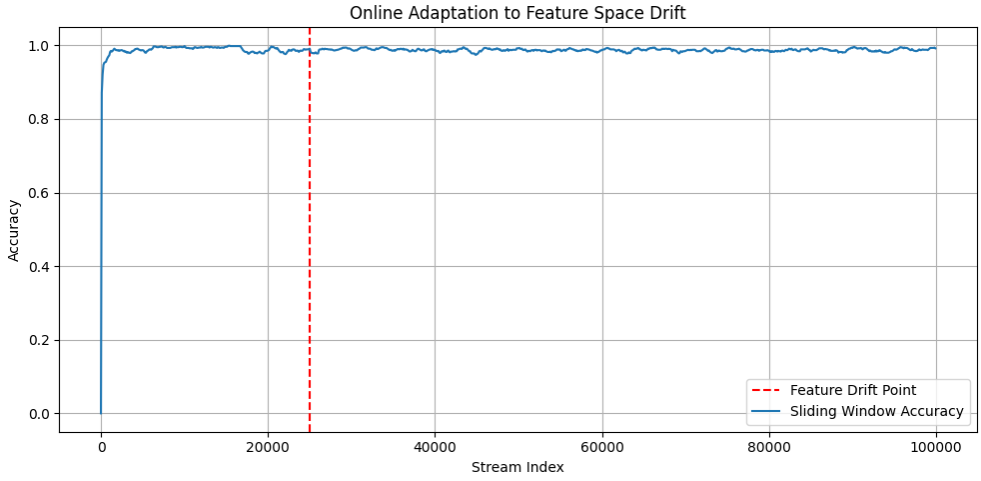


Figure 4.9: The performance of the model with reference to its adaptation to feature drift on the SCADA dataset.

Across all four datasets, the model demonstrated great performance in the presence of adversarial perturbations and noisy data by maintaining comparable accuracy

under noise and adversarial conditions. For example, on the NF-BoT IoT dataset, the model recorded an accuracy of 0.9983 across clean, noisy, and adversarial data. Similarly, the prediction consistency for noisy and adversarial inputs is above 0.99 for all datasets.

Table 4.4: A performance on the proposed system against adversarial perturbation and noisy data. Accuracy 1 represents the accuracy of the model on clean data only, accuracy 2 represents the accuracy of the model on the data samples with Gaussian noise, and accuracy 3 represents the accuracy of the model on the data samples with adversarial samples. Prediction 1 is the prediction consistency of the noisy data, and prediction 2 is the prediction consistency of the adversarial samples.

Accuracy 1	Accuracy 2	Accuracy 3	Prediction 1	Prediction 2
NF BoT IoT				
0.9983	0.9983	0.9983	1.00	1.00
NF ToN IoT				
0.8869	0.8869	0.8864	0.9989	0.9992
EdgeIIoT				
0.9046	0.9045	0.9047	0.9997	0.9999
SCADA				
0.9933	0.9933	0.9933	1.00	1.00

4.5.0.5 Scalability Analysis

The results of the scalability analysis of our proposed system are shown in Table 4.5. The model maintained a stable throughput and a low average latency across all datasets. On the NF-BoT IoT dataset, the system recorded a processing rate from 139.45 to 140.09 samples per second for up to 5,000 samples, recording a latency increase to 8.15 ms at 10,000 samples. On the SCADA dataset, the model recorded a processing rate up to 167.96 samples per second with a latency of 5.90 ms. A similar stability performance was observed when the system was evaluated on NF-ToN IoT and EdgeIIoT datasets. This shows that the proposed system shows scalability for real-time deployment in high-throughput environments.

Table 4.5: A scalability analysis of the proposed system on sample rates of 500, 1000, 5000, and 10000

Samples	**Time (s)**	**# samples/sec**	**Average Latency (ms)**
NF BoT IoT			
500	71.54	139.79	7.42
1000	71.71	139.45	7.09
5000	71.38	140.09	7.06
10000	82.41	121.34	8.15
NF ToN IoT			
500	72.57	137.80	7.17
1000	72.71	137.53	7.19
5000	72.73	137.50	7.19
10000	72.51	137.90	7.17
EdgeIIoT			
500	73.45	136.15	7.26
1000	73.27	136.49	7.24
5000	73.78	135.53	7.29
10000	73.59	135.89	7.27
SCADA			
500	62.62	159.69	6.21
1000	62.47	160.8	6.19
5000	58.97	169.57	5.84
10000	59.54	167.96	5.90

4.6 Discussions

The experimental results obtained after evaluating our proposed model across the four datasets show the efficiency, robustness, and suitability in terms of deployment of the proposed IDS in IoT environments. The consistency of the model in terms of high accuracy, precision, recall, F1 score, and ROCAUC, even after quantization, confirms that applying dynamic quantization significantly reduces the model size without significantly impacting the accuracy negatively. The results support the applicability of deploying the proposed system on resource-constrained devices like an IoT gateway.

Additionally, the drift adaptability analysis showed that the model is highly adaptable to both concept and feature drifts. The rapid post-drift stabilization shown by the model is very crucial when it comes to deploying such models on real-world IoT systems due to the non-stationary and evolving network behaviors of the traffic generated by these IoT devices.

Evaluating the system on noisy data and adversarial samples showed the proposed model's ability to correctly predict both noisy data and adversarial samples to

a large extent, proving the robustness of the model against perturbation-based evasion strategies. With the prediction consistency over 0.99, the model demonstrates a reliable behavior under adversarial perturbation and noisy data.

The scalability experiment confirmed that the proposed system can handle data streams of varying rates with consistent throughput and low latency. Even at higher sample rates, the system recorded lower inference latencies, especially on the SCADA dataset, which recorded the lowest average latency values. The outcome of this experiment highlights the potential of the proposed system to detect intrusions in real-time across diverse IoT domains.

4.6.1 Limitations

The experimental results show that the proposed system maintains a comparable performance even after quantization, has the capability to adapt to drifts, and is robust even in the presence of adversarial samples. Despite these strengths, the proposed system presents some limitations.

1. **Limited Dataset:** The evaluation of the system was limited to four datasets from different IoT domains. These datasets do not fully represent all the variants and complexities that exist in heterogeneous real-world IoT networks.

2. **Adversarial Scope:** The adversarial evaluation adopted basic perturbation strategies. More complex attack vectors, such as adaptive adversaries and transfer-based attacks, were not explored. This is an area that will be considered for future work.

3. **Static Drift Simulation:** Concept and feature drifts were artificially simulated at fixed points. This approach may not fully reflect the continuous and unpredictable nature of real-world drifts in IoT network traffic and device behavior.

4.7 Conclusion and Future Work

In this study, we introduced a novel IDS designed for the IoT ecosystem built using a deep online classifier with dynamic quantization. The proposed technique also uses a confidence-aware learning control approach. The proposed system addresses the problems of adaptive learning under concept and feature drifts, efficient model

deployment, and robustness to adversarial and noisy data perturbations, which are challenges in IoT environments.

The core of the proposed system integrates a deep feature extractor and linear classifier, which are both optimized for online learning using a confidence-based window mechanism that controls learning based on prediction certainty. The confidence-aware learning is implemented using a threshold that dynamically suspends or resumes model updates. This technique reduces overfitting and computational overhead. Moreover, the use of a dynamic quantization approach using PyTorch's low-bit quantization significantly compresses the model under 2 KB while retaining the model's performance.

The results after a comprehensive experimental validation show that the proposed model records high performance in terms of accuracy, precision, recall, F1 score, and ROCAUC even after quantization. Quantization reduced the model size by over 96% with minimal latency, maintaining inference time under 1.5 ms. The results confirm the suitability of our proposed technique for deployment on resource-constrained edge devices in IoT environments.

Further experiments show that the proposed IDS model could effectively recover from both label and feature distribution drifts. Additionally, in evaluating adversarial robustness, the model maintained consistent predictions with prediction consistency on Gaussian noise and adversarial inputs. This characteristic is vital for real-world IoT systems, where attackers can use evasion tactics to bypass IDSs.

Scalability analysis showed a stable throughput and low latency across different input stream rates.

While the datasets used are well established in the IoT domain, broader evaluation across real-world industrial deployments is required to assess generalization. The adversarial evaluation adopted basic perturbation techniques, and more complex attack scenarios remain to be explored.

In future work, we intend to expand the scope of the adversarial evaluation by considering more complex attacks like adaptive adversaries and transfer-based attacks.

References

[1] B. Susilo, A. Muis, and R. F. Sari, "Intelligent intrusion detection system against various attacks based on a hybrid deep learning algorithm", *Sensors*, vol. 25, no. 2, p. 580, 2025.

[2] F. Cerasuolo, G. Bovenzi, D. Ciuonzo, and A. Pescapè, "Attack-adaptive network intrusion detection systems for iot networks through class incremental learning", *Computer Networks*, vol. 263, p. 111 228, 2025.

[3] F. Restuccia, S. D'oro, and T. Melodia, "Securing the internet of things in the age of machine learning and software-defined networking", *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4829–4842, 2018.

[4] Z. Wang, H. Chen, S. Yang, X. Luo, D. Li, and J. Wang, "A lightweight intrusion detection method for iot based on deep learning and dynamic quantization", *PeerJ Computer Science*, vol. 9, e1569, 2023.

[5] B. Jacob, S. Kligys, B. Chen, *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[6] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing", *IEEE network*, vol. 32, no. 1, pp. 96–101, 2018.

[7] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding", *arXiv preprint arXiv:1510.00149*, 2015.

[8] R. Chu, P. Jin, H. Qiao, and Q. Feng, "Intrusion detection in the iot data streams using concept drift localization", *AIMS mathematics*, vol. 9, no. 1, pp. 1535–1561, 2024.

[9] Q. Xiang, L. Zi, X. Cong, and Y. Wang, "Concept drift adaptation methods under the deep learning framework: A literature review", *Applied Sciences*, vol. 13, no. 11, p. 6515, 2023.

[10] Z. Yin, H. Chen, H. Ma, T. Hu, and L. Bai, "Caeaid: An incremental contrast learning-based intrusion detection framework for iot networks", *Computer Networks*, vol. 262, p. 111 161, 2025.

[11] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection", *arXiv preprint arXiv:1802.09089*, 2018.

[12] A. Tabassum, A. Erbad, A. Mohamed, and M. Guizani, "Privacy-preserving distributed ids using incremental learning for iot health systems", *IEEE Access*, vol. 9, pp. 14 271–14 283, 2021.

[13] H. Chen, Z. Wang, S. Yang, X. Luo, D. He, and S. Chan, "Intrusion detection using synaptic intelligent convolutional neural networks for dynamic internet of things environments", *Alexandria Engineering Journal*, vol. 111, pp. 78–91, 2025.

[14] Y. N. Soe, Y. Feng, P. I. Santosa, R. Hartanto, and K. Sakurai, "Machine learning-based iot-botnet attack detection with sequential architecture", *Sensors*, vol. 20, no. 16, p. 4372, 2020.

[15] A. Pektaş and T. Acarman, "Deep learning to detect botnet via network flow summaries", *Neural Computing and Applications*, vol. 31, no. 11, pp. 8021–8033, 2019.

[16] S. Manimurugan, S. Al-Mutairi, M. M. Aborokbah, N. Chilamkurti, S. Ganesan, and R. Patan, "Effective attack detection in internet of medical things smart environment using a deep belief neural network", *IEEE Access*, vol. 8, pp. 77 396–77 404, 2020.

[17] B. Alotaibi and M. Alotaibi, "A stacked deep learning approach for iot cyberattack detection", *Journal of Sensors*, vol. 2020, no. 1, p. 8 828 591, 2020.

[18] T. R. Hoens, R. Polikar, and N. V. Chawla, "Learning from streaming data with concept drift and imbalance: An overview", *Progress in Artificial Intelligence*, vol. 1, pp. 89–101, 2012.

[19] F. Jemili, K. Jouini, and O. Korbaa, "Intrusion detection based on concept drift detection and online incremental learning", *International Journal of Pervasive Computing and Communications*, vol. 21, no. 1, pp. 81–115, 2025.

[20] M. Soltani, K. Khajavi, M. Jafari Siavoshani, and A. H. Jahangir, "A multi-agent adaptive deep learning framework for online intrusion detection", *Cybersecurity*, vol. 7, no. 1, p. 9, 2024.

[21] M. A. Shyaa, Z. Zainol, R. Abdullah, M. Anbar, L. Alzubaidi, and J. Santamaría, "Enhanced intrusion detection with data stream classification and concept drift guided by the incremental learning genetic programming combiner", *Sensors*, vol. 23, no. 7, p. 3736, 2023.

[22] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation", *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[23] J. Leo and J. Kalita, "Incremental deep neural network learning using classification confidence thresholding", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 7706–7716, 2021.

[24] H. A. Gabbar, O. G. Adegboro, A. Chahid, and J. Ren, "Incremental learning-based algorithm for anomaly detection using computed tomography data", *Computation*, vol. 11, no. 7, p. 139, 2023.

[25] J. Montiel, M. Halford, S. M. Mastelini, *et al.*, "River: Machine learning for streaming data in python", *J. Mach. Learn. Res.*, vol. 22, no. 1, Jan. 2021, ISSN: 1532-4435.

[26] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets", *Mobile networks and applications*, pp. 1–14, 2022.

[27] M. Teixeira, M. Zolanvari, and R. Jain, *Wustl-iiot-2018*, 2020. DOI: 10.21227/kzgp-7t84. [Online]. Available: https://dx.doi.org/10.21227/kzgp-7t84.

[28] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning", *IEEE Access*, vol. 10, pp. 40 281–40 306, 2022.

[29] A. Noureddine, "Powerjoular and joularjx: Multi-platform software power monitoring tools", in *18th International Conference on Intelligent Environments (IE2022)*, Biarritz, France, Jun. 2022.

Chapter 5

Chapter 5 is published as:

Promise Ricardo Agbedanu, Shanchieh (Jay) Yang, Richard Musabe, Ignace Gatare, James Rwigema, "ALMANET: A hybrid online learning IDS for real-time IoT security," Egyptian Informatics Journal, Volume 31, 2025, 100764, ISSN 1110-8665, https://doi.org/10.1016/j.eij.2025.100764.

ALMANET: A Hybrid Online Learning IDS for Real-Time IoT Security

Authors: Promise Ricardo Agbedanu, Sanchieh Jay Yang, Richard Musabe, Ignace Gatare, James Rwigema

Abstract: Although some modern Intrusion Detection Systems (IDSs) for Internet of Things (IoT) have explored online machine learning (ML) approaches to build these IDSs, most IoT-based IDSs are designed using offline ML techniques. IDSs built with offline ML approaches cannot adapt to rapidly changing IoT network conditions. They need continuous retraining and require a lot of computational power. To address these limitations, we propose ALMANET (ALMA+NET), a hybrid intrusion detection approach combining Approximate Large Margin Algorithm (ALMA) with Stochastic Weight Averaging (SWA) and an online neural network (NET). ALMANET leverages the power of online learning, which updates models incrementally and allows real-time adaptation to evolving network traffic, making it suitable for IoT environments. We validated ALMANET on four benchmark datasets, namely, NF BoT IoT, NF ToN IoT, NF UNSW, and NF CSE 2018 datasets. We demonstrated the proposed technique's performance in terms of accuracy, recall, ROCAUC, and robustness against adversarial attacks. We compared the performance of ALMANET against RF, SVM, LR, and ALMA. ALMANET records up to 98.58% ROCAUC and demonstrates high throughput, low false positive rates, and efficient memory usage of 14.64 KB across all datasets, making it feasible for deployment on edge devices.

5.1 Introduction

Internet speed and computer processing capacity have served as accelerators for the evolution of computing during the last four decades. Computer technology has evolved from vacuum tubes to transistors, integrated circuits, and microprocessors, and now into connected things, also known as the Internet of Things (IoT). Refrigerators, electric meters, air conditioners, heaters, and heart pacemakers can now communicate with other internet-connected devices thanks to the IoT ecosystem [1]. As a result, we have witnessed smart devices widely utilized in various domains, including agriculture [2], medicine and healthcare [3], industry [4], energy [5], and even in our homes [6].

However, the IoT ecosystem is vulnerable to cyber-attacks due to its wide range of applications, limited computational resources, and heterogeneity [7]. Several solutions have been proposed to counteract these attacks. [8] identifies IoT security requirements such as authentication, confidentiality, integrity, and secure architecture. The approaches stated by [8] are insufficient for preventing attacks in IoT environments, prompting the use of other techniques, such as intrusion detection systems (IDS). However, most IDS proposed for IoT systems use offline machine learning approaches, which add computational overhead to IoT systems. Aside from these computational overheads, IDSs designed using these techniques have limitations, such as the inability to adjust to changes in network traffic in real-time and the necessity to constantly update these models by retraining them.

According to [9], [10], online machine learning (ML) allows for continuous learning and adaptation to new threats in real-time, without manual retraining. This method improves the detection of novel and sophisticated attacks. Furthermore, online machine learning or streaming data techniques can be applied to systems with computational constraints. By online machine learning, we mean algorithms that process data one sample at a time.

Online ML algorithms like the Approximate Large Margin Algorithm (ALMA) proposed by [11] can classify data points into distinct categories by finding a hyperplane that maximizes the margin between these categories. The incremental nature of ALMA allows it to process data samples one at a time and update the model using each new sample, making it a good candidate for our proposed hybrid model.

This work proposes a hybrid online ML algorithm combining ALMA with SWA and an incremental neural network. The proposed algorithm is called ALMANET. We hypothesize that ALMANET can achieve superior accuracy when detecting network intrusions in IoT systems while maintaining low latency. This situation will enhance the model's ability to promptly identify and respond to attacks. The contributions of this work are as follows:

- We propose ALMANET, a hybrid online classifier that seamlessly integrates an online margin-based algorithm called ALMA, SWA, and an incremental neural network. This fusion allows for leveraging the rapid adaptability of online algorithms with the powerful pattern recognition capabilities of neural networks, providing a versatile tool for detecting network intrusions in dynamic and real-time environments.
- ALMA with SWA enforces a large-margin boundary. It updates aggressively when the margin falls below a threshold, making it well-suited to evolving data streams, and the neural network captures non-linear patterns and higher-order interactions. The model then averages these two outputs to get the final prediction, updating each component using one data sample at a time.
- Many hybrid methods can be heavy when they process data in batches. By comparison, this method is streamlined for single instance updates, making it more practical for real-time scenarios like intrusion detection for IoT environments.
- Because both ALMA and the incremental neural network adjust parameters after each new instance, the proposed approach can inherently handle adaptiveness within IoT data. By combining a linear margin-based algorithm and a non-linear learner, it can adapt to both subtle and dramatic shifts over time.

Experimental results show ALMANET recording more than 93% ROCAUC on all datasets and performs better than ALMA and Logistic Regression (LR). Although Random Forest (RF) and Support Vector Machines (SVM) performed slightly better than our proposed model in terms of accuracy, recall, and ROCAUC, our model recorded significantly lower memory usage (14.64 KB). The low memory usage makes our proposed model ideal for resource-constrained environments.

The rest of the paper is organized as follows: Section 2 is related work. Our proposed system is presented in Section 3. Section 4 focuses on the experimental

design and validation of the proposed system, with its results presented in Section 5, discussions in Section 6, and the study concluded in Section 7.

5.2 Related Work

Several traditional ML approaches have been employed to address security issues in IoT systems. SVMs have been utilized in different fields to detect unusual network activity. [12] utilized SVM models to detect intrusions in low-power and short-range networks. Davahli et al. [13] suggested a lightweight IDS using SVM, genetic algorithm, and grey wolf optimizer. The authors reported that the hybrid technique outperformed the individual technique. Decision trees have also been shown to be useful anomaly detection algorithms, particularly when combined with other techniques. Douiba et al. [14] proposed a better IDS for IoT systems using gradient boosting and decision trees. In related work, [15] employed a decision tree with improved data quality to produce reliable intrusion detection. Balhareth et al. [16] presented a tree-based IDS for safeguarding the Internet of Medical Things (IoMTs). The tree-based approach combines filter-based feature selection techniques, including mutual information and XGBoost. To prevent malicious usage of IoT systems, [17] proposed a distributed-based architecture that uses ensemble learning techniques comprising XGBoost, KNN, and Gaussian Naive Bayes as the first-level learners. The proposed design then employs the Random Forest algorithm for the final classification task. The proposed strategy outperformed the state-of-the-art methods described in the paper.

Over the years, deep learning-based methods have been explored to solve security problems in IoT systems. According to [18], security becomes more critical as the number of interconnected devices increases. According to [18], IDS lacks optimal feature learning and dataset management, so they proposed a system that combines spider monkey optimization and a stack-deep polynomial network to achieve optimal detection. [19] designed an IDS for the IoT environments using a convolutional neural network model in 1D, 2D, and 3D. To address the problem of imbalanced datasets, [20] proposed an intelligent detection system using singular value decomposition to reduce features while using synthetic minority over-sampling to avoid over and under fitting. A hybrid rule-based feature selection and deep feed-forward neural network model was proposed by [21] to train and

verify information captured from TCP/IP packets. [22] also used a feed-forward neural network with embedding layers to model a multiclass-based IDS and then used transfer learning to encode high-dimensional categorical features to build a binary classifier using a second feed-forward neural network model. DNNs have shown promising results in detecting network intrusions. Thriving on this benefit, [23] used a DNN technique to identify cyber-attacks in IoT systems. In a related study, [24] also used a DNN-based algorithm to model an anomaly detection system for IoT systems, which showed superior accuracy. However, when it comes to the IoT ecosystem, parameters such as model training time, speed of detection, and the amount of computational resources consumed by an IDS are equally important as accuracy, recall, and F1 score. The study would have been more feasible in IoT environments if the authors had reported on their proposed model's training time or how it impacts the computational resources of IoT systems. [25] proposed a technique based on principal component analysis (PCA) to detect drifts in IoT network traffic. In the same work, the author proposed an online DNN that adjusts itself by dynamically changing the sizes of its hidden layers. The adjustment is based on the hedge weighting mechanism. This mechanism allows the model to learn and adapt as new intrusion data is encountered. Modifying the hidden layer sizes enables the DNN model to respond effectively to the changes in the intrusion detection environment. [26] proposed a hybrid deep random neural network for detecting cyber-attacks in Industrial Internet of Things (IIoT) systems.

Intrusion detection in IoT networks is increasingly shifting toward online and incremental learning, an approach designed to handle the dynamic environment and resource constraints of the IoT ecosystem. Traditional IDS often requires retraining on accumulated data, which is infeasible for IoT devices that face evolving attacks and limited computational power. Recent works address this gap by enabling continuous model updates. Constantinides et al. proposed an incremental IDS using a Self-Organizing Incremental Neural Network with an SVM classifier that forgoes signature-based detection and can adapt in real-time to known and new attacks [27]. Similarly, Martina and Foresti developed a continuous learning IDS (SF-SOINN) that incrementally learns from new traffic without frequent offline retraining, achieving fast, noise-robust classification on benchmark datasets like NSL-KDD and CIC-IDS2017 [28]. Nixon et al. [29] developed an online machine learning-based architecture to identify intrusions in IoT environments, addressing

the topic of resource efficiency. These works demonstrated that online learning can maintain high detection accuracy over time.

Other works combine offline and online learning techniques to balance performance and adaptability. Abderrahim and Benosman overcame the rigidity of offline-based IDS by combining a LightGBM trained on historical data with a Passive-Aggressive online classifier that adapts to new attacks without having to be retrained [30]. The proposed approach captures diverse attack patterns with offline training while staying responsive to emerging threats through incremental updates. Experiments on CICIoT2023 show that this hybrid approach can maintain strong accuracy as attack distributions evolve [30]. On the other hand, our technique avoids needing a separate offline phase by using an online algorithm throughout, simplifying its deployment on resource-constrained devices.

Beyond improving adaptability and efficiency, researchers have looked at enhancing IDS's generalization and robustness. One technique is model parameter averaging. Stochastic Weight Averaging (SWA) was shown by Izmailov et al. [31] to improve neural network generalization by averaging weights over training epochs, effectively ensembling models without added runtime cost [32]. As we have done in our work (ALMANET), incorporating such an approach can reduce overfitting to specific traffic conditions, which is crucial for IoT scenarios with non-stationary data.

Another challenge is adversarial attacks against IDS. Vitorino et al. [33], [34] demonstrated that even high-performance classifiers, such as tree ensembles like XGBoost, are vulnerable to adversarial evasion. They highlighted that adversarial training significantly boosts the robustness of IDS models [33], [34]. The above challenge underscores the importance of evaluating IDS against adversarial attacks. Our work aligns with this proposal by evaluating ALMANET's resilience to adversarial inputs and showing improved robustness.

5.3 Proposed Methodology

In this section, we propose a hybrid online classifier called ALMANET. ALMANET combines an adaptive large margin algorithm (ALMA) with a lightweight neural network. We incorporate Stochastic Weight Averaging (SWA) to stabilize ALMA's learning. The final prediction is obtained by ensemble averaging the prob-

abilistic outputs of both components. The framework of ALMANET is shown in Figure 5.1.

5.3.1 ALMA with Stochastic Weight Averaging

Let $\mathbf{x}_t \in \mathbb{R}^d$ be the feature vector and $y_t \in \{-1, +1\}$ be the label at time step t, where +1 denotes the positive class and -1 is the negative class. The ALMA algorithm updates a weight vector $\mathbf{w} \in \mathbb{R}^d$ when the margin constraint is violated:

$$y \cdot \langle \mathbf{w}, x \rangle < (1 - \alpha)\gamma_t \tag{5.1}$$

where:
$\alpha \in (0, 1)$ is the margin relaxation parameter,
$\gamma_t = \frac{B\sqrt{p-1}}{\sqrt{t}}$ is a shrinking margin threshold,
p is the norm order,
B, C are scaling constants,
t is the number of updates.

When the constraint is violated, the weights are updated using:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta_t \cdot y \cdot x \quad \text{with} \quad \eta_t = \frac{C}{\sqrt{(p-1)t}} \tag{5.2}$$

The weights are then projected to ensure $\|\mathbf{w}_{t+1}\|_p \leq 1$. To stabilize training, we apply Stochastic Weight Averaging (SWA):

$$\bar{\mathbf{w}}_t = \frac{1}{n} \sum_{i=1}^{n} \mathbf{w}_{t_i} \tag{5.3}$$

where $\bar{\mathbf{w}}_t$ is the average of past weight vectors collected every k steps after a warm-up period $t_{\text{swa\_start}}$.

The probabilistic prediction is computed using the sigmoid function:

$$P_{\text{ALMA}}(y = 1|x) = \sigma(\langle \bar{\mathbf{w}}, x \rangle) \tag{5.4}$$

5.3.2 Neural Network Classifier

We use a single hidden-layer feed-forward neural network (FNN):

$$z_1 = \text{ReLU}(W_1 x + b_1), \quad \hat{y}_{\text{NN}} = \sigma(W_2 z_1 + b_2) \tag{5.5}$$

Where:

$W_1 \in \mathbb{R}^{h \times d}, b_1 \in \mathbb{R}^h$ are the input to hidden weights and biases,

$W_2 \in \mathbb{R}^{1 \times h}, b_2 \in \mathbb{R}$ are the hidden to output weights and biases,

σ is the sigmoid function, and

h is the number of hidden units.

The neural network is trained online via backpropagation using the binary cross-entropy loss:

$$\mathcal{L}_{\text{NN}} = -y \log \hat{y}_{\text{NN}} - (1 - y) \log(1 - \hat{y}_{\text{NN}}) \tag{5.6}$$

5.3.3 Hybrid Ensemble Prediction

The final prediction probability is a convex combination of both classifiers:

$$P(y = 1|x) = \lambda \cdot P_{\text{ALMA}}(y = 1|x) + (1 - \lambda) \cdot \hat{y}_{\text{NN}} \tag{5.7}$$

Where $\lambda \in [0, 1]$ is a tunable ensemble weight controlling the trade-off between the ALMA and neural network predictions.

The pseudocode of our proposed algorithm is shown in Algorithm 2.

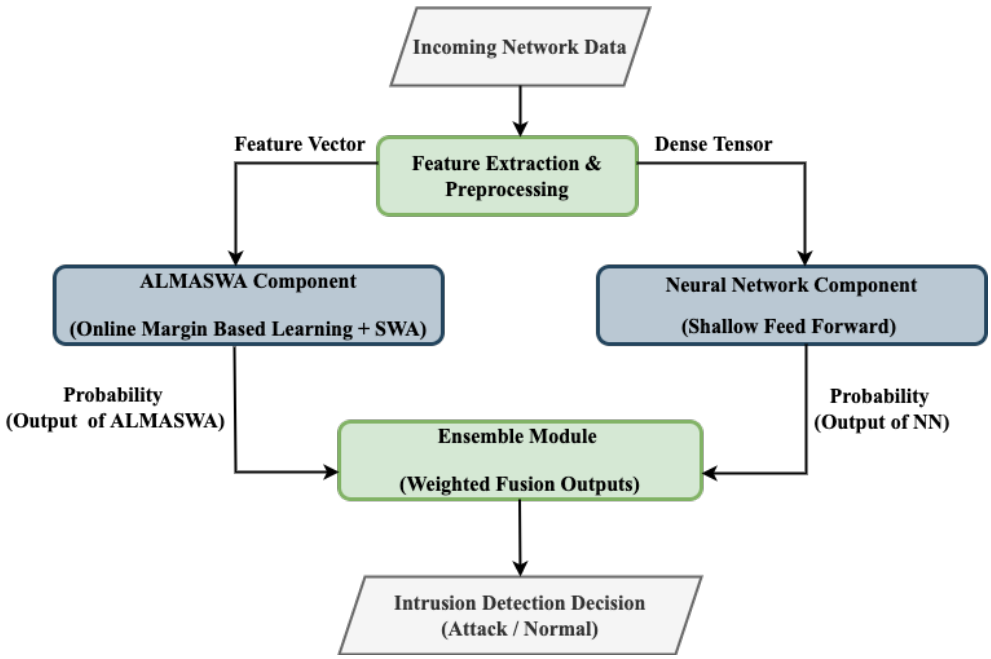


Figure 5.1: Our proposed network intrusion detection framework based on ALMANET

Algorithm 2 ALMANET Classifier

Require: Feature vector $x \in \mathbb{R}^d$, label $y \in \{0, 1\}$, feature index map $\mathcal{F}$
Require: ALMA parameters: α, B, C, p, SWA start step t_{swa}, frequency f_{swa}
Require: Neural network weights (W_1, b_1, W_2, b_2), learning rate η
Require: Ensemble weight $\lambda \in [0, 1]$
Initialize ALMA weights $\mathbf{w} \leftarrow \mathbf{0}$, averaged weights $\bar{\mathbf{w}} \leftarrow \mathbf{0}$
Initialize step counter $t \leftarrow 1$, SWA counter $n \leftarrow 0$
while receiving input (x, y) **do**
  Convert label: $y' \leftarrow 2y - 1$ $\{y' \in \{-1, +1\}\}$
  Compute margin: $\gamma_t \leftarrow \frac{B\sqrt{p-1}}{\sqrt{t}}$
  if $y' \cdot \langle \mathbf{w}, x \rangle < (1 - \alpha) \cdot \gamma_t$ **then**
    $\eta_t \leftarrow \frac{C}{\sqrt{(p-1)t}}$
    for each feature i in x **do**
      $w_i \leftarrow w_i + \eta_t \cdot y' \cdot x_i$
    end for
    if $\|\mathbf{w}\|_p > 1$ **then**
      Project $\mathbf{w} \leftarrow \frac{\mathbf{w}}{\|\mathbf{w}\|_p}$
    end if
  end if
  if $t \geq t_{\text{swa}}$ **and** $(t - t_{\text{swa}}) \bmod f_{\text{swa}} = 0$ **then**
    $\bar{\mathbf{w}} \leftarrow \frac{n \cdot \bar{\mathbf{w}} + \mathbf{w}}{n+1}$
    $n \leftarrow n + 1$
  end if
  Compute ALMA probability: $P_{\text{ALMA}} \leftarrow \sigma(\langle \bar{\mathbf{w}}, x \rangle)$
  Convert x to tensor x_{tensor} using $\mathcal{F}$
  Forward pass: $z \leftarrow \text{ReLU}(W_1 x_{\text{tensor}} + b_1)$
  $\hat{y}_{\text{NN}} \leftarrow \sigma(W_2 z + b_2)$
  Compute loss: $\mathcal{L} \leftarrow -y \log \hat{y}_{\text{NN}} - (1 - y) \log(1 - \hat{y}_{\text{NN}})$
  Update neural network parameters using gradient descent with learning rate η
  Compute final probability:
  $P(y = 1 \mid x) \leftarrow \lambda \cdot P_{\text{ALMA}} + (1 - \lambda) \cdot \hat{y}_{\text{NN}}$
  $t \leftarrow t + 1$
end while

5.3.4 Justification for choice of hyperparameters

We chose the hyperparameters in Table 5.1 to validate our proposed algorithm. The values for α, B, C, and p were chosen because these are the default parameters used in the original ALMA algorithm [11]. The values for SWA_{start} and SWA_{freq} were chosen because our experiments showed that the model takes about 100 data samples to exhibit stability in its prediction probability. Using a SWA_{freq} of 10 after starting SWA_{start} ensures a balance between responsiveness and computational efficiency. Updating SWA parameters based on every sample is not computationally efficient. We chose a hidden dimension of 64 for the neural network because moderate-sized hidden layers between 32-128 neurons are commonly used when developing IoT-based IDS [35], [36]. Choosing a hidden dimension of 64 balances

accuracy and computational usage. The list of hyperparameters used in this study is shown in Table 5.1.

Table 5.1: Hyperparameters used in this work

Parameter	**Value**
α	0.9
B	$\frac{1}{0.9}$
C	$2^{0.5}$
p	2
SWA start	100
SWA freq	10
hidden layer	64
Learning rate	0.001

5.4 Experimental Setup and Evaluation

The experiments were conducted on a MacBook Pro with an M1 chip running on 16 GB of RAM with a disk storage space of 1 TB. We used the River [37] streaming library that supports online ML and real-time learning to model our proposed detection system.

5.4.1 Datasets

Four datasets were used to validate the proposed algorithm. The datasets used are NF BoT IoT, NF CSE 2018, NF UNSW NB15, and NF ToN IoT datasets [38]. These datasets were selected for their representation of diverse IoT attack scenarios. The number of samples in the NF BoT IoT dataset contains 540 (0.36%) benign samples. The NF CSE 2018 dataset has 66540 (88.04%) benign samples. Similarly, the NF UNSW NB15 dataset has 91809 (96.09%) samples. On the other hand, the NF ToN IoT dataset has 22266 (35.81%) samples. Table 5.2 summarizes the four datasets used to validate our proposed model.

Table 5.2: A summary of the number of samples and features of the four datasets we used to validate the proposed algorithm

Dataset	**#Samples**	**#Benign**	**#Features**
NF BoT	151054	540 (0.36%)	45
NF CSE	75575	66540 (88.04%)	45
NF UNSW	95611	91809 (96.09%)	45
NF ToN	67762	24266 (35.81%)	45

5.4.2 Evaluation Metrics

Evaluation metrics such as accuracy, recall, and ROCAUC were chosen to assess the system's ability to distinguish between attack and benign traffic correctly. In online learning, the performance of each metric is tracked over time. Like online ML algorithms, these metrics are incremental; they don't need to store the full dataset. They only track aggregated counts. We used these metrics to evaluate the proposed algorithm because of the imbalanced nature of the datasets used for our experimental validation. We also used metrics such as processing time, throughput, CPU usage, latency, and memory usage to evaluate the proposed model.

5.4.3 Experimental Validation

5.4.3.1 Baseline Performance of Hybrid ALMASWA with Neural Network

The rationale of this experiment was to evaluate and compare the performance of Random Forest (RF), Support Vector Machines (SVM), Logistic Regression (LR), and ALMA against our proposed ALMANET in detecting attacks. The methodology uses a purely online framework, where data is processed one sample at a time. Each data sample is predicted, and then the model learns from it after the prediction. The full datasets were presented to the model in a randomised approach. Performance is assessed based on accuracy, recall, ROCAUC, and memory usage, which provide a quantitative measure of the models' detection capabilities and adoption for use in an IoT environment.

5.4.3.2 Measuring the True Positive Rate (TPR) and False Positive Rate (FPR) of the proposed algorithm

In the second experiment, we compared the performance of ALMA and ALMANET by tracking TPR and FPR. The TPR and FPR are calculated within a fixed-length window to capture how performance evolves as new data arrives. The objective is to evaluate each algorithm's ability to adapt over time while maintaining reliable detection, which is crucial in real-time IoT environments.

5.4.3.3 Evaluating the adversarial robustness of the proposed model

In this experiment, a continuously trained model is evaluated under adversarial conditions to evaluate how resilient the model remains when presented with deliberately manipulated inputs. The objective is to simulate scenarios where an

attacker attempts to compromise detection systems in real-time, highlighting the need for adaptive defense mechanisms. We implemented a Fast Gradient Sign Method (FGSM) style attack adapted to the streaming and online learning context.

We adopted a targeted evasion attack that aims to reduce the model's probability of detecting an attack by flipping predictions from true (attack) to False (benign).

Let the input be a feature vector $\mathbf{x} \in \mathbb{R}^d$, and let $f_\theta(\mathbf{x}) \in [0, 1]$ be the model's predicted probability of the positive class ("Attack"). We craft an adversarial example $\mathbf{x}_{\text{adv}}$ by perturbing the input in the direction of the gradient of the loss, as follows:

$$\mathbf{x}_{\text{adv}} = \mathbf{x} - \epsilon \cdot \text{sign}\left(\nabla_{\mathbf{x}} \mathcal{L}(f_\theta(\mathbf{x}), y)\right) \tag{5.8}$$

Where:
ϵ is the perturbation magnitude ($\epsilon \in [0.1, 0.3]$)
$\mathcal{L}$ is the model output interpreted as a loss signal
sign denotes the element of the sign function.

All features in the input vector were perturbed uniformly. We did not apply feature selection or masking. The adversarial perturbation is applied in an online manner, meaning that it is computed on-the-fly for each incoming sample, after the model's clean prediction and update.

5.4.3.4 Drift Evaluation

To assess the robustness of ALMANET under non-stationary streaming conditions, we simulate and evaluate its performance under four types of synthetic label drift scenarios. The drifts considered are gradual, sudden, recurring, and incremental. Each scenario introduces changes to the label distribution in a controlled manner, while the input features remain unchanged. We use the same original dataset and synthetically manipulate the labels y to emulate concept drift without introducing structural noise to the features X. During the gradual drift, the drift is introduced starting at 25% of the stream and gradually increases over the next 25% of samples. During this interval, labels are probabilistically flipped with increasing intensity. When it comes to sudden drift, at the halfway point in the stream (50%), a sudden shift is introduced, flipping 20% of labels instantly. For recurring drift, label flips occur at two known drift points, 20% and 60%, reverting between original and altered label distributions. The incremental drift starts at 33% of the stream and

ends at 66%, with the label flipping probability increasing linearly across that interval.

5.4.3.5 Evaluating the latency and throughput of ALMANET

We measured the real-time performance of a continuously updated model by monitoring both inference time per sample and overall throughput in a streaming environment. This experiment assesses how efficiently the model can generate predictions and update itself on incoming data without substantial latency. This experiment is important for real-world applications like IoT, where timely responses to new data are critical. The results are evaluated by analyzing the timing measurements for individual predictions and computing how many samples can be processed per second.

5.4.3.6 Evaluating scalability with increasing data

We evaluated how the training process scales when only portions of the dataset are used, exploring whether the model can maintain acceptable performance while processing differing amounts of data. The core idea behind this approach is to observe trade-offs in accuracy and time consumption as the dataset size changes. Scalability is important for applications where computational resources are limited or data volume can fluctuate, ensuring the model's efficiency remains viable under different loading conditions. Performance is assessed by measuring the model's final accuracy and the time taken to train on each sampled fraction of the dataset.

5.4.3.7 Model Efficiency and Resource Utilization

To assess the algorithm's ability to handle evolving data distributions in resource-constrained environments, we used an incremental approach where the model is updated continuously as new data arrives in small batches. The evaluation metrics are accuracy, time, and memory, as the model processes chunks sequentially. Resource utilization is to provide an overview of the trade-offs between accuracy and computational resource usage.

5.5 Results

5.5.1 Baseline Performance of ALMA and ALMANET

Table 5.3 presents the comparative performance of RF, SVM, LR, and ALMA against our proposed ALMANET on the four datasets. ALMANET consistently outperforms ALMA across all datasets regarding accuracy, recall, and ROCAUC. For instance, on the NF BoT IoT dataset, ALMA recorded 70.73% accuracy while ALMANET recorded an accuracy of 93.04%. On the other hand, ALMANET recorded slightly lower accuracy, recall, and ROCAUC against RF and SVM. Comparing the performance of ALMANET against LR shows ALMANET achieving a comparable performance when evaluated on the NF BoT IoT. However, evaluating ALMANET on the other datasets shows it performed better than LR. Regarding memory usage, except for ALMA, ALMANET recorded the lowest memory usage (14.64 KB). ALMANET achieves a good trade-off between detection and memory usage.

5.5.2 Measuring the True Positive Rate (TPR) and False Positive Rate of the proposed algorithm

Figures 5.2–5.5 provide rolling estimates of TPR and FPR over sample windows of size 10,000 for both ALAM and ALMANET(Hybrid) across the BoT IoT, CSE 2018, UNSW NB15, and ToN IoT datasets. In all cases, the proposed method's TPR starts high and quickly stabilizes around 90–95%, whereas ALMA's TPR generally remains lower. Moreover, our proposed approach consistently exhibits a lower FPR than ALMA over time, reaching near zero for extended segments of the data streams. These improvements are particularly evident in the BoT IoT dataset, where the proposed method maintains a near-perfect TPR while gradually reducing the FPR to almost 0%, indicating its robust detection capabilities in large-scale network traffic.

5.5.3 Evaluating the adversarial robustness of ALMANET

Table 5.4 summarizes the robustness of our hybrid method under varying levels of adversarial noise (ϵ) across the four datasets. Despite increasing adversarial perturbations from 0.1 to 0.3, the proposed model sustains high accuracy with

Table 5.3: Comparing the accuracy, recall, ROCAUC, and memory usage of RF, SVM, LR, and ALMA against ALMANET on the four datasets.

Algorithm	Accuracy	Recall	ROCAUC	Memory (KB)
NF BoT IoT				
RF	99.97%	99.99%	96.81%	545472
SVM	99.95%	99.97%	97.26%	690639.97
LR	99.88%	99.97%	87.71%	701471.95
ALMA	70.73%	70.75%	68.43%	3.89
ALMANET	99.49%	99.56%	93.04%	14.64
NF CSE				
RF	99.52%	96.07%	98.03%	782528
SVM	99.26%	94.17%	97.05%	820367.97
LR	97.90%	86.20%	92.84%	601392.03
ALMA	82.86%	85.72%	84.10%	3.89
ALMANET	98.57%	93.14%	96.22%	14.64
NF UNSW NB15				
RF	99.67%	98.11%	98.92%	349408.05
SVM	99.50%	99.05%	99.29%	597711.97
LR	99.09%	90.26%	94.85%	402032.03
ALMA	83.37%	95.55%	89.21%	3.89
ALMANET	99.11%	98.00%	98.58%	14.64
NF ToN				
RF	99.23%	99.46%	99.14%	283727.97
SVM	96.83%	98.51%	96.14%	523807.95
LR	81.42%	81.94%	81.21%	233712.03
ALMA	67.04%	66.99%	67.07%	3.89
ALMANET	94.88%	96.92%	94.07%	14.64

minimal performance degradation. For instance, in the NF BoT IoT dataset, clean accuracy hovers around 99.50%, and even the adversarial accuracy remains above 99.53%, leading to a flip rate of around 0.005% to 0.007%. Similar trends are evident in the other datasets (NF CSE 2018, NF UNSW NB15, and NF ToN), where the adversarial attacks fail to produce significant misclassifications, reflecting a consistently strong defense profile against injected noise.

5.5.4 Drift Evaluation

Table 5.5 presents the performance of ALMANET under four types of simulated concept drifts across the four datasets. Across all datasets, ALMANET demonstrates resilience to various drift types, with accuracy ranging from 85.91% to 94.74%, and F1 cores ranging from 85.90% to 93.90%. The incremental and recurring drifts typically result in the highest model performance, particularly in the BoT-IoT dataset, where ALMANET achieves 94.74% accuracy and an F1-score of 93.90%. Although the model performs competitively under sudden drifts, it is

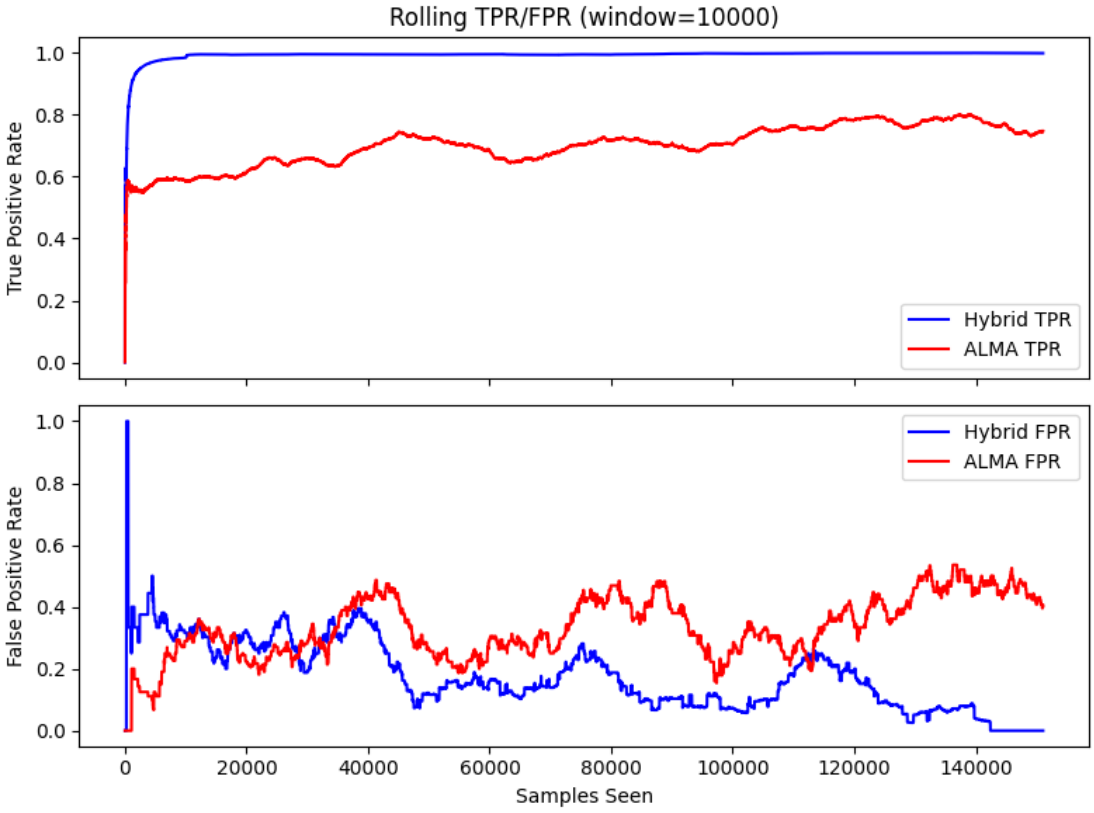


Figure 5.2: A visual representation of TPR and FPR of ALMA and ALMANET on the BoT IoT dataset

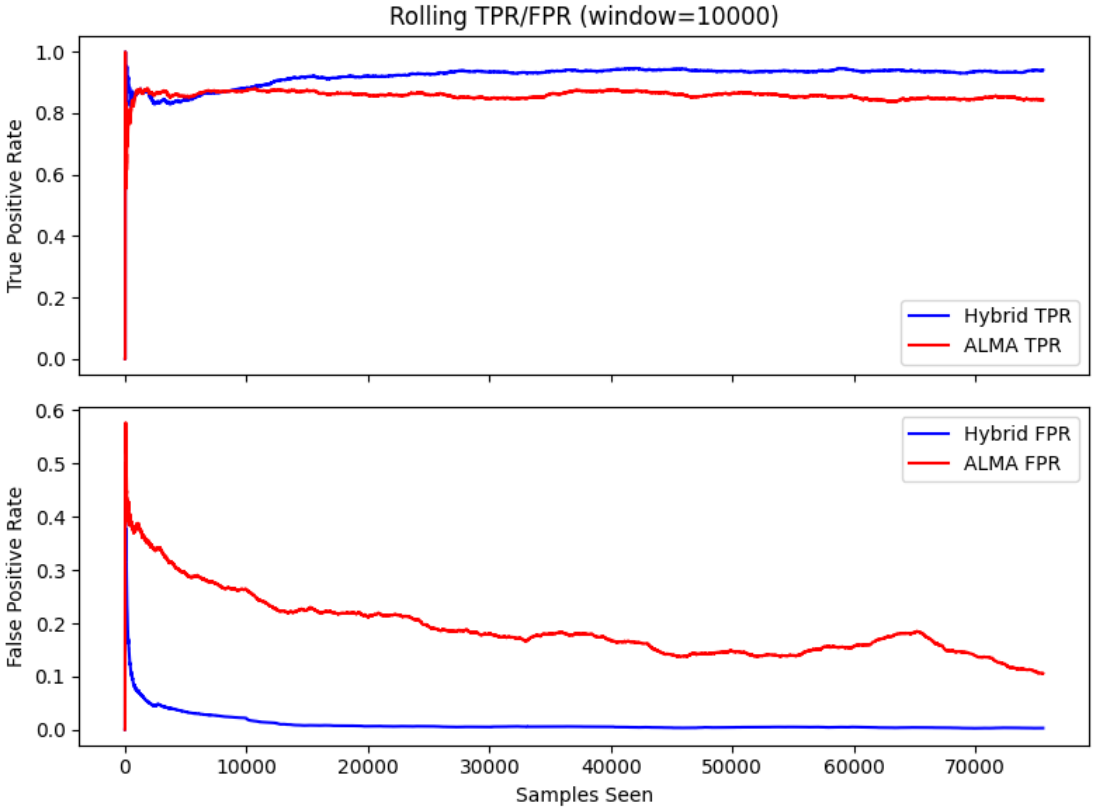


Figure 5.3: A visual representation of TPR and FPR of ALMA and ALMANET on the CSE 2018 dataset

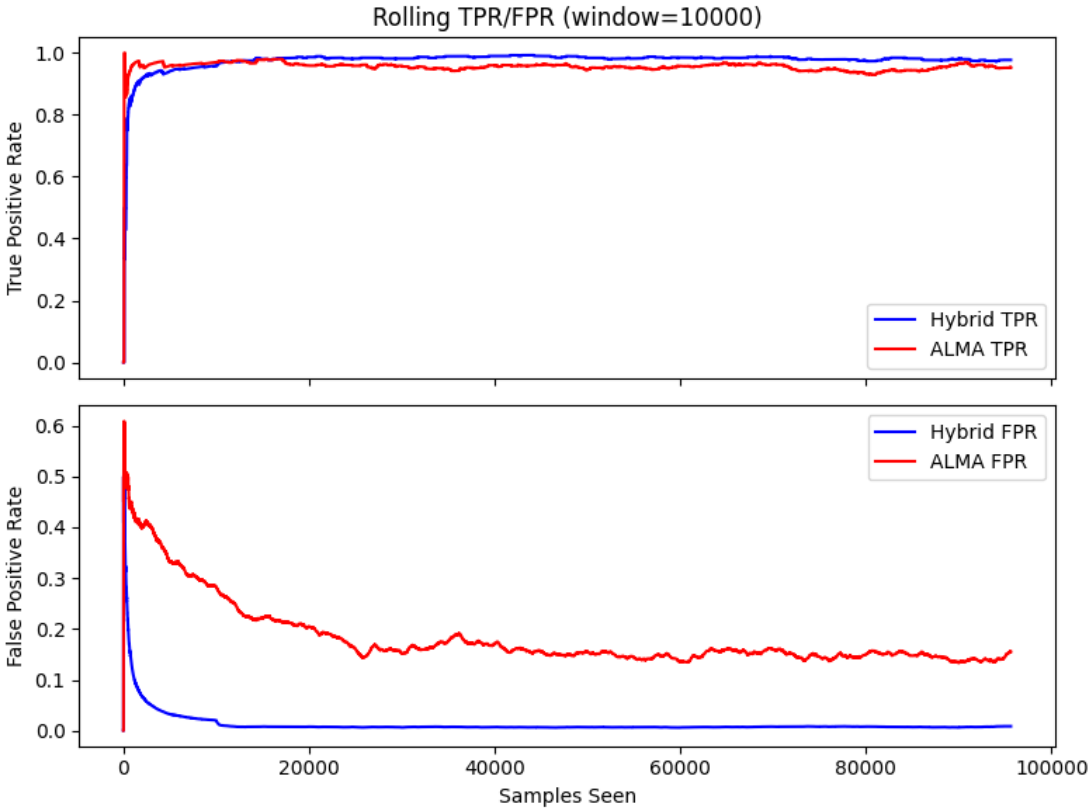


Figure 5.4: A visual representation of TPR and FPR of ALMA and ALMANET on the UNSW NB15 dataset

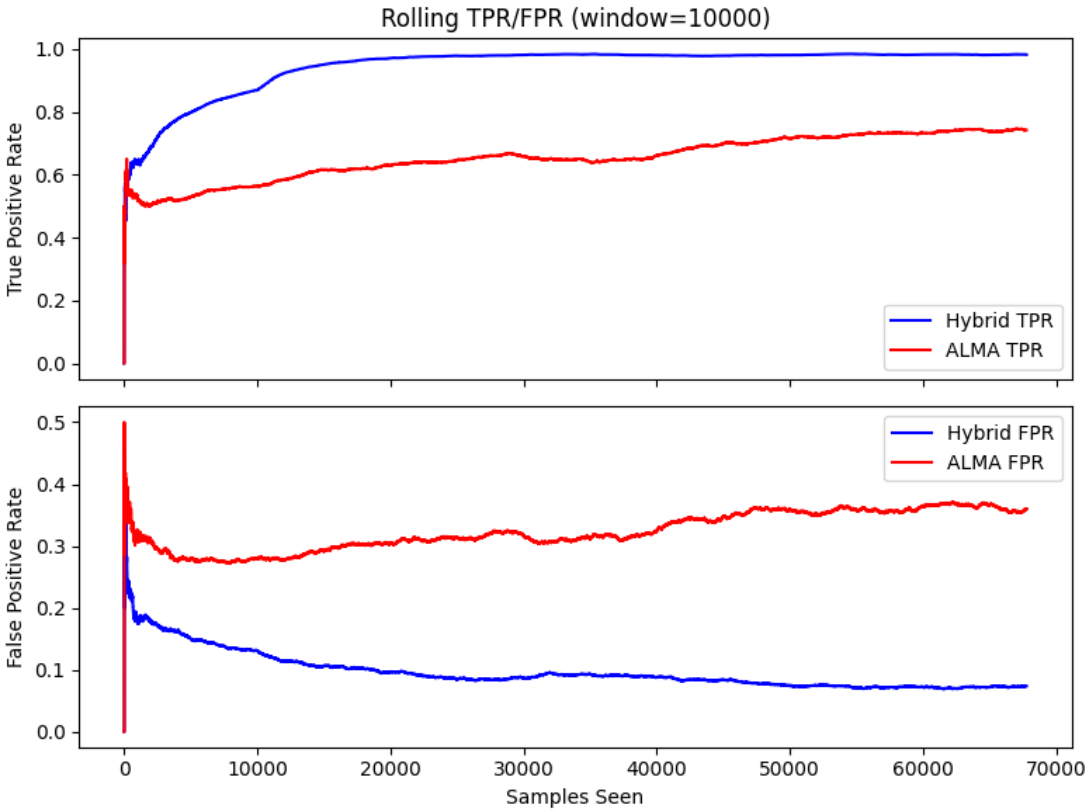


Figure 5.5: A visual representation of TPR and FPR of ALMA and ALMANET on the ToN IoT dataset

Table 5.4: Evaluating the adversarial robustness of the proposed ALMANET

Adversarial ϵ	**Clean Accuracy**	**Adversarial Accuracy**	**Successful Flips**
NF BoT IoT			
0.1	99.51%	99.54%	0.0053%
0.2	99.49%	99.53%	0.0066%
0.3	99.50%	99.54%	0.0053%
NF CSE 2018			
0.1	98.52%	98.61%	0.0622%
0.2	98.53%	98.63%	0.0701%
0.3	98.55%	98.65%	0.0701%
NF UNSW NB15			
0.1	99.12%	99.16%	0.0303%
0.2	99.02%	99.06%	0.0272%
0.3	99.11%	99.16%	0.0335%
NF ToN			
0.1	94.22%	94.51%	0.1491%
0.2	94.40%	94.79%	0.2155%
0.3	94.57%	94.93%	0.1845%

generally the most challenging form of concept drift based on the results. Recording an accuracy above 86% in all cases and exceeding 90% on BoT-IoT, CSE-CIC, and UNSW-NB15 datasets.

Table 5.5: Performance of ALMANET in the midst of gradual, sudden, recurring, and incremental drift

Type of Drift	**Accuracy**	**Precision**	**F1**
NF BoT IoT			
Gradual	93.13%	93.00%	93.06%
Sudden	91.40%	90.39%	90.30%
Recurring	94.14%	93.38%	93.37%
Incremental	94.74%	93.97%	93.90%
NF CSE 2018			
Gradual	91.29%	91.10%	91.16%
Sudden	90.23%	89.89%	89.72%
Recurring	93.04%	92.83%	92.73%
Incremental	93.74%	93.57%	93.46%
NF UNSW NB15			
Gradual	92.16%	91.98%	92.06%
Sudden	90.73%	89.98%	0.89.97%
Recurring	93.46%	92.95%	93.00%
Incremental	94.11%	93.64%	93.67%
NF ToN			
Gradual	85.91%	85.89%	85.90%
Sudden	86.58%	86.51%	86.51%
Recurring	89.88%	89.85%	89.94%
Incremental	90.97%	90.94%	90.92%

5.5.5 Evaluating the latency and throughput of the model

Figures 5.6, 5.7, 5.8, and 5.9 illustrate the real-time inference latency and throughput rates for our hybrid model across the four datasets. Overall, the mean inference time per sample remains low, with most measurements hovering below 0.05 seconds, although a few spikes reaching 0.3 to 0.4 seconds are occasionally observed. Meanwhile, throughput stabilizes between 80 and 120 samples per second after an initial warm-up phase, indicating the model's ability to handle large traffic volumes in real-time. The highest sustained throughput is achieved in the BoT IoT dataset, exceeding 100 samples per second over extended time frames, while the other datasets show only modest drops in processing speed.

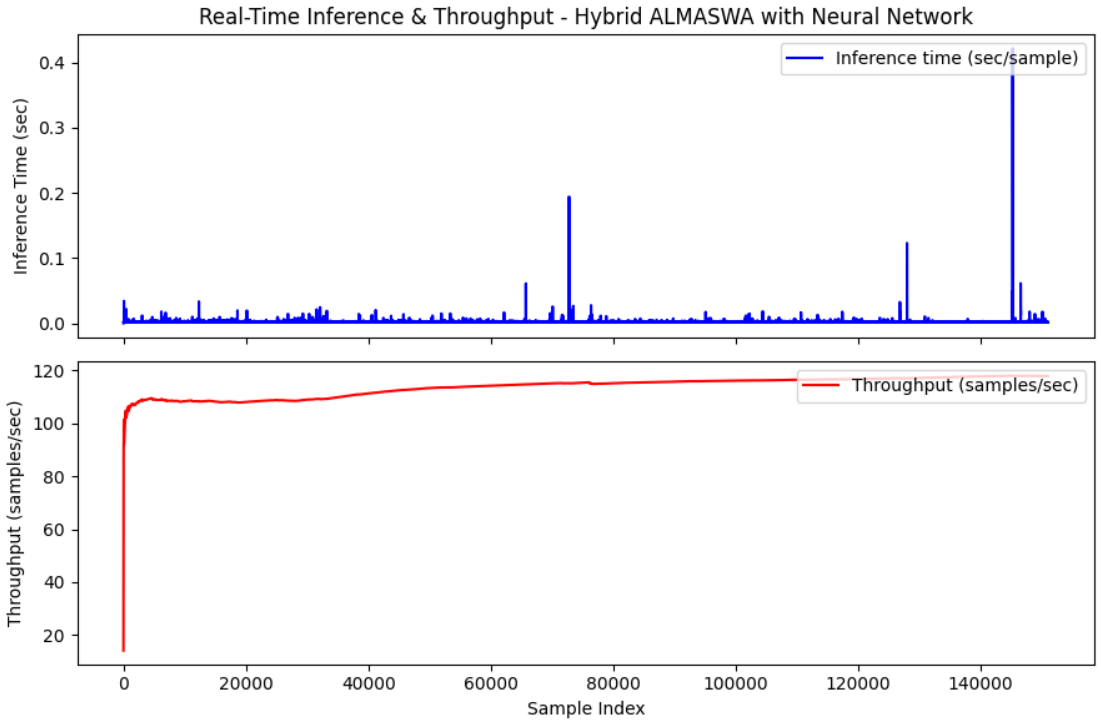


Figure 5.6: The throughput and latency of the proposed model evaluated with the BoT IoT dataset

5.5.6 Evaluating scalability with increasing data

Table 5.6 outlines the proposed model's continuous accuracy and the total processing time as the fraction of each dataset increases from 0.1 up to the full dataset (1.0). For instance, in the NF BoT IoT dataset, accuracy improves from 98.93% at 15,105 samples to 99.52% at 151,054 samples, with processing time growing from 292.76 seconds to 2728.34 seconds. A similar pattern emerges in the other datasets: accuracy rises steadily with more data, peaking at over 98% in NF CSE 2018 and NF UNSW NB15 and nearly 95% in NF ToN. Despite the increasing computational

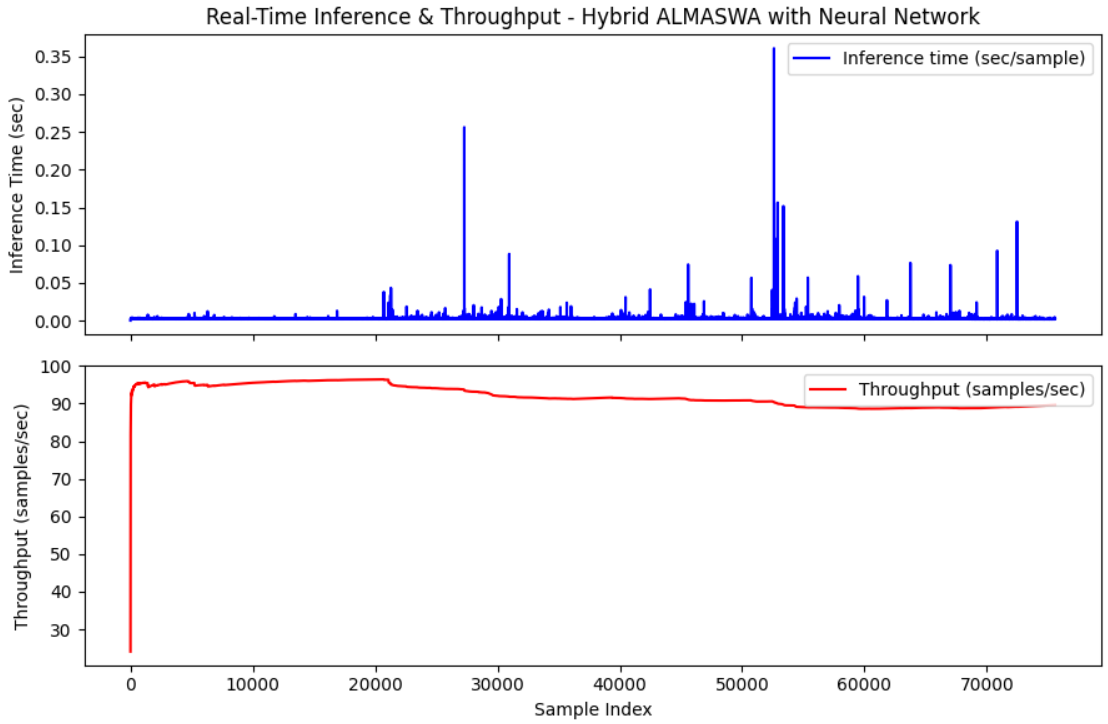


Figure 5.7: The throughput and latency of the proposed model evaluated with the CSE 2018 dataset

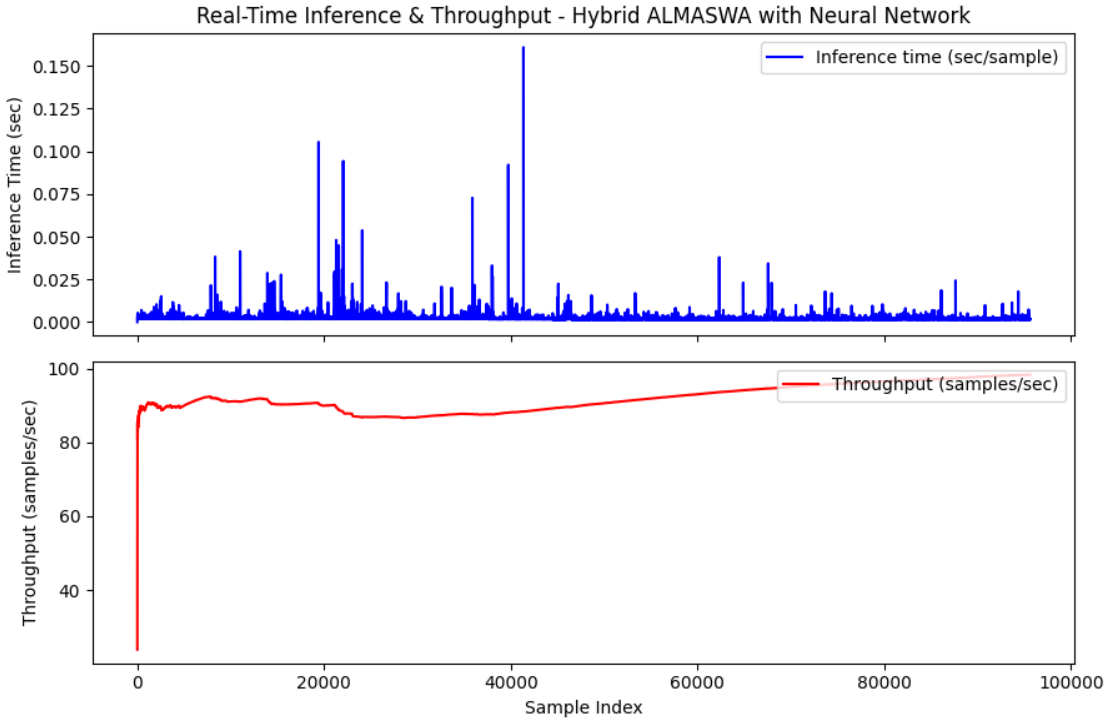


Figure 5.8: The throughput and latency of the proposed model evaluated with the UNSW NB15 dataset

cost, the results confirm that larger training volumes offer marginal but consistent gains in detection accuracy across these diverse network flow scenarios.

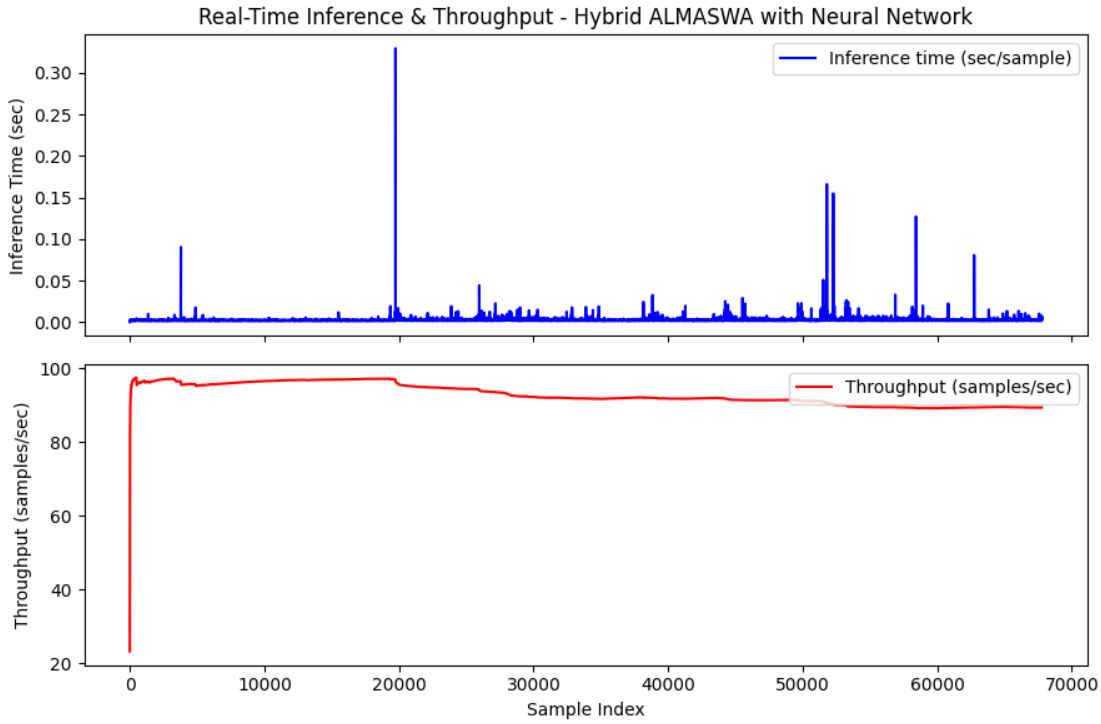


Figure 5.9: The throughput and latency of the proposed model evaluated with the ToN IoT dataset

5.5.7 Model Efficiency and Resource Utilization

Table 5.7 provides a snapshot of runtime, memory usage, and final classification accuracy when the algorithm is evaluated across the four datasets. For the NF BoT IoT dataset, the proposed algorithm recorded a processing time of 302.41 seconds and moderate memory utilization of 89.54 MB. On the NF CSE 2018 dataset, the proposed algorithm recorded a processing time of 85.86 seconds while using 89.95 MB of memory. Similarly, on the NF UNSW NB15 dataset, the proposed method used a processing time of 210.97 seconds and 88.22 MB of memory. The NF ToN dataset registered the shortest runtime of only 68.32 seconds, albeit at a higher memory footprint of 196.47 MB.

Table 5.6: Evaluating the scalability of ALMANET by measuring the continuous accuracy and processing time

Fraction of data	#Samples	Accuracy	Time (s)
NF BoT IoT			
0.1	15105	98.93%	292.76
0.25	37764	99.34%	810.19
0.5	75527	99.43%	975.65
1.0	151054	99.52%	2728.34
NF CSE 2018			
0.1	7558	96.31%	86.41
0.25	18894	97.39%	215.01
0.5	37788	98.08%	421.25
1.0	75575	98.52%	861.17
NF UNSW NB15			
0.1	9561	98.40%	218.47
0.25	23903	98.83%	456.74
0.5	47806	99.05%	783.44
1.0	95611	99.15%	1694.35
NF ToN			
0.1	6776	85.18%	63.90
0.25	16940	90.38%	157.36
0.5	33881	92.77%	314.92
1.0	67762	94.25%	648.24

Table 5.7: Measuring the efficiency and resource utilization of ALMANET

Time (s)	Memory (MB)	Accuracy
NF BoT IoT		
302.41	89.54	99.49
NF CSE 2018		
85.86	89.95	98.57%
NF UNSW NB15		
210.97	88.22	99.11%
NF ToN		
68.32	196.47	94.88%

5.6 Discussions

The results collectively highlight the effectiveness and robustness of ALMANET. The comparative metrics shown in Table 5.3 show that our approach achieves consistently higher accuracy, recall, and ROCAUC values than the baseline ALMA across four diverse network flow datasets. The rolling TPR and FPR plots shown in Figures 5.2–5.5 further underline these gains, with the hybrid model exhibiting sustained high true positive rates and significantly lower false positive rates throughout the data streams. This result means our proposed algorithm learns quickly while maintaining stability on the data stream. Even in challenging traffic

scenarios that typically degrade detection performance, the hybrid approach adaptively learns from misclassifications, preserving accuracy and minimizing false alarms.

In terms of adversarial robustness, as shown in Table 5.4, ALMANET again demonstrates notable resilience. Despite increasing adversarial noise, adversarial accuracy remains nearly on par with the clean condition, while the fraction of successful flips remains extremely low, often below 0.01%. The adversarial result indicates that even intentional perturbations fail to exploit vulnerabilities effectively. Alongside accuracy, real-time considerations are crucial for intrusion detection systems.

Regarding performance in the presence of drift, as shown in Table 5.5, ALMANET shows good results in handling incremental and recurring drift across the four datasets. The results suggest that ALMANET can effectively capture slow-evolving changes in IoT traffic. Sudden drift leads to the most significant drop in performance, particularly in the UNSW dataset (F1 of 89.97%) and CSE (F1 of 89.72%). The drop in performance reflects the model's challenge in handling abrupt, high-magnitude changes in label distribution. Performance on the ToN-IoT dataset is slightly lower than that of the other datasets, likely due to its greater class imbalance. Nevertheless, ALMANET achieves 90.97% accuracy under incremental drift, showing it can adapt effectively.

The throughput and latency findings in Figures 5.6, 5.7, 5.8, and 5.9 confirm that our approach can handle large-scale data streams, consistently processing between 80–120 samples/second with inference times typically under 0.05 seconds. The occasional latency spikes observed in Figures 5.6 - 5.9 are rare and caused by background Python garbage collection activity. Even during these spikes, we observed that throughput remains above 1800 samples/sec, well within real-time processing bounds.

Moreover, the scalability assessments in Table 5.6 reveal that while accuracy and processing time both rise with larger fractions of the dataset, the method remains computationally feasible and benefits from more extensive training data, particularly in noisy or highly varied traffic environments like NF ToN IoT. Our proposed method scales linearly in time with increasing data volume while maintaining or, in some instances, improving accuracy.

Lastly, the resource utilization data shown in Table 5.7 showed a moderate

memory footprint in general, ranging from about 88 MB to just under 200 MB, with runtime largely dependent on dataset size and complexity. NF BoT IoT and UNSW NB15 require comparatively more time to process fully but yield a detection accuracy above 90%, whereas NF ToN IoT runs faster at the cost of higher memory usage and slightly lower accuracy. Overall, these results illustrate a robust yet flexible framework that can be tailored to different network demands.

While the proposed model exhibits strong performance, certain constraints remain. The occasional latency spikes, although small on average, suggest that system overheads or resource contention could momentarily affect real-time throughput. Additionally, the proposed method's reliance on labeled data also implies that performance could degrade for highly novel threats or zero-day attacks that substantially differ from known patterns. Finally, although adversarial resilience is encouraging, more extensive testing against a wider range of sophisticated attack vectors could further validate the model's robustness under real-world adversarial conditions.

5.7 Conclusion and Future Work

Traditional IDS, particularly those relying on offline machine learning techniques, struggle with the real-time and dynamic nature of IoT networks. These systems often face high computational costs and an inability to adapt to new and evolving cyber threats rapidly. To overcome these challenges, we propose ALMANET, a hybrid IDS that combines ALMA, SWA, and an incremental neural network. This approach integrates the strengths of online machine learning to achieve real-time adaptability, reduced computational overhead, and improved attack detection capabilities.

The experimental validation of ALMANET on four benchmark datasets demonstrates its superior performance compared to ALMA and LR. ALMANET consistently achieves a ROCAUC of more than 93% across all datasets and a memory usage of 14.64 KB. Although the performance of the proposed model was slightly lower compared to RF and SVM, the memory usage of ALMANET and its performance in the presence of drift make it a good candidate for resource-constrained devices.

In terms of practical deployment, ALMANET is designed to function effi-

ciently even on resource-constrained devices, such as Raspberry Pi or an edge device, without sacrificing performance. This is a significant advantage, especially in IoT environments where computational resources are often limited. The system demonstrates real-time processing capabilities with low latency and high throughput, processing between 80–120 samples per second with average inference times under 0.05 seconds and a memory usage of 14.64 KB. These results make ALMANET a viable candidate for real-time IoT intrusion detection, where quick response times and high detection rates are critical.

The system's resilience to adversarial attacks is another noteworthy feature. Even when subjected to adversarial perturbations, ALMANET maintains high accuracy, with a minimal decrease in performance. This robustness is essential for ensuring that the system remains effective even in the face of sophisticated attack strategies designed to evade detection.

Despite these promising results, there are several areas for future research and improvement. While ALMANET performs well in binary classification, future work will focus on extending the system to handle multi-class classification. This would allow for a more granular classification of various attack types, improving the system's utility in detecting a wide range of threats. To further enhance privacy and scalability, federated learning could be incorporated into ALMANET. This would enable multiple IoT devices to learn from shared data while keeping sensitive data local, collaboratively, thus reducing the risk of data breaches.

References

[1] P. R. Agbedanu, S. J. Yang, R. Musabe, I. Gatare, and J. Rwigema, "A scalable approach to internet of things and industrial internet of things security: Evaluating adaptive self-adjusting memory k-nearest neighbor for zero-day attack detection", *Sensors*, vol. 25, no. 1, p. 216, 2025.

[2] C. Brewster, I. Roussaki, N. Kalatzis, K. Doolin, and K. Ellis, "Iot in agriculture: Designing a europe-wide large-scale pilot", *IEEE communications magazine*, vol. 55, no. 9, pp. 26–33, 2017.

[3] M. K. Hasan, T. M. Ghazal, R. A. Saeed, *et al.*, "A review on security threats, vulnerabilities, and counter measures of 5g enabled internet-of-medical-things", *IET Communications*, vol. 16, no. 5, pp. 421–432, 2022.

[4] J. C. Cano, V. Berrios, B. Garcia, and C. K. Toh, "Evolution of iot: An industry perspective", *IEEE Internet of Things Magazine*, vol. 1, no. 2, pp. 12–17, 2018.

[5] M. Ghiasi, Z. Wang, M. Mehrandezh, S. Jalilian, and N. Ghadimi, "Evolution of smart grids towards the internet of energy: Concept and essential components for deep decarbonisation", *IET Smart Grid*, 2022.

[6] M. Q. Aldossari and A. Sidorova, "Consumer acceptance of internet of things (iot): Smart home context", *Journal of Computer Information Systems*, vol. 60, no. 6, pp. 507–517, 2020.

[7] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in internet of things", *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[8] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in internet of things: The road ahead", *Computer networks*, vol. 76, pp. 146–164, 2015.

[9] P. Dini, A. Elhanashi, A. Begni, S. Saponara, Q. Zheng, and K. Gasmi, "Overview on intrusion detection systems design exploiting machine learning for networking cybersecurity", *Applied Sciences*, vol. 13, no. 13, p. 7507, 2023.

[10] J. Korstanje, *Machine Learning for Streaming Data with Python: Rapidly build practical online machine learning solutions using River and other top key frameworks*. Packt Publishing Ltd, 2022.

[11] C. Gentile, "A new approximate maximal margin classification algorithm", *Journal of Machine Learning Research*, vol. 2, no. Dec, pp. 213–242, 2001.

[12] C. Ioannou and V. Vassiliou, "Network attack classification in iot using support vector machines", *Journal of sensor and actuator networks*, vol. 10, no. 3, p. 58, 2021.

[13] A. Davahli, M. Shamsi, and G. Abaei, "A lightweight anomaly detection model using svm for wsns in iot through a hybrid feature selection algorithm based on ga and gwo", *Journal of Computing and Security*, vol. 7, no. 1, pp. 63–79, 2020.

[14] M. Douiba, S. Benkirane, A. Guezzaz, and M. Azrour, "An improved anomaly detection model for iot security using decision tree and gradient boosting", *The Journal of Supercomputing*, vol. 79, no. 3, pp. 3392–3411, 2023.

[15] A. Guezzaz, S. Benkirane, M. Azrour, and S. Khurram, "A reliable network intrusion detection approach using decision tree with enhanced data quality", *Security and Communication Networks*, vol. 2021, no. 1, p. 1 230 593, 2021.

[16] G. Balhareth and M. Ilyas, "Optimized intrusion detection for iomt networks with tree-based machine learning and filter-based feature selection", *Sensors*, vol. 24, no. 17, p. 5712, 2024.

[17] P. Kumar, G. P. Gupta, and R. Tripathi, "A distributed ensemble design based intrusion detection system using fog computing to protect the internet of things networks", *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 10, pp. 9555–9572, 2021.

[18] Y. Otoum, D. Liu, and A. Nayak, "Dl-ids: A deep learning–based intrusion detection framework for securing iot", *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 3, e3803, 2022.

[19] I. Ullah and Q. H. Mahmoud, "Design and development of a deep learning-based model for anomaly detection in iot networks", *IEEE Access*, vol. 9, pp. 103 906–103 926, 2021.

[20] S. Soliman, W. Oudah, and A. Aljuhani, "Deep learning-based intrusion detection approach for securing industrial internet of things", *Alexandria Engineering Journal*, vol. 81, pp. 371–383, 2023.

[21] J. B. Awotunde, C. Chakraborty, and A. E. Adeniyi, "Intrusion detection in industrial internet of things network-based on deep learning model with rule-based feature selection", *Wireless communications and mobile computing*, vol. 2021, no. 1, p. 7 154 587, 2021.

[22] M. Ge, N. F. Syed, X. Fu, Z. Baig, and A. Robles-Kelly, "Towards a deep learning-driven intrusion detection approach for internet of things", *Computer Networks*, vol. 186, p. 107 784, 2021.

[23] S. Choudhary and N. Kesswani, "Analysis of kdd-cup'99, nsl-kdd and unsw-nb15 datasets using deep learning in iot", *Procedia Computer Science*, vol. 167, pp. 1561–1573, 2020.

[24] R. Khilar, K. Mariyappan, M. S. Christo, *et al.*, "Artificial intelligence-based security protocols to resist attacks in internet of things", *Wireless Communications and Mobile Computing*, vol. 2022, 2022.

[25] O. A. Wahab, "Intrusion detection in the iot under data and concept drifts: Online deep learning approach", *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 19 706–19 716, 2022.

[26] Z. E. Huma, S. Latif, J. Ahmad, *et al.*, "A hybrid deep random neural network for cyberattack detection in the industrial internet of things", *IEEE Access*, vol. 9, pp. 55 595–55 605, 2021.

[27] C. Constantinides, S. Shiaeles, B. Ghita, and N. Kolokotronis, "A novel online incremental learning intrusion prevention system", in *2019 10th IFIP International conference on new technologies, mobility and security (NTMS)*, IEEE, 2019, pp. 1–6.

[28] M. R. Martina and G. L. Foresti, "A continuous learning approach for real-time network intrusion detection", *International Journal of Neural Systems*, vol. 31, no. 12, p. 2 150 060, 2021.

[29] C. Nixon, M. Sedky, and M. Hassan, "Practical application of machine learning based online intrusion detection to internet of things networks", in *2019 IEEE Global Conference on Internet of Things (GCIoT)*, IEEE, 2019, pp. 1–5.

[30] N. W. Abderrahim and A. Benosman, "Adaptive intrusion detection in iot: Combining batch and incremental learning for enhanced security", *Engineering Research Express*, vol. 7, no. 1, p. 015 278, 2025.

[31] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization", *arXiv preprint arXiv:1803.05407*, 2018.

[32] C. Demir, A. Sharma, and A.-C. N. Ngomo, "Adaptive stochastic weight averaging", *arXiv preprint arXiv:2406.19092*, 2024.

[33] J. Vitorino, N. Oliveira, and I. Praça, "Adaptative perturbation patterns: Realistic adversarial learning for robust intrusion detection", *Future Internet*, vol. 14, no. 4, p. 108, 2022.

[34] J. Vitorino, I. Praça, and E. Maia, "Towards adversarial realism and robust learning for iot intrusion detection and classification", *Annals of Telecommunications*, vol. 78, no. 7, pp. 401–412, 2023.

[35] M. A. Ferrag, L. Shu, O. Friha, and X. Yang, "Cyber security intrusion detection for agriculture 4.0: Machine learning-based solutions, datasets, and future directions", *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 3, pp. 407–436, 2021.

[36] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set", in *2009 IEEE symposium on computational intelligence for security and defense applications*, Ieee, 2009, pp. 1–6.

[37] J. Montiel, M. Halford, S. M. Mastelini, *et al.*, "River: Machine learning for streaming data in python", *J. Mach. Learn. Res.*, vol. 22, no. 1, Jan. 2021, ISSN: 1532-4435.

[38] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets", *Mobile networks and applications*, pp. 1–14, 2022.

Chapter 6

Chapter 6 is based on the following article:

P. R. Agbedanu, S. J. Yang, R. Musabe, I. Gatare, J. Rwigema, "Adaptive Multi-Label Intrusion Detection in IoT Networks Using ALMA with Stochastic Weight Averaging," Submitted for publication (under review).

Adaptive Multi-Label Intrusion Detection in IoT Networks Using ALMA with Stochastic Weight Averaging

Authors: Promise Ricardo Agbedanu, Sanchieh Jay Yang, Richard Musabe, Ignace Gatare, James Rwigema

Abstract: In recent years, intrusion detection systems (IDSs) for the Internet of Things (IoT) have focused on building detection systems using binary or multi-class algorithms. However, most modern attacks are not mutually exclusive. In other words, a malware attack may coincide with another attack, making it difficult for an IDS designed using a binary or multi-class algorithm to detect such an attack. Approximate Large Margin Algorithm (ALMA) is an online algorithm that is fast, lightweight, and adaptive, but performs poorly on noisy data. In this paper, we propose ALMA integrated with SWA and using an adaptive learning rate decay to reduce the learning rate gradually. The proposed algorithm is then integrated into an online chain classifier to build a multi-label algorithm. The results after validating the proposed algorithm on four datasets show the model recording an exact match of 81.38% and a Hamming loss as low as 0.0758. The chain classifier significantly increases the detection accuracy of attacks that ALMA and ALMA with SWA struggle to detect. The proposed method approximately uses 0.02 MB of memory and 0.82% of CPU to process one data sample.

6.1 Introduction

The IoT ecosystem has led to the creation of smart industries and made it possible for everyday objects like refrigerators, air conditioners, and microwaves to

communicate over the internet. This notwithstanding, the massive increase in the adoption and usage of IoT has created a serious security concern. As the number of cyber-attacks targeted at IoT devices increases, intrusion detection systems (IDS) have become a mitigation tool against these attacks. However, these IDS can either distinguish between normal and malicious traffic. In the real world, distinguishing between normal and malicious traffic is not adequate. It is possible for different attacks to happen concurrently. This issue of concurrent attacks is a critical problem in the IoT ecosystem, not just because of the computational constraints of IoT devices, but also due to the fact that most IDS are not designed to detect concurrent attacks.

Most IoT security academic research focuses on designing IDSs that use either binary or multi-class classification techniques. Detecting modern attacks demands a more advanced approach that is capable of detecting concurrent attacks. For example, a malware attack may be deployed with a spoofing attack. Attack detection in the IoT ecosystem needs systems that use less energy, processing power, and memory. The idea of detecting concurrent attacks using lightweight approaches in a highly heterogeneous environment is one of the research challenges faced by security researchers.

Approximate Large Margin Algorithm (ALMA) is an online machine learning (ML) algorithm that is lightweight, adaptive, and uses norm projection of the weight vector, which helps control model complexity and keeps the model stable during updates [1]. However, our initial assessment of the robustness of ALMA on noisy data indicated that ALMA has a generalization to noisy data. Stochastic Weight Averaging (SWA) was shown by Izmailov et al. [2] to improve neural network generalization by averaging weights over training epochs, effectively ensembling models without added runtime cost. This motivated us to incorporate SWA into ALMA to handle the generalization problem while using an online chain classifier approach to handle the dependencies that may exist in different attacks.

The contributions of this work are as follows:

- We propose the first integration of Stochastic Weight Averaging (SWA) into the Approximate Large Margin Algorithm (ALMA), a convex, norm-constrained online learner. This fusion improves the model's generalization and temporal stability without sacrificing ALMA's fast convergence and simplicity.

- We introduce a novel pause and resume learning control based on the model's confidence in its predictions. Learning is paused when prediction confidence is high and resumed when uncertainty increases, thereby reducing unnecessary updates and mitigating overfitting in situations where data is stable.
- The model is extended to a multi-label classifier chain, where each label is predicted conditionally based on prior label predictions. Each node in the chain is an independent SWA regularized ALMA classifier, enabling the system to model inter-label dependencies effectively in streaming contexts.
- The proposed algorithm can dynamically expand its label space, automatically initializing new classifiers for unseen labels encountered during the data stream.

The rest of the paper consists of Section 2, which focuses on related works. The proposed methodology is presented in Section 3. The experimental design is presented in Section 4, while the experimental results are presented in Section 5. The discussion of the study is presented in Section 6, and finally, the conclusions of the study are presented in Section 7.

6.2 Related Works

Several recent studies have recognized the need for multi-label intrusion detection to handle overlapping cyber-attacks. Hallaji et al. proposed an ensemble of sequential classifiers that preserves separate label sets during training to avoid interference [3]. By initializing each model with the previous model's parameters, their framework captures label correlations and improves multi-label detection performance without added complexity. In a similar work, Huang et al. proposed a two-stage model fusion to handle network attacks. The approach first performs a binary classification on a selected label and then uses that outcome as a feature in a multi-label classifier. This approach boosted detection accuracy over single-stage baselines [4]. Our work shares the goal of multi-label IDS but differs in methodology. Instead of multiple training phases, we employ an online classifier chain with ALMA incorporated with SWA. This makes our IDS lightweight and adaptive, using only 0.02 MB of memory and 0.82% CPU per sample, significantly lower overhead than deep ensemble models, while still capturing label dependencies.

Unlike Hallaji et al. and Huang et al., who use batch learning with complex model sequencing, our online approach continuously updates the model and is naturally suited for streaming IoT data.

Other researchers have explored multi-label detection with advanced machine learning techniques. Sharma et al. investigated low-rank machine learning and deep learning models for multi-label IoT attack classification [5]. They combined BoT-IoT and UNSW-NB15 datasets and applied support vector machines (SVM) as well as CNN-based models, optimizing hyperparameters via Bayesian methods. Their best model (a hybrid CNN-MLP) achieved high accuracy, for example, correctly detecting a UDP flooding attack simultaneously labeled as both analysis and DoS with 98.5% accuracy [5]. This demonstrates the benefit of designing models to detect concurrent attack labels. However, Sharma's deep models are computationally heavier. In contrast, our ALMASWA chain classifier provides a fast, adaptive alternative that maintains an exact match of 81.38%, and a Hamming loss of 0.0758 while using minimal resources. Karunanayake et al. also addressed multi-label intrusion detection for Tor encrypted malware traffic. They introduced a message passing neural network that outperformed traditional multi-label methods like binary relevance, classifier chains, and label powerset. The proposed technique achieved over 90% precision and recall in classifying multiple malware families simultaneously. They further used explainable AI to interpret the model and tested adversarial robustness [6]. Our work approaches multi-label detection at a more general network level, not restricted to Tor traffic, and uses a simpler linear model chain. While Karunanayake et al. [6] achieved impressive results with a complex graph-based model, our method focuses on broad IoT applicability and building a noise-resilient detection system. For instance, integrating SWA into ALMA to stabilize online learning, since basic ALMA can degrade with noisy data. Similar to the work of Karunanayake et al. [6], we found that modeling label relationships improves detection. Our chain classifier significantly boosts accuracy on attack combinations that a single ALMA model struggled with.

Beyond multi-label algorithms, many IDS researchers have tackled the multi-class classification of numerous attack types across various domains. Qaddoura et al. presented a multi-layer IoT IDS that first detects the existence of an intrusion and then determines the type of attack. Applying a single hidden-layer feed-forward neural network to an over-sampled dataset, they improved detection

of minority attacks, achieving a G-mean of 78% higher than traditional single-stage classifiers [7]. This two-stage design is effective for sequential attack identification, but it still assumes one attack type per instance. Building IDSs that can detect multiple attacks has been an area of interest in cybersecurity research. Sahu et al., [8] designed a multi-class IDS using a deep learning technique with the aim of classifying multi-class attacks more accurately. A Hidden Naïve Bayes model was used by [9] to build an IDS that could detect normal and attack events. In another study, [10] developed a multi-class IDS using a Ramp Loss K-Support Vector Classification-Regression (Ramp-KSVCR). The authors also used the Alternating Direction Method of Multipliers (ADMM) to make the model adaptable and applicable in large-scale environments and reduce the model training time. In a similar study, [11] used a chi-square to reduce the dimension of data and then used a multi-class support vector machine to develop an IDS that can detect different network attacks. Alaiz et al., [12] proposed models for detecting multi-class attacks in IoT environments. The models considered were LSTM, GRU, and XGBoost. The performance of various ensemble ML algorithms was evaluated on a multi-class dataset [13]. Besides focusing on a multi-class IDS, the authors also focused on designing an efficient IDS that has good accuracy and good execution speed. Similarly, [14] used heterogeneous ensemble learning algorithms to execute both binary and multi-class approaches to detect attacks in IoT environments. In the quest to design an efficient method to detect attacks in the IoT space, [15] explores deep learning algorithms, including transformers, to solve the problem. To detect cyber-attacks in IIoT systems, [16] tried to design an IDS that can detect multi-class attacks as well as handle the problem of imbalanced data distributions within most IIoT datasets. The authors used XGBoost to model an IDS using X-IIoTDS and TON_IoT datasets, two modern IIoT imbalanced datasets. The authors showed that their proposed method achieved excellent attack detection. An adaptive IDS can effectively improve the attack detection rate. A two-layer multi-class detection was proposed by [17] to design an adaptive IDS using a combination of C5.0 and Naïve Bayes algorithm. In a scenario with concurrent attacks (for example, a malware that performs DDoS while exfiltrating data), a single-type classifier might only catch the dominant attack. Our chain classifier, by contrast, would output multiple labels in such cases, detecting all co-occurring threats.

Silivery and Kovvur likewise focused on multi-attack classification using deep

learning. They evaluated recurrent and deep neural networks on KDD'99, NSL-KDD, and UNSW-NB15 datasets. Their LSTM-RNN model with an Adamax optimizer achieved the best accuracy and detection rate on NSL-KDD, outperforming shallow models. They also demonstrated a multi-model approach (combining RNN, LSTM, and DNN) that learned features across datasets and improved classification of up to 15 attack classes [18]. Even though these datasets are older (KDD'99 dates to 1998 and NSL-KDD dates to 2009), many recent works still use them as benchmarks [19]. This highlights a gap that traditional IDS datasets rarely include overlapping attacks. Methods like the one proposed by Silivery and Kovvur evaluate multi-class performance but not true multi-label scenarios. By validating our approach on the four datasets used in our study, we ensure that our multi-label IDS is effective on both classic attacks and modern IoT attack patterns.

Ensemble and hybrid models are a common strategy to boost multi-class IDS accuracy. Abbas et al. proposed an ensemble voting classifier combining logistic regression, Naïve Bayes, and decision trees, and showed improved accuracy on the CIC-IDS2017 dataset. Their ensemble outperformed individual learners in both binary and multi-class detection. In a follow-up study, Abbas et al. extended this idea by evaluating various ensemble algorithms for IoT intrusion detection, confirming that ensembles can robustly detect diverse attacks in IoT traffic [20]. We similarly find that combining learning strategies is beneficial. Our method can be seen as an ensemble of label classifiers. The chain feeds each predicted label as a feature to the next, effectively ensembling decisions across labels. However, unlike a standard ensemble of different algorithms as in Abbas et al., our chain model is a unified online algorithm, which simplifies deployment and avoids maintaining multiple classifiers. It's also worth noting that ensembles and deep networks have been tuned for specific attack categories. The challenge that our work addresses is detecting multiple attack types when they occur together. Our results, an exact match of 81%, indicate the proposed multi-label chain can accurately identify complex attack combinations, whereas a typical multi-class IDS might misclassify or miss some of these if forced to pick only one label.

Finally, we review some deep learning-based IDS approaches across different domains to underscore how they handle attack diversity. Many works aim to improve feature extraction or address class imbalance, but still assume a single-label output. For instance, Halbouni et al. combined CNN and LSTM networks to detect

intrusions, demonstrating improved accuracy over plain CNNs [21]. However, they did not evaluate in a specific industrial environment, and their method, like others, outputs one class per instance. Hassan et al. [22] proposed a CNN with a weight-dropped LSTM to capture long-range dependencies, which improved detection rates on the benchmark datasets used for their experimental evaluation. Jin et al. developed a multi-scale 1D CNN with BiLSTM for ICS networks and applied SMOTE; while this boosted minority class recall, the synthetic data added some noise, limiting generalization [23]. Wu et al. [24] used a stacked transformer encoder–decoder for network traffic and improved detection in cloud environments, though extracting multi-scale features remained difficult and overfitting occurred on imbalanced data. Similarly, Long et al. [25] achieved an accuracy greater than 93% with a Transformer IDS for cloud traffic, yet found it challenging to handle varied attack granularities. In our work, by using a multi-label approach, we detect each aspect of such attacks without conflating them into an exponential number of classes. This flexibility is a key advantage of our chain classifier.

6.3 Proposed System

We propose an online learning classifier based on the Approximate Large Margin Algorithm (ALMA), enhanced with Stochastic Weight Averaging (SWA) and dynamic confidence-based learning control. The algorithm maintains a weight vector $\mathbf{w} \in \mathbb{R}^d$ for classification and is updated in an online manner for each instance $(\mathbf{x}_t, y_t)$, where $\mathbf{x}_t \in \mathbb{R}^d$ is the feature vector and $y_t \in \{-1, +1\}$ is the true label (binary).

6.3.1 Prediction

The prediction score at time t is given by:

$$s_t = \mathbf{w}_t^\top \mathbf{x}_t \tag{6.1}$$

The probability estimate is computed using the sigmoid function:

$$\hat{p}_t = \sigma(s_t) = \frac{1}{1 + e^{-s_t}} \tag{6.2}$$

When Stochastic Weight Averaging is enabled (after a burn-in period), predictions use the averaged weights $\bar{\mathbf{w}}$:

$$s_t = \bar{\mathbf{w}}^\top \mathbf{x}_t, \quad \bar{\mathbf{w}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{w}_i \tag{6.3}$$

6.3.2 Learning Rule

Given a misclassified or low-confidence example, the weight vector $\mathbf{w}_t$ is updated as:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta_t y_t \mathbf{x}_t \tag{6.4}$$

where $\eta_t = \frac{C}{\sqrt{1+t}}$ is a time-decaying learning rate, and $y_t \in \{-1, +1\}$. After the update, weights are projected back onto the ℓ_p-norm ball of radius B:

$$\mathbf{w}_{t+1} \leftarrow \frac{\mathbf{w}_{t+1}}{\max(1, \|\mathbf{w}_{t+1}\|_p)} \tag{6.5}$$

The model only updates when the margin condition is violated:

$$y_t(\mathbf{w}_t^\top \mathbf{x}_t) < (1 - \alpha)\gamma_t \quad \text{where} \quad \gamma_t = \frac{B\sqrt{p-1}}{\sqrt{t}} \tag{6.6}$$

6.3.3 Confidence-Based Learning Control

Learning is paused if the predicted confidence $\max(\hat{p}_t, 1 - \hat{p}_t) \geq \theta_{\text{pause}}$, and resumed when it drops below a lower threshold θ_{resume}. This dynamic adaptation improves robustness to overfitting. Our inspiration to include a confidence-based learning control in the proposed algorithm is drawn from [26], [27], although the approach proposed by these authors is different from what we propose in this work.

An approach in industrial anomaly detection uses a thresholding technique aimed at ensuring high prediction confidence during continuous online adaptation. This functions akin to gating mechanisms used to decide whether to update the model [27].

This work introduces a confidence threshold to control when a neural network should instantiate new classes during incremental learning. Once confidence drops below a set threshold, the model considers the sample as "unknown" and updates accordingly—mirroring your Single-Label ALMASWA gating logic [26].

6.3.4 Stochastic Weight Averaging (SWA)

At every k-th step after a delay t_0, SWA updates the averaged weights as:

$$\bar{\mathbf{w}} \leftarrow \frac{1}{N+1}\left(N \cdot \bar{\mathbf{w}} + \mathbf{w}_t\right) \tag{6.7}$$

This averaging promotes convergence to flatter minima, improving generalization in online settings.

6.3.5 Multi-Label ALMA Classifier Chain

We extend the base ALMASWA model using a classifier chain approach for multi-label classification. Let $\mathcal{L} = \{l_1, l_2, \ldots, l_m\}$ be the set of labels. The prediction of each label l_j is conditioned on both the input features and the predictions of previous labels in the chain:

$$\hat{y}_{l_j} = f_j(\mathbf{x}, \hat{y}_{l_1}, \ldots, \hat{y}_{l_{j-1}}) \tag{6.8}$$

At training time, the model learns each label sequentially, augmenting the input with the predicted probability $\hat{p}_{l_i}$ for each previous label:

$$\mathbf{x}_j = \mathbf{x} \cup \{\hat{p}_{l_1}, \ldots, \hat{p}_{l_{j-1}}\} \tag{6.9}$$

Each classifier f_j is an independent instance of the ALMASWA model. The learning update and confidence control are applied per label independently.

Let $\mathcal{H} = \{h_1, \ldots, h_m\}$ denote the collection of ALMASWA models for each label. The full model learns from instance $(\mathbf{x}_t, \mathbf{y}_t)$, where $\mathbf{y}_t \in \{0, 1\}^m$ represents the binary label vector, by:

$$\forall j \in \{1, \ldots, m\}: \quad h_j.\text{learn\_one}(\mathbf{x}_j, y_{l_j}) \tag{6.10}$$

New labels not previously seen are dynamically added to the chain with a fresh model instantiation.

The algorithmic pseudocode of the proposed model is shown in Algorithm 3.

Algorithm 3 Multi-Label ALMA Classifier Chain with SWA and Confidence-Based Control

Require: Input feature vector $\mathbf{x}_t$, label dictionary $\mathbf{y}_t$, model chain $\mathcal{H}$, order $\mathcal{L}$

for all label $l_j \in \mathcal{L}$ **do**
  $h_j \leftarrow \mathcal{H}[l_j]$
  $\hat{p}_j \leftarrow \sigma(h_j^\top \mathbf{x}_t)$ {Predict probability}
  $\text{conf}_j \leftarrow \max(\hat{p}_j, 1 - \hat{p}_j)$
  if h_j is paused **then**
    if $\text{conf}_j < \theta_{\text{resume}}$ **then**
      Unpause h_j
    else
      continue {Skip update}
    end if
  else
    if $\text{conf}_j \geq \theta_{\text{pause}}$ **then**
      Pause h_j
      continue
    end if
  end if
  $y_j \leftarrow$ get true label for l_j
  $s_j \leftarrow h_j^\top \mathbf{x}_t$ {Margin score}
  $\gamma_t \leftarrow \frac{B\sqrt{p-1}}{\sqrt{t}}$
  if $y_j \cdot s_j < (1 - \alpha) \cdot \gamma_t$ **then**
    $\eta_t \leftarrow \frac{C}{\sqrt{1+t}}$
    $h_j \leftarrow h_j + \eta_t \cdot y_j \cdot \mathbf{x}_t$
    Project h_j onto ℓ_p-ball of radius B
  end if
  if $t \geq t_0$ **and** $(t - t_0) \bmod f = 0$ **then**
    $\bar{h}_j \leftarrow \frac{1}{n+1}(n \cdot \bar{h}_j + h_j)$
    $n \leftarrow n + 1$
  end if
  Augment $\mathbf{x}_t \leftarrow \mathbf{x}_t \cup \{\text{pred}_{l_j} = \hat{p}_j\}$
end for
for all unseen label $l_k \in \mathbf{y}_t \setminus \mathcal{L}$ **do**
  Add l_k to $\mathcal{L}$ and initialize h_k
  Repeat steps 2–22 for l_k
end for

6.3.6 System Architecture

The proposed IDS is deployed on a Raspberry Pi Model 5, serving as an IoT gateway. The traffic of the various IoT devices is sent to the gateway to be processed by the IDS. The IDS then determines whether the traffic is an attack or benign. The architecture of our proposed system is shown in Figure 6.1.

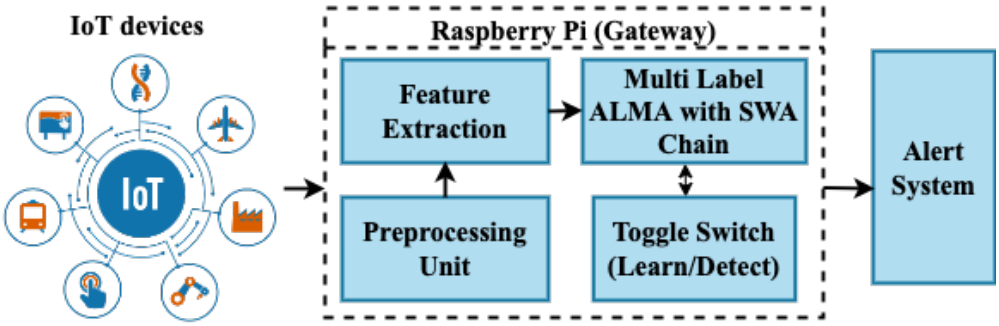


Figure 6.1: A system architecture of the proposed IDS

6.4 Experimental Design

6.4.1 Datasets

Four datasets were used to validate the proposed multi-label algorithm. The datasets used are NF-BoT-IoT and NF-ToN-IoT datasets [28]. The third dataset is the EdgeIIoT created by [29], and the fourth is the CIC IoT dataset created by [30]. We chose four attack categories from each of the datasets. Where the dataset contained more than four attack categories, we chose the four with the most observations. Table 6.1 summarizes the four attack categories with benign samples chosen from each dataset and their respective number of samples.

6.4.2 Experimental Validation

6.4.2.1 Baseline Performance of ALMA, ALMA with SWA, and Multi-Label ALMA with SWA

In this experiment, we compared the performance of three models: the original ALMA, ALMA with SWA, and multi-label ALMA with SWA. The goal was to determine whether the addition of SWA improves the predictive capabilities of the ALMA classifier and whether it enhances the performance of our multi-label classifier.

All models were tested using streaming data, where each data instance is processed sequentially. As each instance arrives, the models make a prediction and then update their learning parameters based on the true label, simulating a real-time learning scenario. Accuracy was the metric used to measure the performance of the models.

Table 6.1: A summary of the number of samples of the four attack categories selected from each dataset

Attack Category	**#Samples**
NF BoT IoT	
DDoS	56844
DoS	56833
Reconnaissance	470655
Theft	1909
Benign	13859
NF ToN IoT	
DDoS	326345
Injection	468539
Password	156299
xss	99944
Benign	270279
EdgeIIoT	
DDoS	49396
Ransomware	10925
SQL Injection	10311
Uploading	10269
Normal	24301
CIC IoT	
DDoS	325946
DoS	77526
Malware	25135
Spoofing	4788
Benign	10525

6.4.2.2 Performance of Multi-Label ALMA with SWA

In this experiment, we explored the performance of our proposed online multi-label classification model. We utilized a streaming approach where the model processes data instances sequentially, making predictions and updating its learning parameters in real-time. The metrics measured in this experiment are exact match score and Hamming loss.

6.4.2.3 Performance of Multi-Label ALMA compared with SWA against Multi-Label ALMA without SWA

In this experiment, we evaluated and compared the performance of two variations of the proposed multi-label classification model: one incorporating SWA and the other without it. Both models were tested under identical conditions to ensure a fair assessment. We conducted multiple runs with different random seeds to shuffle the data differently each time, enhancing the robustness of our evaluation.

For each run, we initialized the two models and processed the data stream

sequentially. As the models predict and learn from each data instance in real-time, we tracked their predictive performance using exact match and Hamming loss.

After all runs were completed, we aggregated the metrics collected from each run to perform a statistical analysis. Specifically, we use paired t-tests to determine if the differences in performance of the two models were statistically significant. This analysis helps us ascertain whether incorporating SWA into the multi-label ALMA yields a meaningful improvement over the standard approach.

6.4.2.4 Performance on Simulated Concurrent Attacks

This experiment explored the impact of concurrent label occurrences on the performance of the proposed multi-label classification model. The experiment simulated scenarios where multiple labels are activated simultaneously to assess how effectively the model can learn and predict these concurrent attacks. This is particularly relevant in domains like IoT, where different attacks may happen simultaneously. The dataset was modified to include instances where multiple attacks occur simultaneously. This was achieved by randomly selecting some attacks to automatically happen when a particular attack is launched, thereby creating artificial concurrency of attacks. A comprehensive set of performance metrics was defined to evaluate the model. These include exact match score, hamming loss, precision, and recall.

6.4.2.5 Resource Utilization

In this experiment, we assessed the performance and efficiency of our proposed multi-label classification model when processing streaming data. Beyond the predictive performance, the experiment placed a significant emphasis on computational resource utilization. We meticulously tracked various performance indicators, such as the average processing time per data instance, the memory footprint of the model, and the CPU usage throughout the processing. We repeated this experiment 10 times and recorded the mean of the metrics. By repeating the experiment multiple times, we aimed to obtain averaged metrics that provide a reliable insight into the model's efficiency and scalability in a streaming environment. This comprehensive evaluation helps in understanding not only how well the model predicts but also how resource-intensive it is during operation.

6.4.2.6 The Impact of Label Order on the Performance of the Classifier

The experiment investigated the impact of label ordering on the performance of the proposed algorithm. The purpose of this experiment was to determine whether the sequence in which labels are introduced affects the model's accuracy and to identify the optimal label order that yields the best predictive performance. Two primary metrics were used to evaluate the model's performance for each label order: exact match and hamming loss. The results were sorted based on the performance metrics, prioritizing higher exact match scores and lower hamming losses. The top two performing and bottom two performing label orders were identified to understand the impact of label sequencing.

6.5 Results

6.5.0.1 Baseline Performance of ALMA, ALMA with SWA, and Multi-Label ALMA with SWA

For each attack category under the respective dataset, as shown in Table 6.2, multi-label ALMA with SWA consistently outperforms the performance of the other models. For instance, on the NF-BoT-IoT dataset, multi-label ALMA with SWA achieves an accuracy of 90.26% on DDoS and DoS, compared to 59.96% and 47.24%, respectively, recorded by ALMA with SWA. This trend is consistent across datasets, where multi-label ALMA with SWA model exhibits stronger predictive performance, indicating that the inclusion of the classifier chain with SWA of the proposed algorithm stabilizes and improves the model's predictive accuracy. Figures 6.2, 6.3, 6.4, and 6.5 depict the accuracy per output of ALMA, ALMASWA, and Multi-Label ALMA with SWA when evaluated with NF BoT, NF ToN, EdgeIIoT, and CIC IoT dataset, respectively.

6.5.0.2 Performance of Multi-Label ALMA with SWA

As seen in Table 6.3, the exact match and hamming loss metrics suggest that Multi-Label ALMA with SWA performs well in multi-label attack classifications. Evaluating the model on the NF-BoT-IoT dataset, the model achieved an exact match of 72.63% and a hamming loss of 0.0758. On the CIC IoT dataset, the model recorded the highest exact match score of 81.38% with a hamming loss of 0.0778. The lowest exact match was recorded on the EdgeIIoT dataset with an

Table 6.2: A Comparison of Accuracy Per Output of ALMA, ALMA with SWA, and Multi-Label ALMA with SWA

Attack	**ALMA**	**ALMA with SWA**	**Multi-Label ALMASWA**
NF-BoT-IoT			
Reconnaissance	82.52%	89.22%	89.22%
DDoS	53.56%	59.96%	90.26%
DoS	53.93%	47.24%	90.26%
Theft	62.28%	67.42%	99.68%
NF-ToN-IoT			
Injection	61.00%	72.83%	72.83%
DDoS	63.09%	76.01%	87.05%
Password	49.69%	51.89%	88.63%
XSS	51.11%	49.53%	92.75%
EdgeIIoT			
DDoS	81.68%	86.63%	86.63%
Injection	64.14%	72.72%	79.99%
Information Gathering	65.41%	80.41%	89.24%
Malware	55.87%	64.11%	86.56%
CIC IoT			
DDoS	78.96%	85.69%	85.69%
DoS	53.04%	58.08%	84.27%
Malware	89.88%	98.45%	99.86%
Spoofing	62.82%	80.68%	99.05%

Table 6.3: The exact match and hamming loss of the proposed IDS

Exact Match	**Hamming Loss**
NF-BoT-IoT	
72.63%	0.0758
NF-ToN-IoT	
62.94%	0.1468
EdgeIIoT	
49.06%	0.1644
CIC IoT	
81.38%	0.0778

exact match of 49.06%.

6.5.0.3 Performance of Multi-Label ALMA compared with SWA against Multi-Label ALMA without SWA

Comparing Multi-Label ALMA with and without SWA, as shown in Table 6.4, the SWA-enhanced model demonstrates substantial gains in exact match and reduced hamming loss. For example, in NF-ToN-IoT, the exact match improves from 43.41% to 62.94% with a reduction in hamming loss from 0.2197 to 0.1468. This significant improvement highlights SWA's role in refining the model's multi-label capabilities, particularly in complex datasets with multiple attack labels.

Table 6.5 shows that the performance of our proposed model is not by chance.

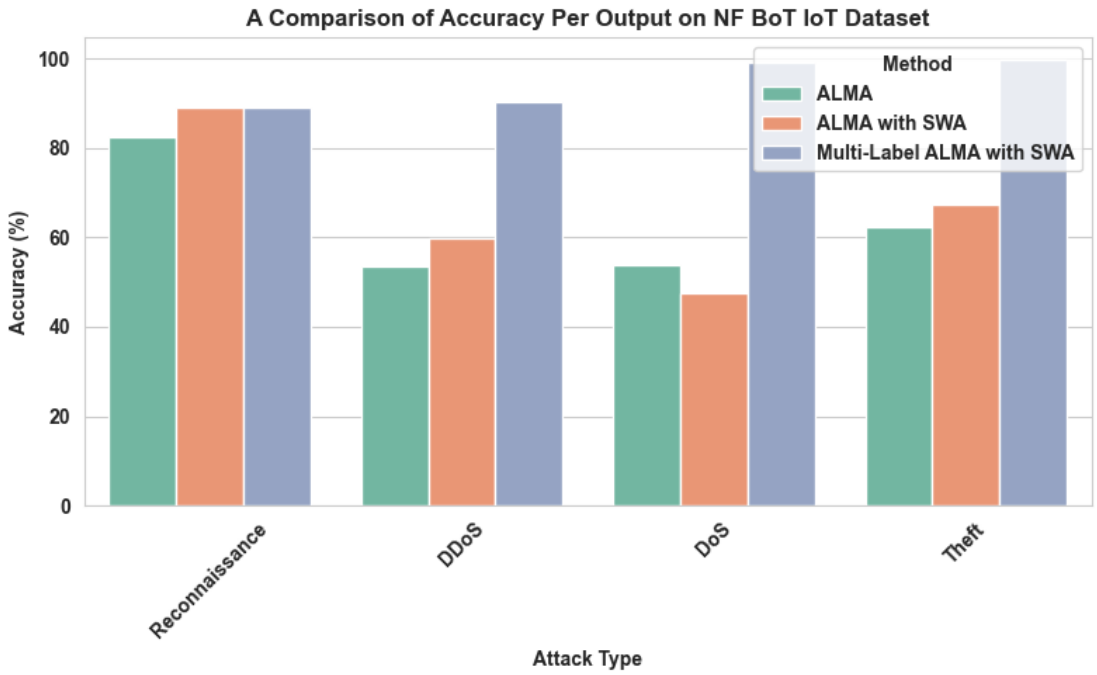


Figure 6.2: Comparing the accuracy per output of ALMA, ALMASWA, and the proposed Multi-Label ALMASWA when evaluated on the NF BoT IoT dataset

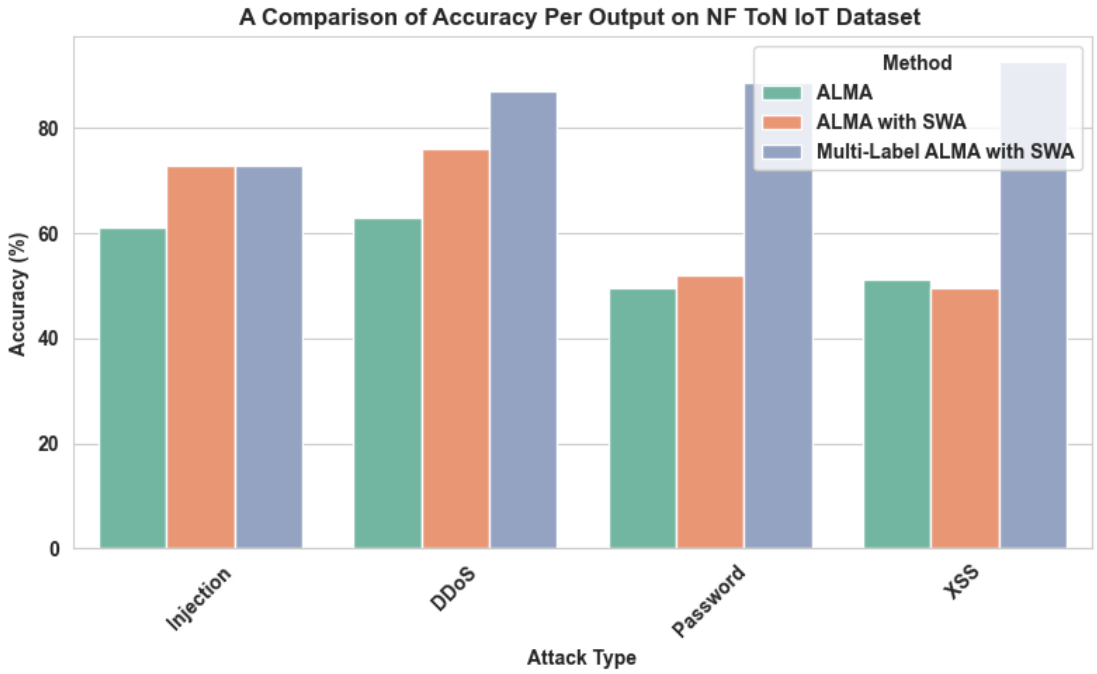


Figure 6.3: Comparing the accuracy per output of ALMA, ALMASWA, and the proposed Multi-Label ALMASWA when evaluated on the NF ToN IoT dataset

After running multi-label ALMA and multi-label ALMA with SWA five times using different seed values, the results show that multi-label ALMA with SWA performs better in all cases.

From table 6.6, a T-statistic confirms the significance of SWA's impact on exact match and hamming loss. Across all datasets, the T-statistics for exact match and hamming loss are highly significant, affirming that the observed performance

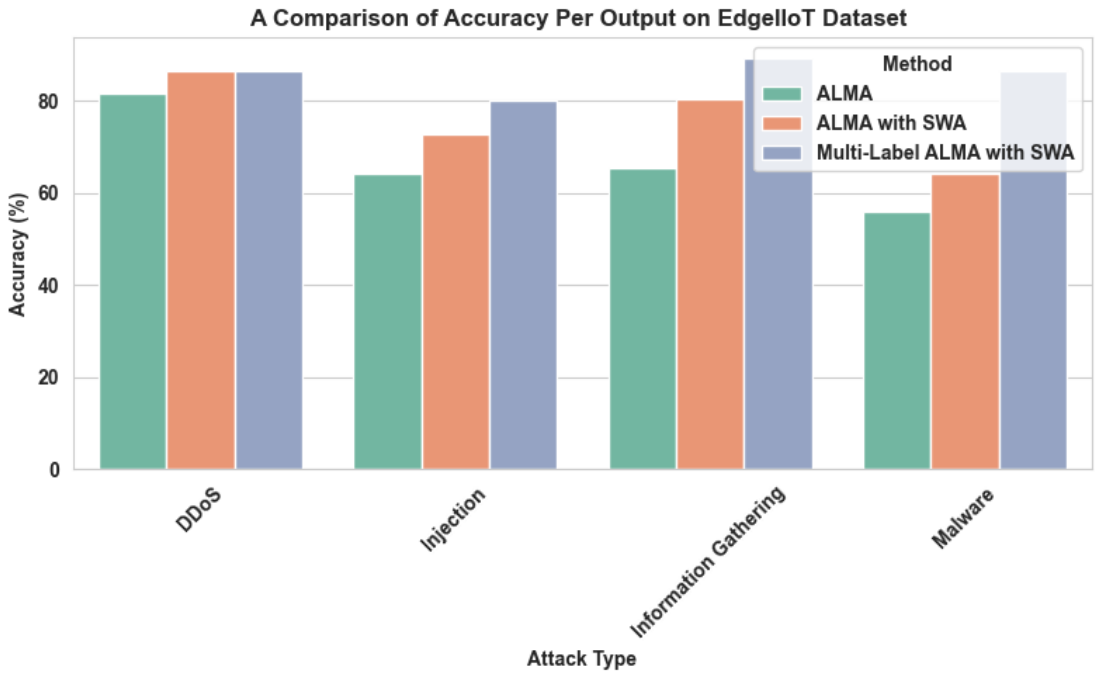


Figure 6.4: Comparing the accuracy per output of ALMA, ALMASWA, and the proposed Multi-Label ALMASWA when evaluated on the Edge IIoT dataset

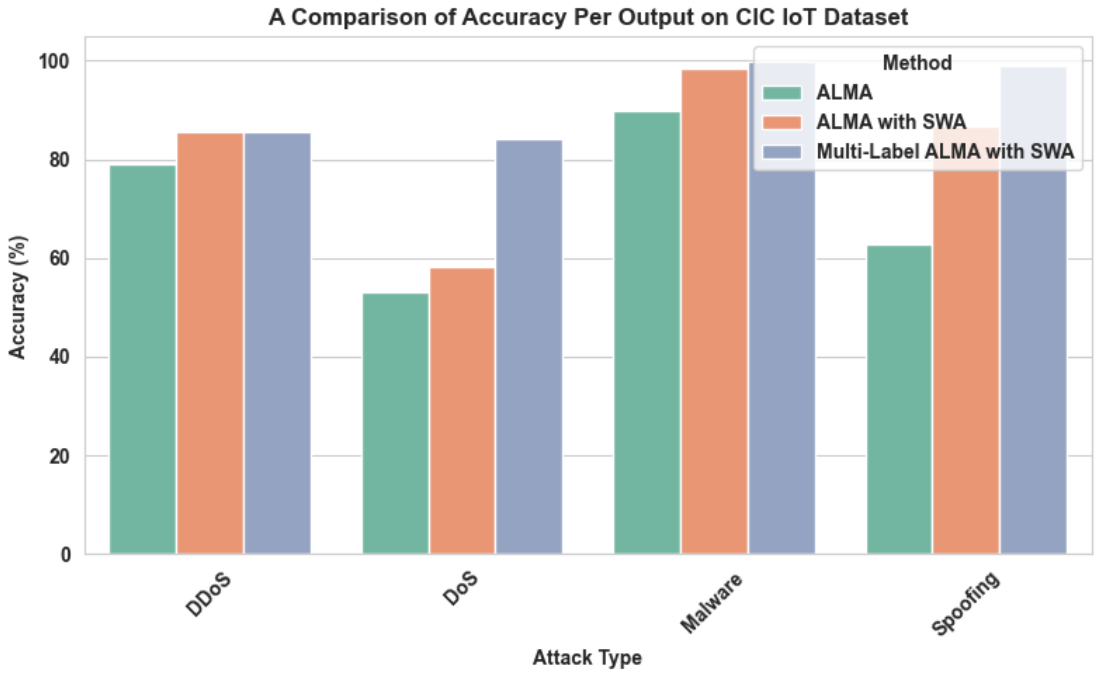


Figure 6.5: Comparing the accuracy per output of ALMA, ALMASWA, and the proposed Multi-Label ALMASWA when evaluated on the CIC IoT dataset

improvements are statistically meaningful. These results corroborate the robustness of SWA in enhancing ALMA's multi-label prediction accuracy and error minimization.

Figures 6.6 and 6.7 present the exact match and hamming of multi-label ALMA and multi-label ALMASWA when evaluated with each of the four datasets.

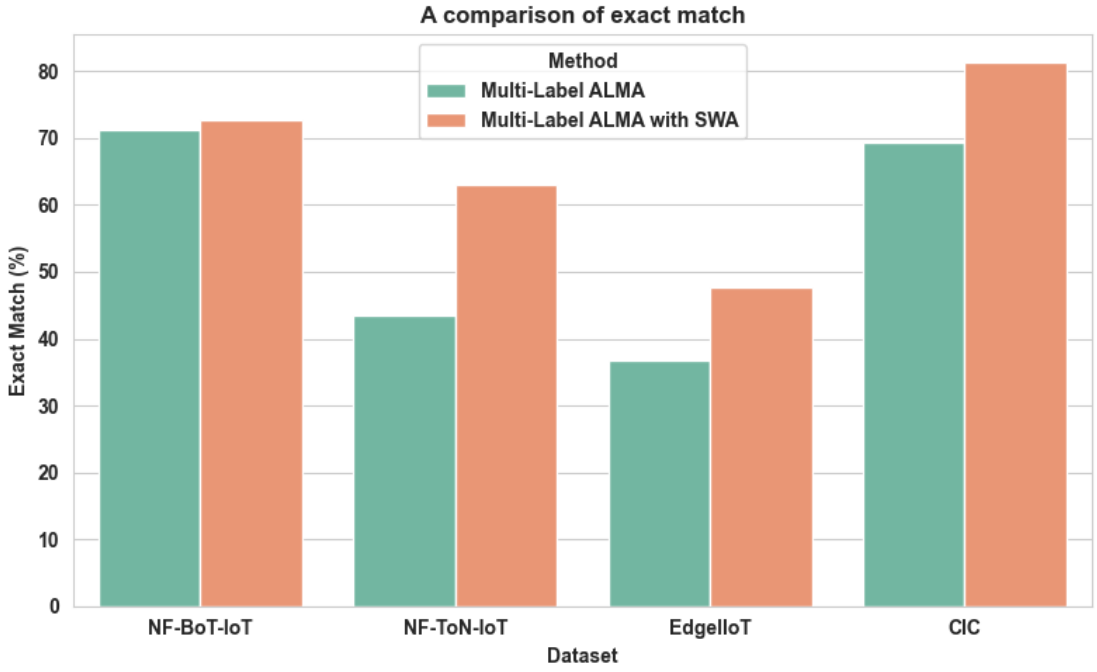


Figure 6.6: Comparing the exact match of Multi-Label ALMA, and Multi-Label ALMASWA across the four datasets

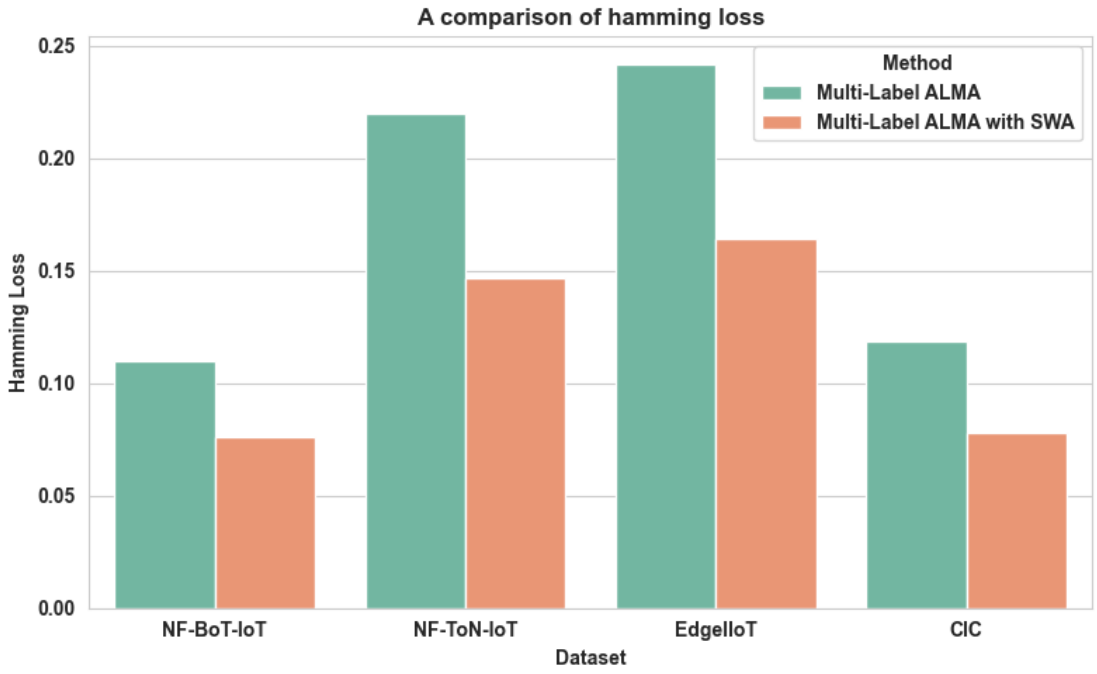


Figure 6.7: Comparing the hamming loss of Multi-Label ALMA, and Multi-Label ALMASWA across the four datasets

6.5.0.4 Performance on Simulated Concurrent Attacks

Multi-Label ALMA with SWA, as shown in Table 6.7, shows resilience in handling concurrent attacks, achieving reasonable precision, recall, exact match, and hamming loss values. For example, in NF-BoT-IoT, the model reaches a precision of 78.56% and a recall of 84.93% in detecting concurrent DDoS and DoS attacks, with an exact match score of 71.00% and a hamming loss of 0.1467. This indicates

Table 6.4: Comparing the performance of Multi-Label ALMA and Multi-Label ALMA with SWA

Algorithm	**Exact Match**	**Hamming Loss**
NF BoT IoT		
Multi-Label ALMA	71.34%	0.1099
Multi-Label ALMA with SWA	72.63%	0.0758
NF ToN IoT		
Multi-Label ALMA	43.41%	0.2197
Multi-Label ALMA with SWA	62.94%	0.1468
EdgeIIoT		
Multi-Label ALMA	36.85%	0.2419
Multi-Label ALMA with SWA	47.60%	0.1644
CIC IoT		
Multi-Label ALMA	69.30%	0.1188
Multi-Label ALMA with SWA	81.38%	0.0778

Table 6.5: A comparison of the mean of exact match and hamming loss of Multi-Label ALMA and Multi-Label ALMA with SWA after five runs using different random seed values

Algorithm	**Mean (Exact Match)**	**Mean (Hamming Loss)**
NF BoT IoT		
Multi-Label ALMA	71.28%	0.1099
Multi-Label ALMA with SWA	72.72%	0.0788
NF ToN IoT		
Multi-Label ALMA	43.27%	0.2201
Multi-Label ALMA with SWA	63.26%	0.1503
EdgeIIoT		
Multi-Label ALMA	36.86%	0.2423
Multi-Label ALMA with SWA	48.01%	0.1641
CIC IoT		
Multi-Label ALMA	69.08%	0.1199
Multi-Label ALMA with SWA	81.09%	0.0776

the model's ability to accurately detect multiple simultaneous attack vectors, an essential capability in modern cybersecurity landscapes.

6.5.0.5 Resource Utilization

As shown in Table 6.8, the proposed Multi-Label ALMA with SWA is resource-efficient, consuming minimal memory and CPU usage across all datasets. For instance, NF-BoT-IoT requires only 0.02 MB of memory and 0.82% CPU usage per round, making it highly suitable for IoT environments where computational resources are limited.

Table 6.6: The T-Statistics and P-Value of the Exact Match and Hamming Loss between Multi-Label ALMA and Multi-Label ALMA with SWA

Metric	T-Statistics	P-Value
NF BoT IoT		
Exact Match	3.6538	0.0217
Hamming Loss	-33.8797	0.0000
NF ToN IoT		
Exact Match	46.3732	0.0000
Hamming Loss	-50.6833	0.0000
EdgeIIoT		
Exact Match	19.2927	0.0000
Hamming Loss	-42.3625	0.0000
CIC IoT		
Exact Match	83.5527	0.0000
Hamming Loss	-116.1132	0.0000

Table 6.7: Performance of Multi-Label ALMA with SWA on simulated concurrent attacks

Concurrent Attacks	Precision	Recall	Exact Match	Hamming Loss
NF BoT IoT				
DDoS & DoS	78.56%	84.93%	71.00	0.1467
NF ToN IoT				
Injection & XSS	77.42%	80.24%	69.85	0.1817
EdgeIIoT				
Injection & Malware	71.04%	77.73%	60.34	0.2093
CIC IoT				
DDoS & DoS	76.89%	75.34%	68.78	0.2145

Table 6.8: The average resource usage of our proposed model after running rounds of the same experiment

Processing time per sample (s)	Model Memory Usage (MB)	CPU Usage (%)	Peak Memory of the system (MB)	Final Memory of the system (MB)
NF BoT IoT				
0.000227	0.02	0.82	87.38	39.36
NF ToN IoT				
0.000244	0.02	0.45	200.62	90.28
EdgeIIoT				
0.000691	0.04	0.05	63.21	10.18
CIC IoT				
0.000727	0.05	0.25	200.84	28.94

6.5.0.6 The Impact of Label Order on the Performance of the Classifier

The analysis of Multi-Label ALMA with SWA performance based on label order further underscores the method's effectiveness in accurately detecting and differentiating between multiple attack classes across various IoT datasets. Tables 6.9 and 6.10 present the findings on exact match, hamming loss, and accuracy per attack class based on the optimal and suboptimal label order configurations.

The results in Table 6.9 show that label order has a considerable impact on exact match and hamming loss values for each dataset. For instance, in the NF-BoT-IoT dataset, the best-performing label order ("Reconnaissance - Theft - DDoS - DoS") yields an exact match of 72.69% and a hamming loss of 0.0758. Conversely, the worst-performing order ("DoS - DDoS - Theft - Reconnaissance") reduces the exact match to 42.68% with an increased hamming loss of 0.1906. This pattern of performance degradation with suboptimal label order is consistent across datasets, emphasizing that the sequence of attack labels plays a critical role in maximizing detection accuracy and minimizing error.

In the NF-ToN-IoT dataset, optimal label ordering achieves an exact match of 62.95% and a hamming loss of 0.1468. The exact match drops significantly to 29.81%, and hamming loss rises to 0.2578 with a less favorable label sequence. Similar trends are observed in EdgeIoT and CIC, where performance varies with label order, underscoring the importance of selecting an optimal label sequence to enhance multi-label classification results.

Table 6.10 further demonstrates how label ordering influences accuracy across specific attack classes. For instance, in the NF-BoT-IoT dataset, the best label order yields 90.44% accuracy for DDoS, whereas the worst label order reduces this accuracy to 90.26%. More drastic differences are observed in the DoS attack class, where the accuracy plummets from 90.35% in the optimal order to 47.24% in the suboptimal order. This suggests that certain attack types are particularly sensitive to label order, with their detection accuracy being significantly impacted by the sequencing of labels.

Similarly, in NF-ToN-IoT, the accuracy for the DDoS class decreases from 87.05% in the optimal order to 78.92% in the worst order. The CIC dataset exhibits less variation across label orders, with high accuracy maintained for most classes regardless of order, indicating that certain datasets may be more resilient to label order variations. However, even in EdgeIoT, where overall accuracy is less affected, there is still a noticeable drop in performance for the Injection class from 80.07% in the best order to 64.11% in the worst order.

Table 6.9: The exact match and hamming loss of the best two (1 & 2) and worst two (3 & 4) label orders

#	Label Order	Exact Match	Hamming Loss
NF BoT IoT			
1	Reconnaissance - Theft- DDoS- DoS	72.69%	0.0758
2	Reconnaissance - Theft- DoS- DDoS	72.68%	0.0758
3	DoS - Theft - DDoS - Reconnaissance	42.68%	0.1881
4	DoS - DDoS - Theft - Reconnaissance	42.68%	0.1906
NF ToN IoT			
1	Injection - DDoS - XSS - Password	62.95%	0.1468
2	Injection - DDoS - Password - XSS	62.94%	0.1469
3	XSS - Injection - Password - DDoS	31.15%	0.2541
4	Password - XSS - DDoS - Injection	29.81%	0.2578
EdgeIIoT			
1	Information Gathering - Injection - Malware - DDoS	58.82%	0.1483
2	DDoS - Injection - Malware - Information Gathering	58.44%	0.1277
3	Malware - Injection - Information Gathering - DDoS	51.47%	0.1855
4	Malware - Injection - DDoS - Information Gathering	50.86%	0.1864
CIC IoT			
1	DDoS - Spoofing - DoS - Malware	81.53%	0.0777
2	DDoS - Spoofing - Malware - DoS	81.51%	0.0778
3	DoS - Spoofing - DDoS - Malware	44.67%	0.1436
4	DoS - Malware - Spoofing - DDoS	44.57%	0.1439

Table 6.10: The Accuracy Per Attack Class based on the best (Accuracy 1) and worst (Accuracy 2) label orders

Attack Class	Accuracy 1	Accuracy 2
NF BoT IoT		
Reconnaissance	89.22%	86.57%
DDoS	90.44%	90.26%
DoS	90.35%	47.24%
Theft	99.67%	99.68%
NF ToN IoT		
Infection	72.83%	73.40%
DDoS	87.05%	78.92%
Password	88.64%	51.89%
XSS	92.72%	92.65%
EdgeIIoT		
DDoS	87.13%	64.11%
Injection	80.07%	80.02%
Information Gathering	86.62%	86.64%
Malware	80.41%	89.94%
CIC IoT		
DDoS	85.69%	85.48%
DoS	84.33%	58.08
Malware	99.87%	99.84%
Spoofing	99.03%	99.05%

6.6 Discussion

The study demonstrates the effectiveness of integrating SWA into the Multi-Label ALMA for IoT security, particularly in multi-label attack detection scenarios. Incorporating SWA significantly enhances key performance metrics, including accuracy, F1 score, exact match, and hamming loss. These improvements highlight SWA's ability to stabilize model predictions and bolster robustness across various IoT datasets and diverse attack types.

Statistical analysis confirms that the performance gains from SWA are significant and not due to random variation, underscoring its practical value in real-world applications. The Multi-Label ALMA with SWA excels in detecting concurrent attacks while maintaining resource efficiency, making it highly suitable for resource-constrained IoT environments. The model achieves low memory and CPU usage, facilitating its deployment in settings where computational resources are limited without compromising detection performance.

Resource efficiency is another significant advantage of the proposed technique. By focusing computational power on critical attacks while maintaining low memory and CPU usage, the proposed system aligns well with the constraints of IoT environments. The ability to adaptively reweigh classifiers based on specific dataset needs ensures high accuracy and resilience in real-time applications, where both speed and precision are paramount.

The study also highlights the sensitivity of Multi-Label ALMA with SWA to label order, which affects exact match, hamming loss, and class-specific accuracy. Optimal label ordering significantly enhances classification accuracy, particularly in datasets with high label correlation, by maximizing detection accuracy and reducing error rates. This insight is crucial for tailoring model performance to specific deployment environments and optimizing detection for high-priority threats.

Future work may explore automated feature selection techniques and incorporate temporal and spatial dependencies in IoT traffic to enhance feature interactions and improve overall classification performance. These advancements could further optimize model efficiency and accuracy, solidifying the framework's applicability in dynamic and complex IoT security landscapes.

6.7 Conclusion

This paper has proposed a novel approach to detecting concurrent cyber-attacks in IoT environments by leveraging ALMA integrated with SWA. The motivation for this research stems from the growing need for effective IDSs that can handle concurrent attacks in real-time, while operating within the computational constraints of IoT devices.

Throughout the paper, we discussed the limitations of existing IDS models that typically struggle with multi-label classification and the computational demands of IoT systems. By integrating SWA into ALMA and using it to build an online chain classifier, we introduced a method that not only efficiently handles multi-label classification but also maintains low computational resource usage.

Our experimental results, derived from four distinct IoT datasets, demonstrated that the proposed system can detect concurrent attacks by capturing the inter-label relationship among attacks. Additionally, the system's ability to process streaming data and update classifiers incrementally without retraining from scratch makes it a suitable solution for real-time attack detection in IoT environments.

This research has significant implications for the future of IoT cybersecurity, as it offers a practical solution for mitigating the risks posed by concurrent cyber-attacks.

However, there are some limitations to this study. While the results are promising, the performance of the proposed model can be influenced by factors such as the order of labels and the diversity of attack types. The optimal label ordering was found to significantly affect the accuracy, which suggests that further research is needed to develop automated methods for label ordering and to better understand the interplay between different attack types. Additionally, the study focused on four IoT datasets, and the proposed model's generalization to other datasets with different attack scenarios or environmental conditions could be explored further.

Future work could aim to enhance the feature selection process by incorporating more advanced techniques, such as automated feature engineering, to improve classification accuracy. Additionally, exploring the temporal and spatial dependencies within IoT traffic could further enhance the model's predictive capabilities. The integration of anomaly detection techniques alongside the multi-label classification framework could provide additional insights into identifying previously unknown

attack patterns. Lastly, extending this framework to handle encrypted traffic and scaling it to accommodate larger, heterogeneous IoT networks would be essential steps toward improving its real-world applicability.

References

[1] C. Gentile, "A new approximate maximal margin classification algorithm", *Advances in Neural Information Processing Systems*, vol. 13, 2000.

[2] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization", *arXiv preprint arXiv:1803.05407*, 2018.

[3] E. Hallaji, R. Razavi-Far, and M. Saif, "Expanding analytical capabilities in intrusion detection through ensemble-based multi-label classification", *Computers & Security*, vol. 139, p. 103 730, 2024.

[4] Y. Huang, J. Gou, Z. Fan, Y. Liao, and Y. Zhuang, "A multi-label network attack detection approach based on two-stage model fusion", *Journal of Information Security and Applications*, vol. 83, p. 103 790, 2024.

[5] A. Sharma, S. Rani, D. K. Sah, Z. Khan, and W. Boulila, "Homlc-hyperparameter optimization for multi-label classification of intrusion detection data for internet of things network", *Sensors*, vol. 23, no. 19, p. 8333, 2023.

[6] I. Karunanayake, M. AlSabah, N. Ahmed, and S. Jha, "Examining the rat in the tunnel: Interpretable multi-label classification of tor-based malware", *arXiv preprint arXiv:2409.16639*, 2024.

[7] R. Qaddoura, A. M. Al-Zoubi, H. Faris, and I. Almomani, "A multi-layer classification approach for intrusion detection in iot networks based on deep learning", *Sensors*, vol. 21, no. 9, p. 2987, 2021.

[8] S. K. Sahu, D. P. Mohapatra, J. K. Rout, K. S. Sahoo, Q.-V. Pham, and N.-N. Dao, "A lstm-fcnn based multi-class intrusion detection using scalable framework", *Computers and Electrical Engineering*, vol. 99, p. 107 720, 2022.

[9] L. Koc, T. A. Mazzuchi, and S. Sarkani, "A network intrusion detection system based on a hidden naı¨ve bayes multiclass classifier", *Expert Systems with Applications*, vol. 39, no. 18, pp. 13 492–13 500, 2012.

[10] S. M. H. Bamakan, H. Wang, and Y. Shi, "Ramp loss k-support vector classification-regression; a robust and sparse multi-class approach to the intrusion detection problem", *Knowledge-Based Systems*, vol. 126, pp. 113–126, 2017.

[11] I. S. Thaseen and C. A. Kumar, "Intrusion detection model using fusion of chi-square feature selection and multi class svm", *Journal of King Saud University-Computer and Information Sciences*, vol. 29, no. 4, pp. 462–472, 2017.

[12] H. Alaiz-Moreton, J. Aveleira-Mata, J. Ondicol-Garcia, A. L. Muñoz-Castañeda, I. García, and C. Benavides, "Multiclass classification procedure for detecting attacks on mqtt-iot protocol", *Complexity*, vol. 2019, no. 1, p. 6 516 253, 2019.

[13] D. Rani, N. S. Gill, P. Gulia, and J. M. Chatterjee, "An ensemble-based multiclass classifier for intrusion detection using internet of things", *Computational Intelligence and Neuroscience*, vol. 2022, no. 1, p. 1 668 676, 2022.

[14] J.-R. Jiang and C.-L. Li, "Binary-and multi-class network intrusion detection with adaptive synthetic sampling and deep learning", in *2021 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, IEEE, 2021, pp. 1–2.

[15] S.-M. Tseng, Y.-Q. Wang, and Y.-C. Wang, "Multi-class intrusion detection based on transformer for iot networks using cic-iot-2023 dataset", *Future Internet*, vol. 16, no. 8, p. 284, 2024.

[16] T.-T.-H. Le, Y. E. Oktian, and H. Kim, "Xgboost for imbalanced multiclass classification-based industrial internet of things intrusion detection systems", *Sustainability*, vol. 14, no. 14, p. 8707, 2022.

[17] Y. Yuan, L. Huo, and D. Hogrefe, "Two layers multi-class detection method for network intrusion detection system", in *2017 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, 2017, pp. 767–772.

[18] A. K. Silivery, R. M. R. Kovvur, R. Solleti, L. S. Kumar, and B. Madhu, "A model for multi-attack classification to improve intrusion detection performance using deep learning approaches", *Measurement: Sensors*, vol. 30, p. 100 924, 2023.

[19] R. Yao, N. Wang, P. Chen, D. Ma, and X. Sheng, "A cnn-transformer hybrid approach for an intrusion detection system in advanced metering infrastructure", *Multimedia Tools and Applications*, vol. 82, no. 13, pp. 19 463–19 486, 2023.

[20] M. V. Taydea, R. B. Adhaob, and V. K. Pachghare, "Ensemble method for multi-label classification on intrusion detection system", in *Recent Advances in Material, Manufacturing, and Machine Learning*, CRC Press, 2023, pp. 761–767.

[21] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "Cnn-lstm: Hybrid deep neural network for network intrusion detection system", *IEEE Access*, vol. 10, pp. 99 837–99 849, 2022.

[22] M. M. Hassan, A. Gumaei, A. Alsanad, M. Alrubaian, and G. Fortino, "A hybrid deep learning model for efficient intrusion detection in big data environment", *Information Sciences*, vol. 513, pp. 386–396, 2020.

[23] K. Jin, L. Zhang, Y. Zhang, D. Sun, and X. Zheng, "A network traffic intrusion detection method for industrial control systems based on deep learning", *Electronics*, vol. 12, no. 20, p. 4329, 2023.

[24] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "Rtids: A robust transformer-based approach for intrusion detection system", *IEEE Access*, vol. 10, pp. 64 375–64 387, 2022.

[25] Z. Long, H. Yan, G. Shen, X. Zhang, H. He, and L. Cheng, "A transformer-based network intrusion detection approach for cloud security", *Journal of Cloud Computing*, vol. 13, no. 1, p. 5, 2024.

[26] J. Leo and J. Kalita, "Incremental deep neural network learning using classification confidence thresholding", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 7706–7716, 2021.

[27] H. A. Gabbar, O. G. Adegboro, A. Chahid, and J. Ren, "Incremental learning-based algorithm for anomaly detection using computed tomography data", *Computation*, vol. 11, no. 7, p. 139, 2023.

[28] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets", *Mobile networks and applications*, pp. 1–14, 2022.

[29] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning", *IEEE Access*, vol. 10, pp. 40 281–40 306, 2022.

[30] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment", *Sensors*, vol. 23, no. 13, p. 5941, 2023.

This page has been intentionally left blank.

Chapter 7

Chapter 7 is published as:

P. R. Agbedanu, S. J. Yang, R. Musabe, I. Gatare, J. Rwigema, "A Scalable Approach to IoT and IIoT Security: Evaluating Adaptive SAMKNN for Zero-Day Attack Detection," *Sensors*, 25(1), 216.

A scalable approach to Internet of Things and Industrial Internet of Things security: Evaluating Adaptive Self-Adjusting Memory K-Nearest Neighbor for Zero-Day attack detection

Authors: Promise Ricardo Agbedanu, Sanchieh Jay Yang, Richard Musabe, Ignace Gatare, James Rwigema

Abstract: The Internet of Things (IoT) and Industrial Internet of Things (IIoT) have drastically transformed industries by enhancing efficiency and flexibility, but also introduced substantial cybersecurity risks. The rise of zero-day attacks, which exploit unknown vulnerabilities, poses significant threats to these interconnected systems. Traditional signature-based Intrusion Detection Systems (IDS) are insufficient for detecting such attacks due to their reliance on pre-defined attack signatures. This study investigates the effectiveness of Adaptive SAMKNN, an adaptive k-nearest neighbor with self-adjusting memory (SAM), in detecting and responding to various attack types in Internet of Things (IoT) environments. Through extensive testing, our proposed method demonstrates superior memory efficiency, with a memory footprint as low as 0.05 MB, while maintaining high accuracy and F1 scores across all datasets. The proposed method also recorded a detection rate of 1.00 across all simulated zero-day attacks. In scalability tests, the proposed technique sustains its performance even as data volume scales up to 500,000 samples, maintaining low CPU and memory consumption. However, while it excels under gradual, recurring, and incremental drift, its sensitivity to

sudden drift highlights an area for further improvement. This study confirms the feasibility of Adaptive SAMKNN as a real-time, scalable, and memory-efficient solution for IoT and IIoT security, providing reliable anomaly detection without overwhelming computational resources. Our proposed method has the potential to significantly increase the security of IoT and IIoT environments by enabling real-time, scalable, and efficient detection of sophisticated cyber threats, thereby safeguarding critical interconnected systems against emerging vulnerabilities.

7.1 Introduction

The Internet of Things (IoT) and the Industrial Internet of Things (IIoT) have revolutionized how industrial systems are designed, operated, and managed, increasing efficiency, reliability, and flexibility. However, these systems' increased connectivity has also presented new and complex security challenges, as they are more vulnerable to cyber attacks that can disrupt critical infrastructure and compromise sensitive data [1].

In recent years, there has been a surge in the number of cyber attacks targeting industrial systems, raising concerns about the security of the IIoT [2], [3]. Cyber attacks pose a significant threat to the IIoT and IoT, leading to several studies trying to solve this problem [4]–[7]. Cyber attacks like malware can infiltrate an IIoT system and remain undetected for long periods, leading to significant economic losses, safety risks, and potential environmental damage. Malware can also spread rapidly across interconnected systems, making detecting and containing the attack even more challenging. Designing effective and efficient systems for detecting these attacks in the IIoT is essential to mitigate these risks.

The proliferation of IoT devices across diverse domains has significantly increased the frequency and complexity of cyberattacks targeting these networks [8]. A particularly concerning threat is the emergence of zero-day attacks, which exploit previously unknown vulnerabilities before they can be detected and mitigated [8]. Traditional signature-based intrusion detection systems (IDS) struggle to detect such novel attacks effectively, as they rely on prior knowledge of attack patterns [9]. Researchers have explored machine learning (ML) techniques to develop more adaptive and intelligent IDS for IoT environments [10].

Most intrusion detection methods are not well-suited to the unique requirements

of IoT and IIoT because they are computationally expensive, need to be retrained in an off-production mode to adapt to new attack trends, and cannot detect zero-day attacks. Additionally, most intrusion detection methods are based on offline ML techniques. Offline ML-based approaches are often too complex and computationally expensive to be implemented in computationally constrained devices like IoT and IIoT. Secondly, ML-based intrusion detection systems are not real-time adaptive. Because they are trained using historical data, they cannot adapt to the dynamic nature of the IIoT ecosystem. Finally, updating offline models requires the models to be retrained on new data, which introduces delays between when a new attack sample appears and when the model is updated to detect those new samples. Due to the critical nature of IIoT systems, this short delay can make these systems vulnerable to zero-day attacks, which may end up causing damage to these production systems.

Based on the above-mentioned drawbacks of current systems that detect cyber-attacks in IoT and IIoT, it is important to use a more effective and efficient solution to detect attacks within these ecosystems. This solution should be able to detect zero-day attacks and be computationally inexpensive. Online ML models, in particular, offer a promising approach for detecting zero-day attacks in resource-constrained IoT systems [11]. By continuously learning from network traffic data, these models can adapt to evolving attack patterns without the need for extensive retraining [11].

This paper proposes an IDS for IoT and IIoT using an online ML technique to build an adaptive IDS capable of identifying previously unseen threats in real-time, thereby enhancing the security of IoT and IIoT networks.

This article is a revised and expanded version of a paper entitled "An Online Adaptive Approach to Detecting Zero-day Attacks in IoT and IIoT Systems" [12], which was presented at a workshop at the IEEE Global Communications Conference in Cape Town, South Africa, from 8-12 December 2024. This expanded version contains about 60% new work, which includes the addition of a new dataset used for validating this study. In the workshop paper, we used a Generative Adversarial Network (GAN) to introduce some synthetic attacks into the respective datasets. However, this work presents two types of zero-attacks, the first being unseen attack classes and the second being synthetic attacks created using a Conditional Tabular Generative Adversarial Network (CTGAN). The expanded validation also includes an expanded validation that focuses on scalability, false

positives, detection rate, performance under drift, resource utilization, ablation study, and expanded statistical analysis. This study also analyzed the complexity of the proposed classifier.

The contributions of this research can be summarized as follows:

- **Development of an Adaptive online k-Nearest Neighbors (kNN) with Self Adjusting Memory (SAM) Classifier:** Introduces the proposed classifier, an enhanced version of the SAMKNN classifier tailored for online learning scenarios. Our proposed adaptive SAMKNN dynamically adjusts its memory allocation between Short-Term Memory (STM) and Long-Term Memory (LTM) based on real-time performance metrics, enabling it to effectively handle evolving data distributions inherent in IoT and IIoT environments.
- **Dynamic Memory Allocation Based on Performance Metrics:** Implements a mechanism where the proportion of memory allocated to STM and LTM is dynamically adjusted based on the classifier's recent performance. By monitoring metrics such as accuracy over a sliding window, the system can allocate more resources to STM during periods of rapid concept drift and shift towards LTM when data distributions stabilize.
- **Efficient Memory Management for Resource-Constrained Environments:** Employs Python's double-ended queue (deque) for managing STM, allowing efficient append and pop operations with fixed maximum lengths. Additionally, it uses NumPy arrays for LTM to facilitate rapid computations and memory compression, making the system suitable for the resource-constrained nature of IoT and IIoT devices.
- **Comprehensive Performance Evaluation and Memory Adjustment Strategies:** Utilizes both performance-based increases and decreases in memory allocation, supported by predefined thresholds and a cool-down mechanism. This bidirectional adjustment ensures the classifier can balance adaptability with stability, maintaining high accuracy even as threat patterns evolve.

The remainder of this paper is organized as follows: Section 2 gives an overview of the related works considered in this study. In Section 3, we present our proposed methodology. Section 4 presents the experimental design of this study, with the experimental results presented in Section 5 and discussions presented in Section 6. The study is concluded in Section 7.

7.2 Related Work

Detecting zero-day attacks in IoT environments has been an area of concern for many years. As such, several works have been done in the quest to solve this problem. Popoola et al., [13], proposed an optimal deep neural network (DNN) architecture to detect zero-day attacks in IoT systems. The DNN was used with federated ML, where the federated algorithm is used to aggregate local model updates. Similarly, [14] developed a framework based on federated learning to detect zero-day botnet attacks in IoT systems. As part of the contributions of their work, [14] also developed a novel aggregation algorithm that handles model aggregation better in IoT systems. Due to the limited computational power, memory, and battery life of IoT devices, training and updating complex DNN models or performing federated learning on these devices can be resource-intensive.

Hairab et al., [15] evaluated the performance of Convolutional Neural Networks (CNN) in detecting zero-day attacks in IoT using two regularization techniques. The authors reported that the use of regularization techniques increased the performance of the proposed system, with the ability to detect zero-day attacks. Similarly, [16] also proposed an anomaly detection system for IoT systems using CNN with regularization methods. In another study, [17] proposed an IDS based on CNN that could detect seen and unseen attacks in the Internet of Vehicles ecosystem. The proposed IDS was developed at the data processing layer of the Internet of Vehicles ecosystem. The rationale behind the development of the IDS at this layer is to speed up the rate of detection of the IDS. The authors also compared the proposed method to SVM and RF, of which the proposed method recorded a higher detection accuracy. IoT devices are highly heterogeneous, varying in terms of hardware, software, and functionality. A CNN model trained on one type of device or environment may not perform well on others.

Federated learning has emerged as a promising approach to enhance detection capabilities against zero-day attacks by sharing attack information among multiple IoT networks [14], [18]. By utilizing federated learning, researchers have developed frameworks that improve the detection of zero-day attacks while preserving user privacy and minimizing communication overhead [19]. These frameworks leverage deep learning (DL) algorithms to detect and resist botnet attacks in IoT networks, showcasing superior performance in identifying new and evolving threats

compared to traditional centralized approaches [14]. The use of federated learning in intrusion detection systems not only enhances detection performance but also addresses resource constraints in IoT devices, offering scalable and cost-effective solutions for early attack detection and patch creation [20]. To detect zero-day attacks, [21] developed an IDS model based on federated incremental learning that aggregates knowledge from different detectors and then updates the model in an incremental manner. IoT devices often generate heterogeneous data due to differences in device types, manufacturers, and deployment environments. Aggregating models effectively in federated learning to account for this heterogeneity is complex and can lead to sub-optimal performance.

To combat zero-day attacks, a hybrid approach combining ML and DL techniques has been proposed for effective detection and mitigation [22]–[24]. By leveraging ML classifiers and Deep Neural Networks (DNNs) trained on diverse data streams, along with Deep Reinforcement Learning (DRL) for dynamic model selection, a robust defense mechanism against zero-day malware in IoT devices is established. This hybrid framework not only enhances detection rates but also minimizes false positives and false negatives, achieving a remarkable 99% detection rate with minimal errors [22]. Saurabh et al., [25] proposed a hybrid IDS using supervised and unsupervised ML approaches that can detect both known and unknown attacks. Hybrid models combining ML, DL, and DRL are computationally intensive. IoT devices often have limited processing power, memory, and battery life, making it challenging to run these complex models locally.

Research has shown the effectiveness of ensemble learning in detecting zero-day attacks in IoT networks [14], [26], [27]. By combining different base anomaly detectors using conventional ML algorithms, ensemble learning can provide highly accurate zero-day attack detection even without prior labeled attack data. Moreover, the use of ensemble learning, particularly with Random Forest (RF) and Extreme Gradient Boosting (XGB) algorithms, has been identified as top-performing in detecting zero-day attacks in IoT systems, outperforming previous methods and enhancing the performance of machine learning models [27]. In an earlier study, [28] developed an ensemble classifier using processed data packets that can detect anomalies and protect IoT systems against zero-day attacks. Additionally, leveraging ensemble learning in a federated framework for IoT networks can achieve superior model aggregation without compromising user privacy, showcasing its

efficacy in zero-day botnet attack detection [14]. Ahmad et al., [29] used an ensemble of DL classifiers to build an IDS. The proposed IDS is trained using four benchmark datasets while testing the model with unknown attacks to validate the system's performance in detecting zero-day attacks. Ensemble learning models, particularly those involving multiple algorithms like RF and XGB, can be computationally intensive. This complexity increases the resource requirements, which can be problematic for resource-constrained IoT devices.

Farrukh et al., [30], focused on proposing a framework based on packet-level data by extracting spatial and temporal patterns from network traffic. The authors also used stacking and sub-clustering techniques to help effectively detect unknown attacks. [27] proposed a computational framework that includes feature selection through fuzzification. The authors reported that RF and Extreme Gradient Boosting (XGB) were the top-performing algorithms that could detect zero-day attacks. In another study, [31] used online reinforcement learning (RL) to propose a framework that learns the correct Moving Target Defense (MTD) to mitigate heterogeneous zero-day attacks in single-board computers (SBCs). The proposed framework works by considering behavioral fingerprinting to represent SBCs' states and RL to learn MTD techniques that mitigate each malicious state. The results of the study show that the proposed framework mitigates all attacks except rootkits while consuming less than 1 MB of storage and approximately 10% of RAM. [32] designed an IDS framework using transfer learning (TL) that could detect both known and zero-day attacks. The transfer learning model used in their study was based on a CNN. Techniques such as stacking, sub-clustering, and TL with CNNs are computationally intensive. IoT devices, especially single-board computers (SBCs), typically have limited processing power, memory, and storage.

Zero-day attacks targeting IoT devices have become a significant concern due to the vulnerabilities in interconnected devices [33]. To combat these threats, researchers have proposed innovative solutions like a fine-grained central processing unit (CPU) security engine, μThingNet, which leverages DL and power analysis to detect unknown malware variants with a high detection rate of 97.49% [34]. Additionally, [33] used honeypot systems to detect malicious activities and analyze zero-day attacks, benefiting from filtering malicious traffic to identify and thwart attacks effectively. Furthermore, a game-theoretic approach has been introduced to strategically allocate honeypots over networks, considering the deceptive nature of

attackers and the impact of zero-day vulnerabilities on defense mechanisms [35]. ML techniques have also emerged as a powerful tool for detecting zero-day attacks by analyzing patterns in network traffic and user behavior, enhancing cybersecurity defenses against these elusive threats [24]. Implementing a fine-grained CPU security engine may require specialized hardware modifications, making it difficult to deploy on existing IoT devices.

Tables 7.1 and 7.2 provide a summary that compares existing methods to our proposed method.

Table 7.1: Comparison of Existing Methods and the Proposed Adaptive SAMKNN for Zero-Day Attack Detection in IoT and IIoT Environments

Reference	Approach	Zero-Day Detection	Resource Requirements	Performance Metrics	Strengths	Limitations
[14]	Novel aggregation algorithm for federated learning to detect zero-day botnet attacks	Effective	Moderate computational resources	Superior model aggregation and detection performance	Enhances detection while preserving privacy, minimizes communication overhead	Complexity in handling heterogeneous data across different IoT devices
[15]	Utilizes CNN with regularization techniques for anomaly detection	Effective	Moderate to high computational resources	Improved detection performance with regularization	Enhanced detection capability with regularization	May not generalize well across heterogeneous IoT environments
[16]	Anomaly detection using CNN enhanced with regularization methods	Effective	Similar to Hairab et al., 2022	High detection accuracy for zero-day attacks	Improved robustness against overfitting and better generalization	High resource consumption makes it less suitable for resource-constrained devices
[17]	IDS at the data processing layer of the Internet of Vehicles, compared with SVM and RF	Effective	Moderate computational resources	Higher detection accuracy than SVM and RF	Faster detection rate by operating at the data processing layer	Performance may degrade on different device types due to CNN's dependency on specific architectures
[31]	Uses online RL for Moving Target Defense (MTD) with behavioral fingerprinting	Effective (except rootkits)	Low storage (1 MB) and 10% RAM usage	Successfully mitigates all tested attacks except rootkits	Low resource consumption, adaptable MTD strategies	Unable to mitigate rootkits, limited by RL model's adaptability to certain attack types

Table 7.2: Comparison of Existing Methods and the Proposed Adaptive SAMKNN for Zero-Day Attack Detection in IoT and IIoT Environments

Reference	**Approach**	**Zero-Day Detection**	**Resource Requirements**	**Performance Metrics**	**Strengths**	**Limitations**
[22], [23], [24]	Combines ML classifiers, DNNs, and DRL for dynamic model selection	Highly Effective	Highly computationally intensive	99% detection rate with minimal errors	Robust defense mechanism with high detection rates and minimal false positives/negatives	Computationally intensive, challenging for deployment on resource-constrained IoT devices
[29]	Builds IDS using an ensemble of deep learning classifiers trained on benchmark datasets	Effective	High computational and memory usage	High performance in detecting zero-day attacks	Validated on multiple datasets, effective against unknown attacks	High resource consumption limits deployment on resource-constrained IoT devices
[32]	IDS framework using TL based on CNN to detect known and zero-day attacks	Effective	High computational resources due to CNN and TL	High detection rates for both known and zero-day attacks	Leverages pre-trained models for improved detection capabilities	High resource demands make it unsuitable for many IoT devices
Proposed Method	**Adaptive k-Nearest Neighbors with Self-Adjusting Memory (SAM) for anomaly detection**	**Highly Effective**	**Low CPU and memory consumption (as low as 0.05 MB)**	**High accuracy and F1 scores; Detection rate of 1.00 for zero-day attacks**	**Real-time, scalable, memory-efficient, maintains performance with large data volumes**	**Sensitive to sudden data drift, requiring further enhancement for handling abrupt changes**

7.3 Proposed Methodology

The proposed IDS is designed to identify zero-day attacks in IoT and IIoT environments by leveraging the proposed classifier. The system is structured into five key modules: Preprocessing, proposed classifier, Attack Detection, Alert and Response, and Performance Monitoring. Each module performs a distinct function to ensure accurate, efficient, and adaptable detection of known and unknown attacks. The following sections describe each module in detail. Figure 7.1 represents a block diagram of our proposed system.

7.3.1 Preprocessing Module

The initial step involves loading the dataset and performing essential preprocessing to enhance its suitability for machine learning applications. This includes:

- **Feature Selection:** Irrelevant and non-informative columns such as source and destination IP addresses, attack types, and redundant labels are removed to streamline the feature set. This focuses the analysis on the most pertinent variables that contribute to intrusion detection.
- **Data Shuffling:** To eliminate any inherent order or bias present in the dataset, the data is randomly shuffled. This ensures that the training and evaluation processes are not influenced by the sequence of the data, promoting a more generalized and robust model performance.
- **Dataset Iterator Preparation:** For online learning scenarios, where models are updated incrementally as new data arrives, the dataset is prepared for streaming. An iterator is created to simulate real-time data flow, allowing the model to process data sequentially in an online manner.

7.3.1.1 Feature Extraction

In developing the proposed system, a set of features was extracted from the datasets to effectively capture the characteristics of network traffic. The feature extraction process involved selecting variables that are indicative of normal and malicious network behavior while excluding those that could introduce noise or redundancy. The dataset initially contained several columns, including IPV4_SRC_ADDR,

IPV4_DST_ADDR, Attack, and Label. These columns were excluded from the feature set for the following reasons:

- IP Addresses (IPV4_SRC_ADDR and IPV4_DST_ADDR): The IP addresses were removed to ensure the model focuses on behavioral patterns rather than specific network endpoints, enhancing the generalization of the IDS across different network configurations.
- Attack Type (Attack): The 'Attack' column, which specifies the type of attack, was excluded to prevent the model from being biased toward known attack signatures. This exclusion is crucial for maintaining the model's ability to detect zero-day attacks that may not conform to predefined attack types.
- Label (Label): The 'Label' column serves as the target variable, indicating whether a network flow is benign or malicious. It was separated from the feature set to facilitate supervised learning.

The remaining columns after the exclusion are L4_SRC_PORT, L4_DST_PORT, PROTOCOL, L7_PROTO, IN_BYTES, OUT_BYTES, IN_PKTS, OUT_PKTS, TCP_FLAGS, and
FLOW_DURATION_MILLISECONDS.

The selection of features was driven by the need to balance comprehensiveness and computational efficiency, particularly given the resource-constrained nature of IoT and IIoT devices. The rationale for selecting the aforementioned features includes:

- Relevance to Intrusion Detection: The chosen features are closely related to network behavior and are effective in capturing anomalies indicative of cyber-attacks. For instance, unusual patterns in flow duration or packet counts can signal potential intrusions.
- Avoidance of Redundancy: By excluding IP addresses and specific attack types, the feature set minimizes redundancy and prevents the model from overfitting to particular network endpoints or known attack signatures. This approach enhances the model's capability to generalize and detect novel threats.

- Efficiency in Processing: The retained features are sufficient to represent the essential aspects of network traffic without imposing excessive computational overhead, which is crucial for real-time processing in IoT and IIoT environments.

7.3.1.2 Feature Scaling

To ensure that all features contribute equally to the learning process and to improve the performance of distance-based classifiers like kNN, feature scaling was employed. We used a standard scaler approach to standardize the feature set. This method standardizes the features by removing the mean and scaling them to unit variance. Specifically, each feature x is transformed using the formula:

$$x_{\text{scaled}} = \frac{x - \mu}{\sigma} \tag{7.1}$$

where μ is the mean and σ is the standard deviation of the feature.
The benefits of standardizing the feature sets include;

- Uniform Feature Contribution: By standardizing the features, the model ensures that no single feature dominates due to its scale, which is particularly important for algorithms sensitive to feature magnitudes.
- Improved Convergence: Standardization can lead to faster and more stable convergence during the training process, enhancing the overall efficiency of the model.
- Compatibility with Distance-Based Algorithms: For classifiers like kNN, standardized features ensure that distance calculations are meaningful and not skewed by disparate feature scales.

7.3.2 Adaptive SAMKNN Module

The core detection engine of the IDS relies on the proposed classifier, which dynamically balances between recent and historical data through two types of memory: STM and LTM.

7.3.2.1 Short-Term Memory (STM)

STM holds recent data samples, allowing the classifier to adapt quickly to emerging attack patterns. It is essential for recognizing and responding to novel attacks that

may not match historical attack signatures.

7.3.2.2 Long-Term Memory (LTM)

LTM contains aggregated historical data, providing a stable base of knowledge for the classifier. LTM enables the system to retain context from past experiences, enhancing its ability to identify well-known attack patterns.

7.3.2.3 Dynamic Memory Adjustment

The classifier dynamically adjusts the balance between STM and LTM based on model performance. If the accuracy improves, the LTM size increases to retain more historical data. Conversely, if performance drops, the LTM size is reduced, allowing the model to focus on recent data. The model's performance is tracked through a sliding window of recent predictions, and memory adjustments are made when sufficient data is available, regulated using a cool-down period. The cool-down technique ensures efficient memory usage and maintains accuracy in stationary and non-stationary environments, adapting to concept drift over time.

7.3.3 Attack Detection Module

The Attack Detection Module is responsible for identifying anomalous behavior that may indicate an attack. It processes incoming data using classifiers to distinguish between benign and malicious activity. The module leverages mechanisms like memory adjustment to handle zero-day attacks effectively. This module includes:

7.3.3.1 Attack Detector

The proposed classifier processes incoming data and classifies each instance as benign or malicious based on learned patterns. In the case of zero-day attacks, the classifier's memory adjustment mechanism enhances its capacity to detect previously unseen attack types without prior knowledge.

7.3.3.2 Zero-Day Analysis

This component emphasizes the identification of novel attack instances by leveraging the classifier's STM. By prioritizing recent data, the classifier can respond

to emerging threats that deviate from normal patterns or known attack signatures, thereby improving the detection of zero-day attacks.

7.3.4 Alert and Response Module

This module is essential for real-time threat handling by promptly notifying administrators of detected threats and automatically responding to them to mitigate damage. It ensures a proactive approach to minimize the impact of attacks.

7.3.4.1 Alert Manager

The Alert Manager triggers an alert whenever malicious activity is detected. This alert can be customized based on the threat level, providing immediate feedback to system administrators and initiating automated responses if necessary.

7.3.4.2 Response

The system can execute pre-configured response actions, such as isolating the compromised device, restricting access to specific network resources, or blocking traffic from suspicious sources. These response mechanisms are crucial for containing and mitigating the impact of detected attacks in IoT/IIoT environments.

7.3.5 Performance Monitoring Module

The performance monitoring module continuously evaluates the system's performance to ensure sustained accuracy and adaptability.

7.3.5.1 Performance Metrics

The system tracks various performance metrics, such as detection accuracy, false positives, and false negatives, to assess the efficacy of the classifier in real-time.

7.3.6 Adaptive SAMKNN

In this subsection, we explain how we implemented the proposed Adaptive SAMKNN, a modification of the original SAMKNN proposed by [36].

The proposed classifier is an enhanced KNN algorithm specifically designed for dynamic streaming environments where the nature of the data may evolve over time. This adaptation incorporates mechanisms for managing memory resources

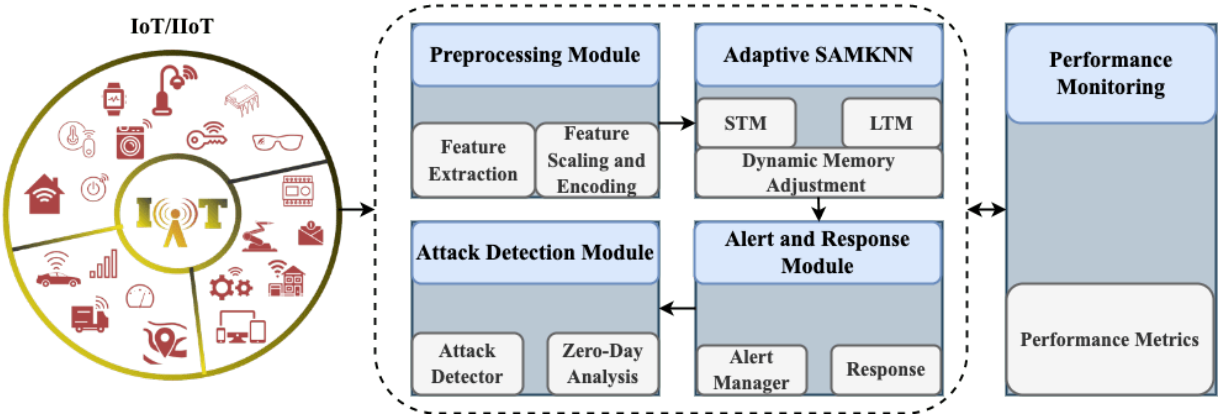


Figure 7.1: An architectural diagram of our proposed IDS

efficiently, dynamically adjusting between short-term and long-term memory based on observed performance, and adapting to drifts in data distribution.

The algorithm begins by initializing two memory structures, STM and LTM. STM is a memory buffer that holds recent data samples, while LTM stores older, stable data that could still be useful for classification. An initial allocation proportion λ_{LTM} is set for LTM to control the balance between the sizes of STM and LTM. Key parameters are also initialized, including performance thresholds ($\delta_{decrease}$ and $\delta_{increase}$) that determine when the memory allocation should be adjusted based on classifier performance. Additionally, a cool-down period is set to prevent frequent adjustments and allow time for performance changes to become evident. Finally, label encoding dictionaries are prepared, which may map categorical labels to numerical values.

For each incoming data sample x, the algorithm follows a structured process. If the label y of the sample is known (i.e., the sample is labeled), the classifier first predicts the label $\hat{y}$ for x using both STM and LTM. This prediction is then used to update the algorithm's accuracy metric, maintained over a performance window $\mathcal{P}$, tracking how well the classifier has been performing recently.

The labeled sample (x, y) is then added to STM. However, if the size of STM exceeds a predefined threshold M_{STM}, the oldest samples in STM may be shifted to LTM (if such shifting is enabled). This process ensures that STM remains a reservoir of recent data without overflow while allowing LTM to accumulate samples that may still hold relevant information for classification.

The adaptive mechanism of SAMKNN comes into play after every performance window $\mathcal{P}$ reaches a size W and the cool-down period has elapsed. At this point, the classifier evaluates its recent performance over $\mathcal{P}$ by computing an average performance metric $\overline{P}$. This metric is compared against previous performance

to detect potential concept drifts. If $\overline{P}$ has dropped by a threshold $\delta_{decrease}$, this indicates that the classifier may be underperforming, possibly due to outdated information in LTM. In response, the algorithm decreases λ_{LTM}, thereby reducing the size of LTM and giving STM a larger proportion of memory resources. This adjustment helps the classifier focus more on recent data (stored in STM), which is likely more relevant to the current data distribution. After decreasing λ_{LTM}, LTM may be compressed, meaning that the oldest samples in LTM are removed to fit within the new, reduced memory allocation. Conversely, if $\overline{P}$ has improved by a threshold $\delta_{increase}$, this suggests that retaining older data might be beneficial. In this case, the algorithm increases λ_{LTM}, enlarging LTM's memory allocation to preserve more long-term information. This adjustment helps the classifier leverage older patterns that have become relevant again. Once either adjustment is made, the cool-down counter τ is reset, ensuring that memory allocation changes are infrequent enough to allow stable adaptation to data changes without oscillating due to short-term performance fluctuations.

For each incoming unlabeled sample, the classifier predicts a label by choosing neighbors from either STM only or from both STM and LTM, depending on the specific configuration. If the classifier is configured to use only STM, it selects the nearest neighbors based on recent data alone, which might be suitable in highly volatile environments where only recent information is relevant. When using both STM and LTM, the classifier computes distances for x across both memory structures, selects the top k neighbors, and then uses either majority voting or distance-weighted voting to make a final prediction. This hybrid approach allows the classifier to balance recent trends with historical patterns when making decisions.

At regular intervals, the classifier updates the sizes of STM and LTM based on the current value of λ_{LTM}. The size of LTM is set to $\lambda_{LTM} \times M_{max}$, where M_{max} is the total available memory. STM's size M_{STM} is then calculated as the remaining memory ($M_{max} - M_{LTM}$).

When LTM's size exceeds its allocated memory M_{LTM}, the algorithm compresses LTM by removing the oldest samples until its size fits within M_{LTM}. This ensures that the classifier maintains memory constraints and prioritizes retaining more recent, potentially more relevant data within LTM. The pseudocode of the proposed classifier is shown in Algorithm 4.

7.3.7 Complexity Analysis of the Adaptive SAMKNN

Understanding the computational complexity of Adaptive SAMKNN is crucial for assessing its scalability and efficiency in practical applications.

7.3.7.1 Time Complexity

During training, the algorithm processes each incoming sample through several steps:

- Label Encoding: The algorithm assigns a unique numerical index to each new label encountered. This operation uses a dictionary lookup and insertion, done in constant time, O(1), per sample.
- The new sample is appended to the STM, implemented as a deque. Appending to a deque is an O(1) operation.
- If the STM exceeds its maximum size, the oldest samples are shifted to the LTM. This involves:
- Deque Operations: Removing samples from the STM, which takes O(k), where k is the number of samples to shift.
- Updating LTM: Adding the shifted samples to the LTM. This may involve concatenating NumPy arrays, which have a time complexity of O(n + k), where n is the current size of the LTM.
- If the LTM exceeds its maximum size, the oldest samples are removed to maintain the size constraint. This involves slicing NumPy arrays, which is an O(1) operation because slicing creates a view rather than copying data.
- The algorithm updates a performance window (a deque) with the latest prediction result. This operation is O(1).
- It calculates the mean performance over the window, which is O(w), where w is the size of the performance window. Since w is a fixed parameter, this operation is effectively O(1) per sample.
- Based on performance evaluation, the algorithm may adjust the sizes of STM and LTM. This decision-making process is O(1).

Algorithm 4 Proposed Classifier

Input: New sample $\mathbf{x}$, label y (if available), hyperparameters
Output: Predicted label $\hat{y}$ for $\mathbf{x}$
{— Initialization —}
Initialize Short-Term Memory (STM) and Long-Term Memory (LTM)
Set initial memory allocation proportion λ_{LTM} for LTM
Define performance thresholds δ_{decrease} and δ_{increase}
Set cooldown period τ
Initialize label encoding dictionaries
{— For each incoming sample $\mathbf{x}$: —}
if label y is available **then**
    Predict $\hat{y}$ for $\mathbf{x}$ using STM and LTM
    Update accuracy metric and performance window $\mathcal{P}$
end if
Update STM with new sample $(\mathbf{x}, y)$
if $|\text{STM}| > M_{\text{STM}}$ **then**
    Shift samples from STM to LTM (if enabled)
end if
if $|\mathcal{P}| = W$ and cooldown period has elapsed **then**
    Compute current performance $\overline{P}$ over $\mathcal{P}$
    if $\overline{P}$ decreased by δ_{decrease} **then**
        Decrease $\lambda_{LTM} \leftarrow \max(\lambda_{LTM} - \Delta\lambda, \lambda_{\min})$
        Call UpdateMemorySizes() and CompressLTM() if necessary
    else if $\overline{P}$ increased by δ_{increase} **then**
        Increase $\lambda_{LTM} \leftarrow \min(\lambda_{LTM} + \Delta\lambda, \lambda_{\max})$
        Call UpdateMemorySizes()
    end if
    Reset cooldown counter τ
end if
{— Prediction Function —}
if using STM only **then**
    Return $\hat{y}$ based on k nearest neighbors in STM
else
    Compute distances $d(\mathbf{x}, \mathbf{x}_i)$ for $\mathbf{x}_i \in \text{STM} \cup \text{LTM}$
    Select top k neighbors based on sorted distances
    Return $\hat{y}$ based on majority or distance-weighted voting
end if
{— UpdateMemorySizes() —}
$M_{\text{LTM}} \leftarrow \lfloor \lambda_{LTM} \times M_{\max} \rfloor$
$M_{\text{STM}} \leftarrow M_{\max} - M_{\text{LTM}}$
{— CompressLTM() —}
Remove oldest samples from LTM to ensure $|\text{LTM}| \leq M_{\text{LTM}}$

The per-sample training time is dominated by operations that are either constant time or depend on fixed-size parameters. Therefore, the overall time complexity for training per sample is effectively O(1), assuming that the sizes of STM, LTM, and the performance window are bounded and relatively small.

Prediction involves several steps that are more computationally intensive:

- Calculating distances between the query sample and all samples in STM. This operation is O(s · d), where s is the number of samples in STM and d is the number of features.
- Similarly, calculating distances to all samples in LTM. This is O(l · d), where l is the number of samples in LTM.
- The total time for computing distances to both STM and LTM samples is O((s + l) · d).
- Instead of sorting all distances, the algorithm needs to find the top k nearest neighbors.
- Using a partial sort or a min-heap, this can be achieved in O((s + l) · log k).
- Aggregating the labels of the nearest neighbors to make a prediction. This operation is O(k).

The total time complexity for prediction per sample is:

$$O\left((s+l)\cdot d + (s+l)\cdot \log k + k\right) \tag{7.2}$$

Since k is typically small and $\log k$ is negligible compared to s and l, the dominant term becomes:

$$O\left((s+l)\cdot d\right) \tag{7.3}$$

This indicates that the prediction time scales linearly with the total number of samples in STM and LTM and the number of features.

7.3.7.2 Space Complexity

The space required by the algorithm is determined by the storage of samples, labels, and auxiliary data structures:

- STM Storage: O(s · d), where s is the maximum size of STM. The space complexity of the STM storing a label is O(s), storing one label per sample.
- LTM Storage: O(l · d), where l is the maximum size of LTM. The space complexity of the LTM storing a label is O(l).
- Performance Window: O(w), where w is the size of the performance window.
- Label-to-Index Mapping: O(c), where c is the number of unique classes.

The total space complexity is:

$$O\left((s+l)\cdot d + s + l + w + c\right) \tag{7.4}$$

Given that s and l are bounded by user-defined parameters ('maxSTMSize' and 'maxLTMSize'), and w and c are relatively small, the space complexity is effectively linear in the maximum allowed memory sizes and the number of features:

$$O\left(\text{maxWindowSize} \cdot d\right) \tag{7.5}$$

7.4 Experimental Design

7.4.1 Experimental Setup

The experimental validation of the model developed in this work is presented in this subsection. Python 3.10 was utilized throughout the experimental validation to develop the proposed system using the River online ML framework [37]. We chose the River online ML framework because it can process streaming data. We used a MacBook Pro with an M1 chip running on 16 GB of RAM and 1TB of disk storage to model the proposed system.

7.4.2 Datasets

We used three datasets to build and test our proposed IDS. The datasets are NF-BoT-IoT, NF-ToN-IoT, and NF-CSE-CIC-IDS 2018 proposed by [38]. The NetFlow records of these datasets were generated from the publicly available pcap files of the respective datasets. Table 7.3 shows a summary of the various datasets as proposed by [38]. We chose two attack classes from each dataset.

Table 7.3: Summary of the datasets as proposed by [38]

Dataset	#Features	Benign	#Samples
NF-BoT-IoT	12	13,859 (2.31%)	600,100
NF-ToN-IoT	12	179647 (13.94%)	1,288,642
NF-CSE-CIC-IDS 2018	12	7,373,198 (87.86%)	8,392,401

7.4.3 Evaluation Metrics

To comprehensively assess the effectiveness and efficiency of the proposed Adaptive SAMKNN system for zero-day attack detection in IoT and IIoT environments, a diverse set of evaluation metrics was employed. These metrics provide insights into various aspects of the system's performance, including accuracy, reliability, computational efficiency, and statistical significance. The following sections define each metric and elucidate its relevance to evaluating the proposed intrusion detection system.

7.4.3.1 Accuracy

Accuracy measures the proportion of correctly classified instances (both true positives and true negatives) out of the total number of instances evaluated. It is calculated as:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Instances}} \tag{7.6}$$

7.4.3.2 F1 Score

The F1 score is the harmonic mean of precision and recall, defined as:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7.7}$$

where Precision is the ratio of true positives to the sum of true positives and false positives, and Recall (or Detection Rate) is the ratio of true positives to the sum of true positives and false negatives.

7.4.3.3 False Positive Rate (FPR)

FPR quantifies the proportion of benign instances incorrectly classified as malicious. It is calculated as:

$$\text{FPR} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}} \tag{7.8}$$

7.4.3.4 Detection Rate (DR)

Detection Rate, also known as Recall or Sensitivity, measures the proportion of actual malicious instances correctly identified by the system. It is expressed as:

$$\text{Detection Rate} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{7.9}$$

DR is fundamental in evaluating the system's effectiveness in identifying real threats. A high detection rate indicates the system's proficiency in recognizing and mitigating malicious activities, which is paramount for maintaining the security integrity of IoT and IIoT networks.

7.4.3.5 Processing Time

Processing Time refers to the duration required by the system to analyze and classify each data instance. It is typically measured in milliseconds or seconds per instance.

In real-time intrusion detection, rapid processing is essential to ensure timely responses to threats. Monitoring processing time assesses the system's capability to operate efficiently under high-throughput conditions typical of IoT environments.

7.4.3.6 Paired t-test

The paired t-test is a statistical method used to compare the means of two related groups to determine if there is a significant difference between them.

Formula for Paired t-Test:

$$t = \frac{\bar{d}}{s_d/\sqrt{n}} \tag{7.10}$$

where:

$\bar{d}$ is mean of the differences, s_d is the standard deviation of the differences, and n is the number of paired observations.

Applying the paired t-test allows for the evaluation of the statistical significance of performance improvements achieved by the proposed classifier over baseline classifiers. It ensures that observed differences in metrics like accuracy or F1 score are not due to random chance.

7.4.3.7 Brier Score

The Brier Score measures the accuracy of probabilistic predictions by calculating the mean squared difference between predicted probabilities and the actual binary outcomes. It ranges from 0 to 1, with lower scores indicating better calibration.

$$\text{Brier Score} = \frac{1}{N}\sum_{i=1}^{N}(f_i - o_i)^2 \tag{7.11}$$

where f_i is the predicted probability and o_i is the actual outcome.

The Brier Score evaluates the reliability of the probability estimates produced by the classifier, ensuring that the system not only makes accurate predictions but also provides well-calibrated confidence levels in its detections.

7.4.3.8 Bootstrap Confidence Interval

Bootstrap Confidence Intervals are statistical intervals estimated by repeatedly resampling the dataset with replacement and computing the metric of interest across these samples to determine the range within which the true metric value lies with a certain confidence level.

Bootstrap Confidence Intervals provide insights into the variability and reliability of the performance metrics, enabling a more robust understanding of the system's performance across different data samples.

7.4.3.9 CPU Usage

CPU Usage measures the percentage of central processing unit resources consumed by the system during operation.

Monitoring CPU usage is essential for evaluating the computational efficiency of the proposed classifier, especially in resource-constrained IoT devices where excessive CPU consumption can lead to performance bottlenecks and increased energy consumption.

7.4.3.10 Memory Usage

Memory Usage quantifies the amount of RAM utilized by the system during its operation.

Efficient memory usage is critical for deploying intrusion detection systems on IoT devices with limited memory resources. Assessing memory consumption

ensures that Adaptive SAMKNN can operate effectively without exceeding the device's memory constraints, thereby facilitating broader deployment in diverse IoT environments.

The selected metrics collectively provide a holistic evaluation of the proposed classifier's performance in detecting zero-day attacks within IoT and IIoT systems. Accuracy and F1 score offer general performance insights, while FPR and DR specifically address the balance between detecting true threats and minimizing false alarms, which is crucial for maintaining system reliability. Processing Time, CPU Usage, and Memory Usage assess the system's operational efficiency, ensuring its practicality for real-time applications in resource-constrained environments. Statistical measures like the Paired t-test, t-statistic, Brier Score, and Bootstrap Confidence Interval validate the robustness and reliability of the performance improvements, ensuring that the system's enhancements are both significant and consistent. Collectively, these metrics ensure that the proposed classifier not only excels in identifying novel and emerging threats but also operates efficiently within the stringent resource limitations typical of IoT and IIoT deployments.

7.4.4 Experimental Validations

Eight experiments were conducted to test and validate our proposed system. The experiments are briefly described below.

7.4.4.1 Baseline Performance Evaluation

The first experiment evaluated the performance of the proposed classifier on the three datasets while recording the accuracy, F1 score, processing time, and memory usage. As part of the baseline performance evaluation, we looked at the performance of five popular offline ML algorithms, namely, Random Forest (RF), KNN, Support Vector Machines (SVM), Decision Tree (DT), Logistic Regression (LR), and compared their performance to our proposed technique. We also compared the performance of our proposed method with the original SAMKNN algorithm.

7.4.4.2 Zero-day Attack Detection

This experiment aims to evaluate the effectiveness of the proposed classifier in detecting both known and zero-day attacks within IoT network traffic data. Zero-day attacks are novel threats that have not been previously encountered or included

in the training data, posing significant challenges for intrusion detection systems. By simulating zero-day attacks and assessing the classifier's performance, we seek to understand its capability in real-world scenarios where new types of attacks emerge. Two different experiments were carried out in this experiment. During the first experiment, we used some attack classes from the datasets used to validate our system as zero-day attacks. In the second experiment, we used the Conditional Tabular Generative Adversarial Network (CTGAN) proposed by [39] to generate synthetic zero-day attacks into the original datasets. CTGAN is a specialized type of GAN designed to generate realistic synthetic tabular data by conditioning on specific data features. It consists of two main components: a generator that creates fake data samples and a discriminator that evaluates whether samples are real or synthetic, both trained simultaneously to improve the quality of generated data. Additionally, it employs a training strategy that focuses on capturing the relationships and dependencies between various features in the dataset, enhancing the fidelity of the synthetic data. The zero-day attacks were simulated by excluding some attack instances from the training dataset, ensuring the classifier had not been exposed to this attack type during training. These unseen instances were then introduced into the testing dataset as zero-day attacks, allowing the evaluation of the model's ability to detect novel and previously unseen threats. In the NF-BoT IoT dataset, Reconnaissance, DDoS, and Theft were used to simulate zero-day attacks. Injection, Malware, and DDoS from the NF-ToN IoT and NF-CSE datasets were used to simulate zero-day attacks. A crucial metric for practical applications is indicating how often benign traffic is incorrectly classified as malicious. The average time taken to predict each instance, indicating the model's efficiency, is also measured.

7.4.4.3 Scalability

In this experiment, we evaluate the scalability and performance of the Adaptive SAMKNN on a large-scale IoT network traffic dataset. The primary focus is to assess how the classifier performs in terms of predictive accuracy and resource utilization (CPU and memory usage) when processing data in a streaming, online learning context. By monitoring these aspects, we aim to determine the model's suitability for real-time intrusion detection in IoT environments, where accuracy and efficiency are critical. We measured these metrics over samples of 50000,

200000, 350000, and 500000.

7.4.4.4 False Positive Rate Evaluation Under Normal Conditions

This experiment assesses the performance of the proposed technique in terms of its false positive rate when exposed solely to benign network traffic. Understanding the FPR under normal operating conditions is crucial for IDSs, as a high rate of false alarms can lead to alert fatigue, resource wastage, and decreased trust in the system. A low false positive rate ensures that security personnel can focus on genuine threats without being overwhelmed by false alarms.

7.4.4.5 Performance Under Drift

In the rapidly evolving landscape of network security, particularly within the IoT and IIoT ecosystem, detecting malicious activities is a complex and dynamic challenge. IDSs must not only identify known threats but also adapt to changes in network behavior over time. One significant challenge in this domain is concept drift, where the statistical properties of the target variable change over time in unforeseen ways. This experiment evaluates the performance of our proposed Adaptive SAMKNN under various types of drift scenarios simulated in IoT network traffic data. This experiment considered four types of drifts: gradual, sudden, recurring, and incremental. Our simulated concept drift was executed by introducing variations in the class label of the dataset.

7.4.4.6 Resource Utilization

In this experiment, we aim to compare the resource utilization and processing efficiency of the proposed classifier with the standard SAMKNN classifier when applied to IoT network traffic data. By evaluating metrics such as CPU usage, memory consumption, and processing time, we seek to understand the computational overhead introduced by the adaptive capabilities of the proposed classifier. This comparison is crucial for determining the practicality of deploying adaptive models in resource-constrained environments typical of IoT networks. To get a good approximation of the amount of resources the proposed model consumes, we ran ten experiments while recording the resource usage and then calculated the mean of those experiments.

7.4.4.7 Statistical Analysis

To assess the proposed model's robustness and ensure the results are replicable and not randomized, we performed statistical analysis using bootstrap confidence interval, Brier score, paired t-test, and t-statistic.

The bootstrap confidence interval assesses the performance of the proposed classifier in detecting zero-day attacks within IoT networks by applying 100 bootstrap samples to each dataset and evaluating the accuracy of each iteration. The resulting accuracies are used to calculate the mean and 95% confidence interval, which are then visualized with a histogram to demonstrate the classifier's reliability and consistency.

Another statistical analysis involves assessing the calibration accuracy of the proposed classifier on the respective dataset by implementing a custom Brier Score metric to evaluate the precision of predicted probabilities for the positive class. The algorithm iterates over each instance to obtain probability predictions and updates the Brier Score accordingly. The study quantifies the reliability of the classifier's probabilistic outputs in effectively detecting zero-day attacks in IoT environments.

On each dataset, we compare the performance of the proposed classifier and the original SAMKNN by conducting five independent runs with randomly shuffled data samples, measuring each classifier's accuracy in detecting zero-day attacks. The resulting accuracy scores are statistically analyzed using a paired t-test to determine the significance of the performance differences between the two classifiers.

4.4.8 Ablation Study

In machine learning, an ablation study is a crucial experimental approach used to assess the contribution of individual components or features within a model. By systematically removing or altering these components, researchers can determine their impact on the overall performance of the system. This experiment involves conducting an ablation study on the proposed classifier within the context of IoT network intrusion detection. The goal is to understand how different aspects of the classifier influence its ability to detect malicious activities in network traffic data. The components removed are the adaptiveness, LTM, and SAM.

7.5 Results

7.5.1 Baseline Performance Evaluation

The experimental results presented in Tables 7.4, 7.5, 7.6, and 7.7 demonstrate the performance of the SAMKNN, Adaptive SAMKNN, and five offline ML algorithms across three benchmark datasets: NF-BoT-IoT, NF-ToN-IoT, and NF-CSE-CIC-IDS 2018. The Adaptive SAMKNN consistently outperforms the SAMKNN in memory efficiency across all datasets, achieving substantial reductions in memory usage while maintaining comparable accuracy and F1 scores. Specifically, Adaptive SAMKNN achieves memory savings of over 99% compared to SAMKNN without compromising detection accuracy, demonstrating its suitability for resource-constrained environments. Additionally, compared with traditional machine learning algorithms such as RF, KNN, SVM, DT, and LR, the Adaptive SAMKNN shows competitive accuracy and low processing time, making it a viable option for real-time attack detection in IoT environments. These results highlight the effectiveness of Adaptive SAMKNN in balancing accuracy, memory efficiency, and processing speed.

Table 7.4: Comparing the accuracy, F1 score, and model memory footprint (KB) of our proposed classifier against the original SAMKNN algorithm

Algorithm	**Accuracy**	**F1**	**Memory (KB)**
NF-BoT-IoT			
SAMKNN	98.52%	98.50%	5539.84
Adaptive SAMKNN	98.90%	98.89%	62.51
NF-ToN-IoT			
SAMKNN	95.76%	95.61%	59002.88
Adaptive SAMKNN	96.34%	96.23%	63.31
NF-CSE-CIC-IDS 2018			
SAMKNN	99.91%	99.91%	57477.12
Adaptive SAMKNN	99.92%	99.92%	55.11

7.5.2 Zero-day Attack Detection

Tables 7.8 and 7.9 present the performance of the proposed classifier in detecting zero-day attacks across three datasets (NF-BoT-IoT, NF-ToN-IoT, and NF-CSE-

Table 7.5: Comparing the accuracy, processing time (s), and model memory footprint (MB) of our proposed classifier against five other popular ML algorithms when evaluated on NF BoT IoT dataset

Algorithm	**Accuracy**	**Time (s)**	**Memory (MB)**
RF	98.93%	14.98	1436.41
KNN	99.13%	26.47	1439.48
SVM	98.81%	2550.37	2043.38
DT	98.96%	1.62	1366.13
LR	98.69%	2.70	982.86
Our method	98.90%	11	0.06

Table 7.6: Comparing the accuracy, processing time (s), and model memory footprint (MB) of our proposed classifier against five other popular ML algorithms when evaluated on NF ToN IoT dataset

Algorithm	**Accuracy**	**Time (s)**	**Memory (MB)**
RF	99.93%	79.51	1152.52
KNN	99.34%	102.49	896.75
SVM	97.94%	10364.42	2547.66
DT	999.91%	3.57	2101.47
LR	995.98%	2.54	2053.06
Our method	96.34%	187	0.06

Table 7.7: Comparing the accuracy, processing time (s), and model memory footprint (MB) of our proposed classifier against five other popular ML algorithms when evaluated on the NF CSE 2018 dataset

Algorithm	**Accuracy**	**Time (s)**	**Memory (MB)**
RF	99.99%	23.71	1515.58
KNN	99.99%	106	1544.66
SVM	99.98%	2428.68	1792.39
DT	99.99%	2.40	1743.98
LR	99.96%	1.82	1754.58
Our method	99.92%	141	0.05

CIC-IDS 2018). The Adaptive SAMKNN demonstrates robust accuracy, achieving over 99% across all attack classes, with a consistent zero-day detection rate (DR) of 1.00, indicating that it successfully identified all simulated zero-day attacks. The classifier maintains a low FPR, particularly for NF-ToN-IoT and NF-CSE-CIC-IDS 2018, while the NF-BoT-IoT dataset shows slightly higher FPR values, especially for Theft and DDoS attacks. Additionally, the average detection latency remains minimal at 0.0001 seconds across all datasets, supporting its suitability for real-time applications. These results illustrate that the Adaptive SAMKNN can effectively identify and respond to novel threats with high precision and minimal delay, making it a promising solution for dynamic IoT and IIoT security challenges.

Table 7.8: The proposed model's accuracy, the detection rate of a zero-day attack, the false positive rate of the zero-day attack, and the average detection latency when some unseen attack classes from the three datasets are used as zero-day attacks

Zero-day	Accuracy	DR	FPR	Detection Latency
NF-BoT-IoT				
Theft	99.62%	1.00	0.1496	0.0001
DDoS	99.70%	1.00	0.1356	0.0001
Recon	99.46%	1.00	0.0385	0.0001
NF-ToN-IoT				
Injection	99.84%	1.00	0.0037	0.0001
Malware	99.76%	1.00	0.0108	0.0001
DDoS	99.76%	1.00	0.0089	0.0001
NF-CSE-CIC-IDS 2018				
DDoS	99.92%	1.00	0.0037	0.0001
Malware	99.99 %	1.00	0.0030	0.0001
Injection	99.99%	1.00	0.0032	0.0001

Table 7.9: The proposed model's accuracy, detection rate of zero-day attack, false positive rate of the zero-day attack, and the average detection latency when unseen attack classes that contain synthetic zero-day attacks created using CTGAN from the three datasets are used as zero-day attacks

Zero-day	Accuracy	DR	FPR	Detection Latency
NF-BoT-IoT				
Theft	99.48%	1.00	0.1678	0.0001
DDoS	99.50%	1.00	0.1487	0.0006
Recon	99.16%	1.00	0.0671	0.0001
NF-ToN-IoT				
Injection	99.90%	1.00	0.0040	0.0001
Malware	99.82%	1.00	0.0119	0.0001
DDoS	99.88%	1.00	0.0097	0.0001
NF-CSE-CIC-IDS 2018				
DDoS	99.94%	1.00	0.0035	0.0001
Malware	99.96 %	1.00	0.0030	0.0001
Injection	99.96%	1.00	0.0028	0.0001

7.5.3 Scalability

The scalability assessment of Adaptive SAMKNN, as shown in Figures 7.2, 7.3, and 7.4 and Table 6.10, demonstrates its capability to maintain consistent performance across increasing data volumes on three datasets: NF-BoT-IoT, NF-ToN-IoT, and NF-CSE-CIC-IDS 2018. With sample sizes increasing from 50,000 to 500,000, Adaptive SAMKNN achieves stable accuracy, retaining over 98% accuracy on NF-BoT-IoT, 96% on NF-ToN-IoT, and 99% on NF-CSE-CIC-IDS 2018. CPU usage remains minimal (around 12.4%) across all datasets, and memory usage is kept consistently low, especially on NF-BoT-IoT with 45.84 MB and slightly higher on

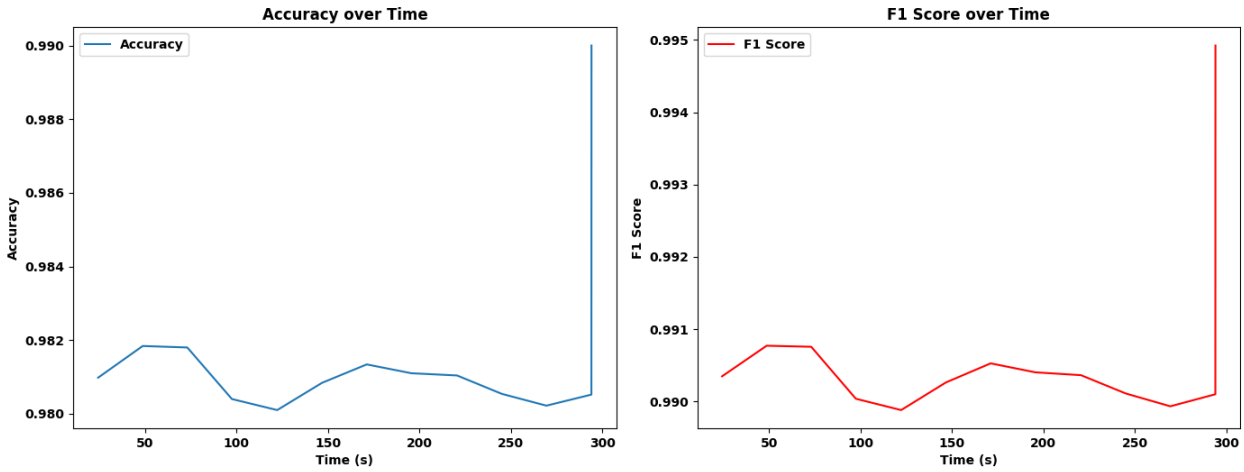
Table 7.10: The model's processing time, CPU usage (%), device memory usage (MB), and accuracy when the number of samples is increased.

# Samples	Time (s)	CPU (%)	Memory (MB)	Accuracy
NF-BoT-IoT				
50000	24.25	12.40	45.84	98.10%
200000	97.41	12.44	45.85	98.04%
350000	171.12	12.45	45.85	98.13%
500000	245.00	12.44	45.85	98.05
NF-ToN-IoT				
50000	24.82	12.40	80.06	96.25%
200000	98.41	12.47	80.06	96.22%
350000	172.30	12.48	80.06	96.38%
500000	245.99	12.49	80.06	96.42
NF-CSE-CIC-IDS 2018				
50000	21.46	12.44	80.05	99.91%
200000	85.99	12.45	80.06	99.92%
350000	151.11	12.48	80.06	99.93%
500000	216.30	12.38	80.06	99.96

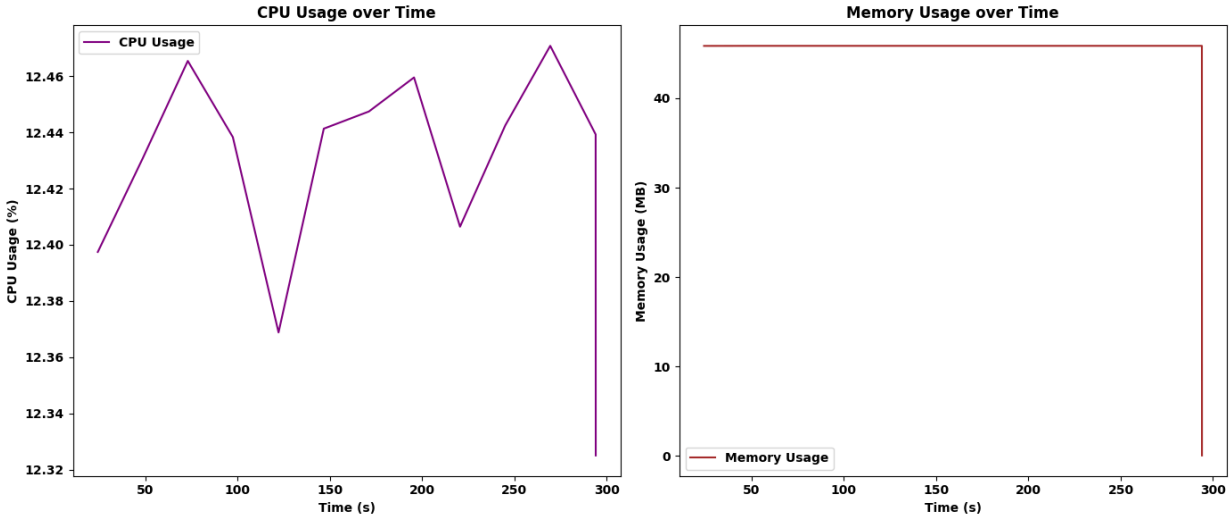
the other datasets. Processing times increase proportionally with sample size but remain manageable for large datasets. The scalability test confirms that Adaptive SAMKNN maintains high accuracy, low CPU usage, and efficient memory management, underscoring its effectiveness for real-time applications even as data volume scales.

7.5.4 False Positive Rate Evaluation Under Normal Conditions

Table 7.11 shows the False Positive Rate (FPR) of the proposed classifier under normal conditions across three datasets: NF-BoT-IoT, NF-ToN-IoT, and NF-CSE-CIC-IDS 2018. The results indicate that Adaptive SAMKNN achieved an FPR of 0.00% on all datasets, demonstrating its ability to identify normal traffic without false alarms correctly. With 13,858, 179,646, and 62,466 normal samples in the NF-BoT-IoT, NF-ToN-IoT, and NF-CSE-CIC-IDS 2018 datasets, respectively, the classifier consistently maintained perfect FPR performance. These findings underscore Adaptive SAMKNN's precision in distinguishing benign and malicious activities under stable, non-attack conditions. The absence of false positives is a significant outcome, as it highlights the model's reliability in recognizing genuine network behavior, minimizing the risk of alert fatigue in real-world monitoring environments where excessive false positives can hinder security operations.



(a) The accuracy and F1 score with respect to the growth of the processing time of the proposed method as the number of samples is increased (scalability) when the NF BoT IoT dataset is used to evaluate the model



(b) The CPU and memory usage with respect to the growth of the processing time of the proposed method as the number of samples is increased (scalability) when the NF BoT IoT dataset is used to evaluate the model

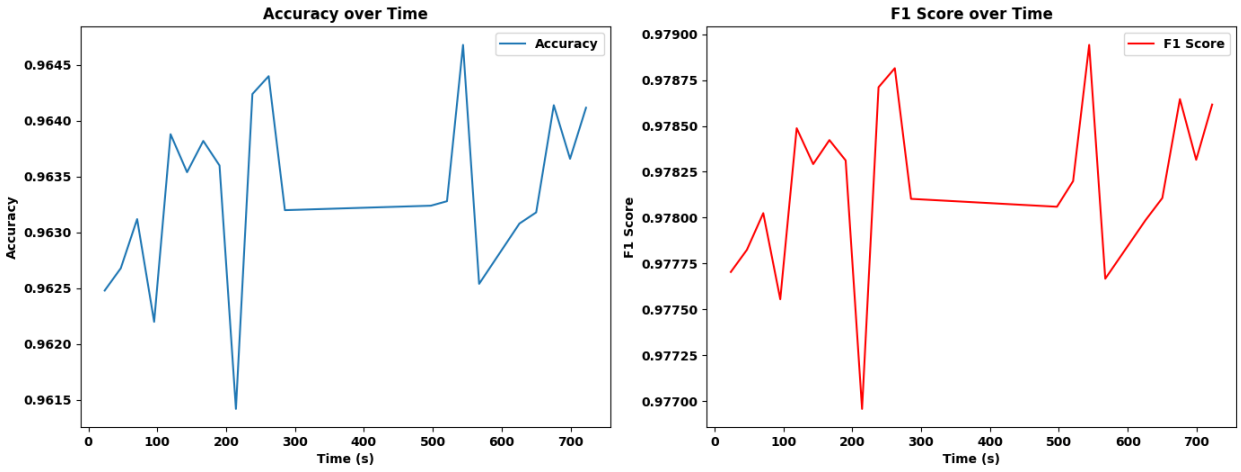
Figure 7.2: The accuracy, F1 score, CPU, and memory usage with respect to the growth of the processing time of the proposed method as the number of samples is increased (scalability) when the NF BoT IoT dataset is used to evaluate the model

Table 7.11: False Positive Rate Evaluation Under Normal Conditions

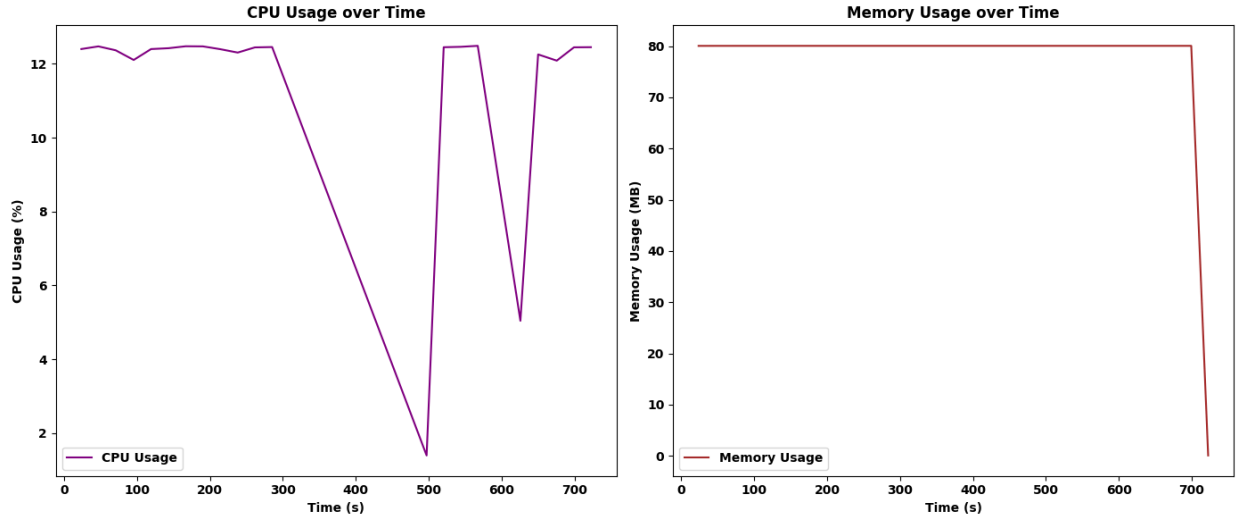
# Normal Samples	FPR
NF-BoT-IoT	
13858	0.00
NF-ToN-IoT	
179646	0.00
NF-CSE-CIC-IDS 2018	
62466	0.00

7.5.5 Performance Under Drift

Tables 7.12, 7.13, and 7.14 present the performance of Adaptive SAMKNN in handling different types of concept drift across three datasets: NF-BoT-IoT, NF-ToN-IoT, and NF-CSE-CIC-IDS 2018. The classifier exhibits varied accuracy



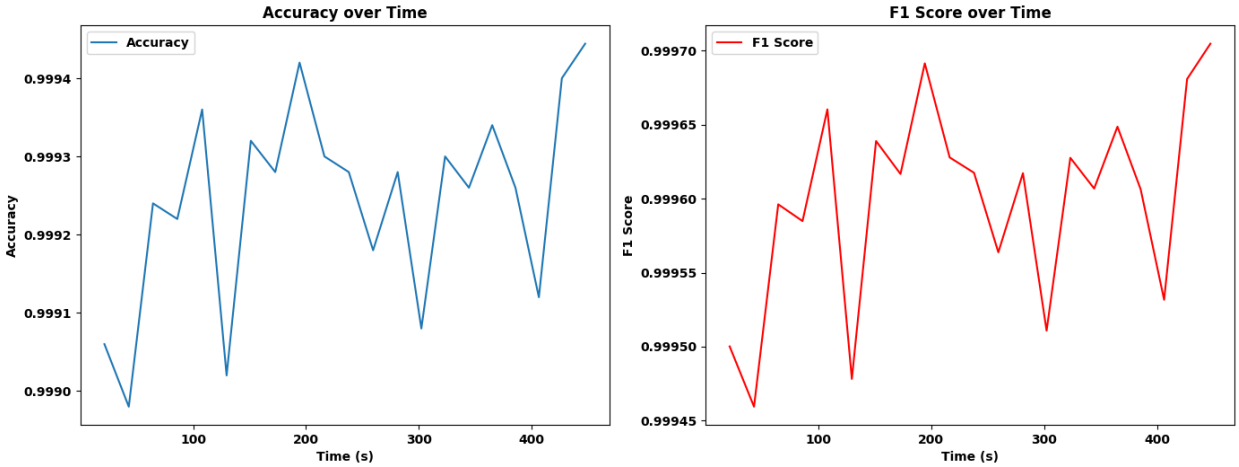
(a) The accuracy and F1 score with respect to the growth of the processing time of the proposed method as the number of samples is increased (scalability) when the NF ToN IoT dataset is used to evaluate the model



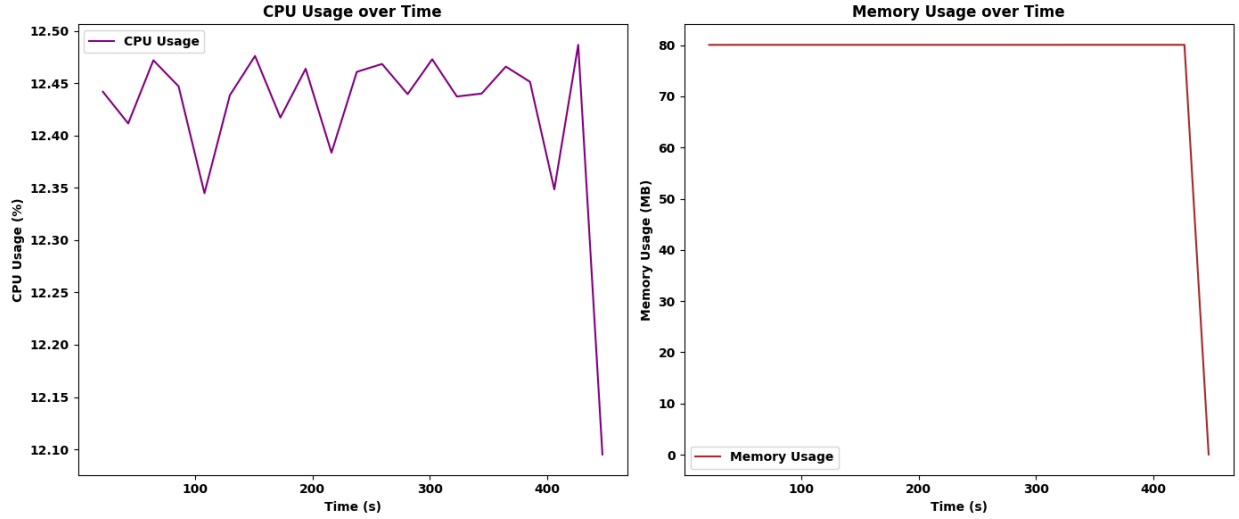
(b) The CPU and memory usage with respect to the growth of the processing time of the proposed method as the number of samples is increased (scalability) when the NF ToN IoT dataset is used to evaluate the model

Figure 7.3: The accuracy, F1 score, CPU, and memory usage with respect to the growth of the processing time of the proposed method as the number of samples is increased (scalability) when the NF ToN IoT dataset is used to evaluate the model

and F1 scores based on the nature of the drift. Adaptive SAMKNN achieves the highest performance for gradual drift, with accuracy and F1 scores exceeding 90% across all datasets. Sudden drift poses a greater challenge, resulting in the lowest performance, particularly on the NF-ToN-IoT dataset, where accuracy and F1 scores drop to around 81%. Performance under recurring and incremental drift remains stable, with accuracy and F1 scores generally above 87%. These results indicate that while Adaptive SAMKNN is capable of adapting to gradual changes in data distribution, its sensitivity to sudden changes suggests potential areas for improvement in environments with abrupt shifts. Overall, the classifier demonstrates robust adaptability to diverse drift types, which is essential for real-world IoT applications with dynamic data patterns.



(a) The accuracy and F1 score with respect to the growth of the processing time of the proposed method as the number of samples is increased (scalability) when the NF CSE dataset is used to evaluate the model



(b) The CPU and memory usage with respect to the growth of the processing time of the proposed method as the number of samples is increased (scalability) when the NF CSE dataset is used to evaluate the model

Figure 7.4: The accuracy, F1 score, CPU, and memory usage with respect to the growth of the processing time of the proposed method as the number of samples is increased (scalability) when the NF CSE dataset is used to evaluate the model

Table 7.12: The accuracy and F1 score of the proposed classifier when evaluated with the NF BoT IoT dataset containing different kinds of Drift

Type of Drift	**Accuracy**	**F1**
Gradual	90.95%	90.81%
Sudden	82.52%	82.38%
Recurring	87.66%	87.46%
Incremental	89.33%	88.97%

Table 7.13: The accuracy and F1 score of the proposed classifier when evaluated with the NF ToN IoT dataset containing different kinds of Drift

Type of Drift	**Accuracy**	**F1**
Gradual	89.55%	89.43%
Sudden	81.34%	81.20%
Recurring	86.31%	86.15%
Incremental	87.90%	87.69%

Table 7.14: The accuracy and F1 score of the proposed classifier when evaluated with the NF CSE 2018 dataset containing different kinds of Drift

Type of Drift	Accuracy	F1
Gradual	92.56%	92.56%
Sudden	83.77%	83.76%
Recurring	89.12%	89.11%
Incremental	90.81%	90.74%

7.5.6 Resource Utilization

Table 7.15 presents a comparative analysis of resource utilization between SAMKNN and Adaptive SAMKNN across the NF-BoT-IoT, NF-ToN-IoT, and NF-CSE-CIC-IDS 2018 datasets. The Adaptive SAMKNN consistently demonstrates lower mean CPU usage, memory consumption, and processing time compared to the standard SAMKNN. For instance, in the NF-BoT-IoT dataset, Adaptive SAMKNN achieves a mean CPU usage of 86.51% and memory consumption of 3.20 KB, whereas SAMKNN requires 94.51% CPU usage and 76.80 KB memory. Across all datasets, Adaptive SAMKNN maintains a processing time of 0.07 seconds, slightly faster than SAMKNN's 0.10 seconds. These findings emphasize Adaptive SAMKNN's computational and memory resource efficiency, making it more suitable for real-time applications in resource-constrained environments like IoT networks. The reduced resource demands of Adaptive SAMKNN, without sacrificing detection accuracy, underscore its advantage for deployment in practical, high-frequency data processing scenarios.

Table 7.15: Comparing the mean CPU usage in percentage(%), the mean memory usage in kilobytes(KB), and the mean processing time in seconds(s).

Algorithm	CPU Usage (%)	Memory (KB)	Time (s)
NF-BoT-IoT			
SAMKNN	94.51	76.80	0.10
Adaptive SAMKNN	86.51	3.20	0.07
NF-ToN-IoT			
SAMKNN	90.27	548.20	0.10
Adaptive SAMKNN	83.61	51.20	0.07
NF-CSE-CIC-IDS 2018			
SAMKNN	87.25	619.20	0.10
Adaptive SAMKNN	81.12	70.40	0.07

7.5.7 Statistical Analysis

The bootstrap confidence intervals for the proposed classifier demonstrate high accuracy across the three datasets, as shown in Figures 7.5, 7.6, and 7.7, with

narrow ranges indicating consistent performance. For the NF-BoT-IoT dataset, the classifier achieves a mean accuracy of 0.9813 with a 95% confidence interval between 0.9800 and 0.9826. Similarly, on the NF-ToN-IoT dataset, the mean accuracy is 0.9635 with a 95% confidence interval of 0.9612 to 0.9658. The NF-CSE-2018 dataset shows the highest accuracy, with a mean of 0.9968 and a 95% confidence interval from 0.9962 to 0.9972.

Additionally, the Brier scores indicate strong calibration, as shown in Table 7.16, with values of 0.0166, 0.0260, and 0.0028 for the NF-BoT-IoT, NF-ToN-IoT, and NF-CSE-2018 datasets, respectively.

The t-tests, as shown in Table 7.17, revealed significant differences between the proposed classifier and SAMKNN, with p-values below 0.001 across all datasets.

Table 7.16: The Brier score of the proposed classifier when evaluated with each of the datasets.

Dataset	**Brier Score**
NF-BoT-IoT	0.0166
NF-ToN-IoT	0.0260
NF-CSE-2018	0.0028

Table 7.17: The t-statistic and p-value when the accuracy of the proposed classifier is compared with SAMKNN

Dataset	**t-statistic**	**p-value**
NF-BoT-IoT	20.9729	0.0000
NF-ToN-IoT	24.1197	0.0000
NF-CSE-2018	8.5327	0.0010

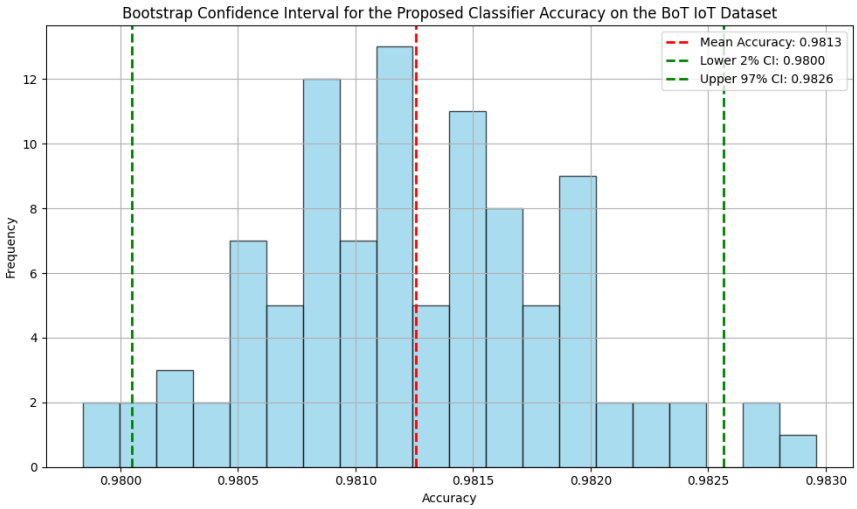


Figure 7.5: The bootstrap confidence interval for the proposed classifier accuracy on the NF BoT IoT dataset

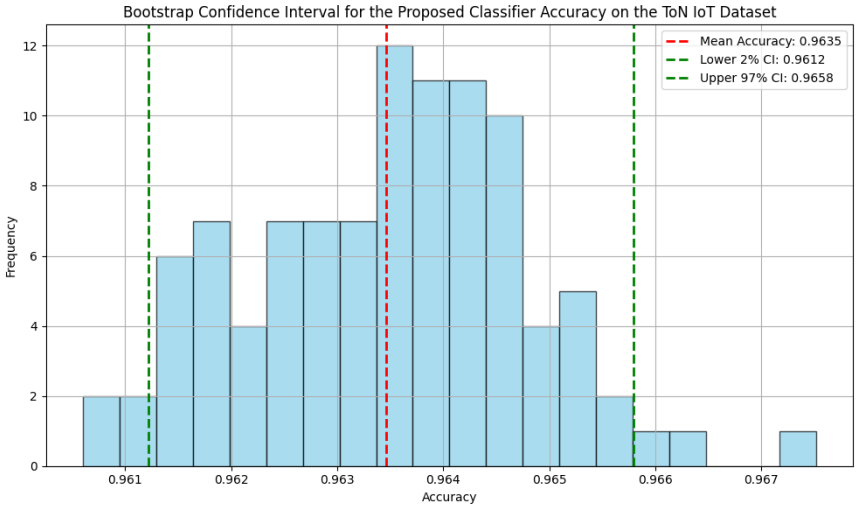


Figure 7.6: The bootstrap confidence interval for the proposed classifier accuracy on the NF ToN IoT dataset

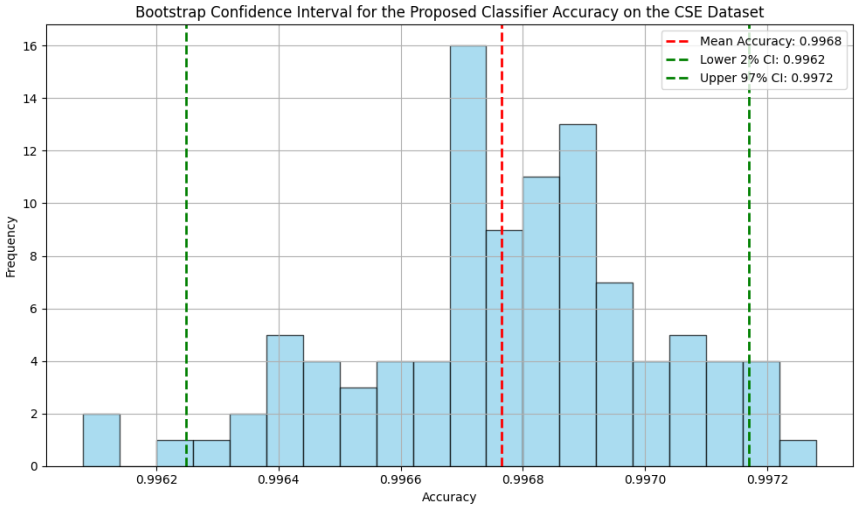


Figure 7.7: The bootstrap confidence interval for the proposed classifier accuracy on the NF CSE 2018 dataset

7.5.8 Ablation Study

The ablation study results in Table 7.18 compare the performance of Adaptive SAMKNN and SAMKNN with and without Long-Term Memory (LTM) across three datasets: NF-BoT-IoT, NF-ToN-IoT, and NF-CSE-CIC-IDS 2018. Adaptive SAMKNN without LTM consistently achieves high accuracy and F1 scores, comparable to the SAMKNN configurations, while maintaining minimal memory usage (approximately 73.91 KB). Including LTM in SAMKNN leads to a substantial increase in memory consumption, with up to 59,002.88 KB for NF-ToN-IoT, highlighting the efficiency of Adaptive SAMKNN without LTM for memory-constrained environments. Moreover, Adaptive SAMKNN and SAMKNN without LTM demonstrate faster processing times than KNN, significantly reducing the computational load. These findings emphasize Adaptive SAMKNN's capability

to provide a balanced trade-off between accuracy, processing speed, and memory efficiency, making it suitable for real-time IoT applications where computational resources are often limited.

Table 7.18: The accuracy, F1 score, processing time, and memory footprint when different components of the proposed classifier, such as adaptiveness, LTM, and SAM, are removed (ablation study).

Ablation Study	Accuracy	F1	Time (s)	Memory (KB)
NF-BoT-IoT				
Adaptive SAMKNN w/o LTM	98.09%	97.68%	110	73.91
SAMKNN with LTM	97.95%	97.39%	207	39782.40
SAMKNN w/o LTM	98.09%	97.68%	93	33024
KNN	97.74%	97.57%	911	5079.04
NF-ToN-IoT				
Adaptive SAMKNN w/o LTM	96.34%	96.23%	193	73.91
SAMKNN with LTM	95.76%	95.61%	344	59002.88
SAMKNN w/o LTM	96.34%	96.23%	165	55101.44
KNN	94.87%	94.80%	1528	5109.76
NF-CSE-CIC-IDS 2018				
Adaptive SAMKNN w/o LTM	99.92%	99.92%	194	73.91
SAMKNN with LTM	99.91%	99.91%	346	59002.88
SAMKNN w/o LTM	99.92%	99.92%	165	55101.44
KNN	98.96%	98.96%	1528	5038.08

7.6 Discussions

The findings highlight the proposed classifier as a viable solution for real-time anomaly detection in IoT networks, effectively addressing critical constraints in resource-limited environments. Unlike the traditional SAMKNN, which demands substantial memory resources, Adaptive SAMKNN significantly reduces memory consumption while maintaining high accuracy. This efficiency enhances its suitability for dynamic, high-throughput data streams typical in IoT systems. Comparative analyses with standard classifiers reveal Adaptive SAMKNN's competitive advantage, especially in scenarios that require rapid, reliable decision-making with minimal computational overhead. These performance attributes position Adaptive SAMKNN as a key component for scalable and sustainable IoT security frameworks. Future research may focus on its application to larger, more heterogeneous datasets and its integration with other adaptive learning frameworks to bolster resilience against evolving cyber threats within IoT ecosystems.

Experimental results demonstrate Adaptive SAMKNN's proficiency in detect-

ing zero-day attacks with exceptional accuracy and low false positive rates (FPR) across various IoT datasets. Notably, the classifier achieved a perfect detection rate of 1.00 for zero-day attacks, underscoring its robustness in identifying previously unseen threats. Although the NF-BoT-IoT dataset showed a slightly higher FPR, the rates remained within manageable limits, indicating that minor model adjustments could further enhance performance. The classifier's minimal detection latency underscores its applicability in real-time security applications, where swift responses are essential. Adaptive SAMKNN's adaptive nature ensures sustained high accuracy and efficiency compared to traditional classifiers that may falter with evolving threats. Future studies could aim to refine the model to reduce FPR across diverse datasets further and improve its adaptability in more complex IoT environments.

Scalability analysis reveals Adaptive SAMKNN's capability to manage large-scale data effectively, making it ideal for IoT environments with continuously growing data streams. The classifier maintains high accuracy despite increasing sample sizes, demonstrating stability and reliability in monitoring extensive datasets without performance degradation. Its low CPU and memory usage emphasize its efficiency, which is crucial for resource-constrained IoT settings where excessive resource consumption can impede practical deployment. Although processing time gradually increases with larger datasets, Adaptive SAMKNN remains responsive within acceptable limits, preserving its real-time applicability. These results confirm Adaptive SAMKNN's ability to scale efficiently, adapting to high-throughput data while maintaining accuracy and resource efficiency. Future work could explore further optimizations in processing speed to handle even larger data volumes, enhancing its applicability across broader IoT and cybersecurity applications.

Under normal conditions, Adaptive SAMKNN exhibits a zero False Positive Rate (FPR) across all tested datasets, indicating exceptional reliability in distinguishing legitimate network activity without generating unnecessary alerts. This zero FPR is particularly valuable in practical deployments, where false positives can disrupt operations and cause alert fatigue among security personnel. The classifier's precision in differentiating benign traffic from potential threats underscores its effectiveness for real-time IoT deployments, where accurate monitoring is critical for maintaining network integrity. These results suggest that Adaptive

SAMKNN can be reliably deployed in production environments with minimal risk of false alarms, enhancing efficient and focused anomaly detection. Future research could investigate its resilience under dynamic conditions to ensure sustained low FPR in varying operational contexts.

Adaptive SAMKNN demonstrates robust performance under various drift conditions, effectively handling gradual, recurring, and incremental drifts, common in IoT data patterns. This adaptability aligns well with the evolving nature of IoT environments. However, the classifier experiences performance declines in sudden drift scenarios, highlighting a need for enhanced mechanisms such as improved memory adjustment or rapid drift detection to better respond to abrupt changes. While Adaptive SAMKNN is well-suited for stable and gradually changing IoT environments, further research is necessary to bolster its robustness against sudden drifts, ensuring broader applicability across diverse IoT security contexts.

Resource utilization comparisons indicate that Adaptive SAMKNN is more efficient than traditional SAMKNN, making it well-suited for IoT environments with limited computational and memory resources. Its lower CPU and memory usage across all datasets enable effective real-time operation without overburdening system resources, which is critical for IoT network deployments. Reduced processing time supports its adaptability to high-speed data streams, ensuring rapid anomaly detection and response. These efficiency gains align with the goals of IoT security, which prioritize minimizing resource consumption to enable continuous monitoring and analysis. Adaptive SAMKNN's demonstrated efficiency suggests its practicality for real-time anomaly detection in IoT applications. Future investigations could assess its scalability in even more constrained environments and explore its integration into lightweight IoT frameworks for broader deployment.

The statistical analysis underscores the robustness and reliability of the proposed classifier, as evidenced by its high accuracy and tight confidence intervals across diverse IoT datasets. The low Brier scores reflect the model's strong predictive calibration, which is particularly critical for IoT applications that demand reliability. The statistically significant t-tests suggest that the classifier consistently outperforms the SAMKNN baseline, highlighting its superiority in real-world scenarios. Furthermore, the performance on the NF-CSE-2018 dataset, with its near-perfect accuracy, suggests the classifier is highly effective in scenarios with minimal noise or well-structured data. These findings collectively emphasize the potential of the

proposed approach for accurate and reliable IoT network traffic classification.

The ablation study emphasizes the role of Long-Term Memory (LTM) in SAMKNN and the proposed classifier, particularly regarding memory usage and processing efficiency. While LTM enhances SAMKNN's adaptability, it introduces significant memory overhead, making it less feasible for resource-limited IoT environments. In contrast, Adaptive SAMKNN without LTM offers a more memory-efficient solution without sacrificing accuracy, making it preferable for dynamic IoT networks. Additionally, Adaptive SAMKNN shows a substantial reduction in processing time compared to the KNN baseline, reinforcing its suitability for real-time applications that require swift decision-making. These results indicate that Adaptive SAMKNN without LTM strikes an optimal balance for IoT anomaly detection, providing reliable performance while minimizing resource demands.

7.6.1 Limitations

Despite its promising performance, the proposed technique exhibits several limitations that warrant attention. Firstly, the classifier is sensitive to sudden drifts in data distributions, leading to temporary declines in detection accuracy during abrupt changes. This sensitivity underscores the necessity for more robust drift detection mechanisms to enhance responsiveness to rapid shifts in network behavior. Secondly, the proposed classifier's effectiveness is somewhat dependent on the specific datasets used for training and evaluation. The model may not generalize uniformly across all types of IoT and IIoT datasets, particularly those with unique features, potentially limiting its applicability in diverse real-world scenarios. Additionally, while the proposed algorithm is designed to be resource-efficient, it still requires a baseline level of computational and memory resources that may be challenging for some extremely low-resource-constrained IoT devices. This constraint can limit the deployment of the classifier on lower-end devices or in environments where resources are highly limited. Addressing these limitations through targeted enhancements, such as incorporating advanced drift detection algorithms and optimizing resource utilization, is essential for improving the robustness and versatility of the proposed algorithm in practical IoT deployments.

7.6.2 Real-World Deployment Scenarios and Challenges

Deploying the proposed algorithm in real-world IoT and IIoT environments presents a multitude of promising opportunities alongside significant challenges. In practical scenarios, the proposed algorithm can be integrated into diverse applications such as smart grids, industrial automation systems, healthcare monitoring networks, and smart home infrastructures. For instance, in smart grids, the model can continuously monitor network traffic to detect and mitigate cyber threats that could disrupt power distribution and critical services. Similarly, in industrial automation, the model can protect automated machinery and control systems from malicious intrusions that could lead to operational downtime or safety hazards.

However, the deployment of the proposed classifier is not without its challenges. One of the primary obstacles is the integration with existing legacy systems, which may have limited compatibility with modern machine learning frameworks. Additionally, the resource-constrained nature of many IoT devices poses another challenge, as deploying even lightweight models can strain limited computational and memory resources. Although the proposed classifier is designed for efficiency, optimizing its deployment to operate within the stringent constraints of edge devices necessitates further refinement.

Real-time processing requirements add another layer of complexity, as the model must deliver prompt detection and response to threats without introducing latency that could disrupt critical operations. Achieving this necessitates robust optimization techniques and possibly distributed processing architectures to balance the load effectively.

Scalability is also a critical factor, as real-world IoT deployments often involve vast and continuously growing networks of devices. The proposed algorithm must maintain its performance and accuracy across expanding datasets and increasingly complex network topologies.

7.7 Conclusion

This study demonstrates that Adaptive SAMKNN is a highly effective and efficient solution for anomaly detection in IoT environments. Through extensive evaluations, Adaptive SAMKNN proves its robustness in handling various datasets, including NF-BoT-IoT, NF-ToN-IoT, and NF-CSE-CIC-IDS 2018, achieving high

accuracy, low memory usage, and minimal CPU consumption. The model's consistent performance across these datasets, even with large data volumes, underscores its scalability, making it a suitable candidate for real-time applications in resource-constrained IoT networks.

One of the key strengths of Adaptive SAMKNN is its ability to maintain a zero false positive rate under normal conditions, a critical factor in ensuring the reliability of IoT security frameworks. This capability reduces alert fatigue, enabling security teams to focus on genuine threats without being overwhelmed by false alarms. The model's low false positive rate and high detection rate make it a dependable choice for distinguishing between benign and malicious network activities, even in dynamic and evolving environments.

The study also explores the impact of different types of concept drift, revealing that Adaptive SAMKNN effectively adapts to gradual, recurring, and incremental drifts. However, it encounters challenges in sudden drift scenarios, where performance slightly declines. This limitation suggests an opportunity for further improvements, such as incorporating rapid drift detection mechanisms to enhance responsiveness in highly volatile environments.

Ablation studies further reveal that Adaptive SAMKNN's performance remains robust even without long-term memory, significantly reducing memory usage while maintaining accuracy and F1 scores. This makes the model highly efficient for real-time applications, where minimizing resource consumption is essential. Additionally, the scalability assessment confirms that Adaptive SAMKNN can handle increasing data volumes without sacrificing performance, highlighting its potential for broader IoT deployments.

Future research directions include improving the model's adaptability to sudden drift and exploring its integration with other adaptive learning frameworks. To do this and integrate seamlessly with lightweight IoT devices, we can incorporate advanced drift detection algorithms like ADWIN and DDM into the Adaptive SAMKNN framework to swiftly identify abrupt data distribution changes. Additionally, we can further optimize the model for resource-constrained environments through techniques such as model pruning, quantization, and the use of lightweight data structures to reduce memory and processing requirements.

References

[1] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey", *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[2] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in industrial internet of things: Architecture, advances and challenges", *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2462–2488, 2020.

[3] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (iiot): An analysis framework", *Computers in industry*, vol. 101, pp. 1–12, 2018.

[4] H. Darabian, A. Dehghantanha, S. Hashemi, S. Homayoun, and K.-K. R. Choo, "An opcode-based technique for polymorphic internet of things malware detection", *Concurrency and Computation: Practice and Experience*, vol. 32, no. 6, e5173, 2020.

[5] H. Karimipour, A. Dehghantanha, R. M. Parizi, K.-K. R. Choo, and H. Leung, "A deep and scalable unsupervised machine learning system for cyber-attack detection in large-scale smart grids", *IEEE Access*, vol. 7, pp. 80 778–80 788, 2019.

[6] Z. Moti, S. Hashemi, and A. Namavar, "Discovering future malware variants by generating new malware samples using generative adversarial network", in *2019 9th International conference on computer and knowledge engineering (ICCKE)*, IEEE, 2019, pp. 319–324.

[7] A. N. Jahromi, S. Hashemi, A. Dehghantanha, *et al.*, "An improved two-hidden-layer extreme learning machine for malware hunting", *Computers & Security*, vol. 89, p. 101 655, 2020.

[8] A. P. Ekong, A. Etuk, S. Inyang, and M. Ekere-obong, "Securing against zero-day attacks: A machine learning approach for classification and organizations' perception of its impact", *Journal of Information Systems and Informatics*, vol. 5, no. 3, pp. 1123–1140, 2023.

[9] S. Oluwadare and Z. ElSayed, "A survey of unsupervised learning algorithms for zero-day attacks in intrusion detection systems.", in *The International FLAIRS Conference Proceedings*, vol. 36, 2023.

[10] G. Guo, "An intrusion detection system for the internet of things using machine learning models", in *2022 3rd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, IEEE, 2022, pp. 332–335.

[11] H. Teymourlouei, D. Stone, and L. Jackson, "Identifying zero-day attacks with machine learning and data reduction methods", in *2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE)*, IEEE, 2023, pp. 2285–2290.

[12] P. R. Agbedanu, S. Jay Yang, R. Musabe, I. Gatare, and J. Rwigema, "An online adaptive approach to detecting zero-day attacks in iot and iiot systems", in *2024 IEEE Globecom Workshops (GC Wkshps)*, 2024, pp. 1–7. DOI: 10.1109/GCWkshp64532.2024.11101063.

[13] S. I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh, and O. Jogunola, "Federated deep learning for zero-day botnet attack detection in iot-edge devices", *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3930–3944, 2021.

[14] J. Zhang, S. Liang, F. Ye, R. Q. Hu, and Y. Qian, "Towards detection of zero-day botnet attack in iot networks using federated learning", in *ICC 2023-IEEE International Conference on Communications*, IEEE, 2023, pp. 7–12.

[15] B. I. Hairab, M. S. Elsayed, A. D. Jurcut, and M. A. Azer, "Anomaly detection based on cnn and regularization techniques against zero-day attacks in iot networks", *IEEE Access*, vol. 10, pp. 98 427–98 440, 2022.

[16] B. Ibrahim Hairab, H. K. Aslan, M. S. Elsayed, A. D. Jurcut, and M. A. Azer, "Anomaly detection of zero-day attacks based on cnn and regularization techniques", *Electronics*, vol. 12, no. 3, p. 573, 2023.

[17] A. Alsaleh, "A novel intrusion detection model of unknown attacks using convolutional neural networks.", *Computer Systems Science & Engineering*, vol. 48, no. 2, 2024.

[18] T. Ohtani, R. Yamamoto, and S. Ohzahata, "Idac: Federated learning-based intrusion detection using autonomously extracted anomalies in iot", *Sensors*, vol. 24, no. 10, p. 3218, 2024.

[19] A. A. Korba, A. Boualouache, B. Brik, R. Rahal, Y. Ghamri-Doudane, and S. M. Senouci, "Federated learning for zero-day attack detection in 5g and beyond v2x networks", in *ICC 2023-IEEE International Conference on Communications*, IEEE, 2023, pp. 1137–1142.

[20] M. Anwer, G. Ahmed, A. Akhunzada, S. Hussain, and M. Khan, "Comparative analysis of soft computing approaches of zero-day-attack detection", in *2022 International Conference on Emerging Trends in Smart Technologies (ICETST)*, IEEE, 2022, pp. 1–5.

[21] D. Jin, S. Chen, H. He, X. Jiang, S. Cheng, and J. Yang, "Federated incremental learning based evolvable intrusion detection system for zero-day attacks", *Ieee Network*, vol. 37, no. 1, pp. 125–132, 2023.

[22] Z. He and H. Sayadi, "Image-based zero-day malware detection in iomt devices: A hybrid ai-enabled method", in *2023 24th International Symposium on Quality Electronic Design (ISQED)*, IEEE, 2023, pp. 1–8.

[23] M. Asaduzzaman and M. M. Rahman, "An adversarial approach for intrusion detection using hybrid deep learning model", in *2022 International Conference on Information Technology Research and Innovation (ICITRI)*, IEEE, 2022, pp. 18–23.

[24] A. Armijos and E. Cuenca, "Zero-day attacks: Review of the methods used based on intrusion detection and prevention systems", in *2023 IEEE Colombian Caribbean Conference (C3)*, IEEE, 2023, pp. 1–6.

[25] K. Saurabh, V. Sharma, U. Singh, R. Khondoker, R. Vyas, and O. Vyas, "Hms-ids: Threat intelligence integration for zero-day exploits and advanced persistent threats in iiot", *Arabian Journal for Science and Engineering*, pp. 1–21, 2024.

[26] S. Guo, T. Sivanthi, P. Sommer, *et al.*, "A zero-day container attack detection based on ensemble machine learning", in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2023, pp. 1–8.

[27] M. Nkongolo and M. Tokmak, "Zero-day threats detection for critical infrastructures", in *Annual Conference of South African Institute of Computer Scientists and Information Technologists*, Springer, 2023, pp. 32–47.

[28] P. Verma, A. Dumka, R. Singh, *et al.*, "A novel intrusion detection approach using machine learning ensemble for iot environments", *Applied Sciences*, vol. 11, no. 21, p. 10 268, 2021.

[29] R. Ahmad, I. Alsmadi, W. Alhamdani, and L. Tawalbeh, "A deep learning ensemble approach to detecting unknown network attacks", *Journal of Information Security and Applications*, vol. 67, p. 103 196, 2022.

[30] Y. A. Farrukh, S. Wali, I. Khan, and N. D. Bastian, "Detecting unknown attacks in iot environments: An open set classifier for enhanced network intrusion detection", in *MILCOM 2023-2023 IEEE Military Communications Conference (MILCOM)*, IEEE, 2023, pp. 121–126.

[31] A. H. Celdrán, P. M. S. Sánchez, J. Von Der Assen, *et al.*, "Rl and fingerprinting to select moving target defense mechanisms for zero-day attacks in iot", *IEEE Transactions on Information Forensics and Security*, 2024.

[32] E. Rodríguez, P. Valls, B. Otero, *et al.*, "Transfer-learning-based intrusion detection framework in iot networks", *Sensors*, vol. 22, no. 15, p. 5621, 2022.

[33] M. Ellouh, M. Ghaleb, and M. Felemban, "Iotzerojar: Towards a honeypot architecture for detection of zero-day attacks in iot", in *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*, IEEE, 2022, pp. 765–771.

[34] Z. Li and D. Zhao, "μThingnet: Leveraging fine-grained power analysis towards a robust zero-day defender", in *2023 IEEE 36th International System-on-Chip Conference (SOCC)*, IEEE, 2023, pp. 1–6.

[35] M. A. Sayed, A. H. Anwar, C. Kiekintveld, B. Bosansky, and C. Kamhoua, "Cyber deception against zero-day attacks: A game theoretic approach", in *International Conference on Decision and Game Theory for Security*, Springer, 2022, pp. 44–63.

[36] V. Losing, B. Hammer, and H. Wersing, "Knn classifier with self adjusting memory for heterogeneous concept drift", in *2016 IEEE 16th international conference on data mining (ICDM)*, IEEE, 2016, pp. 291–300.

[37] J. Montiel, M. Halford, S. M. Mastelini, *et al.*, "River: Machine learning for streaming data in python", *J. Mach. Learn. Res.*, vol. 22, no. 1, Jan. 2021, ISSN: 1532-4435.

[38] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets", *Mobile networks and applications*, pp. 1–14, 2022.

[39] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan", *Advances in neural information processing systems*, vol. 32, 2019.

This page has been intentionally left blank.

Chapter 8

Chapter 8 is published as:

P. R. Agbedanu, S. Jay Yang, R. Musabe, I. Gatare and J. Rwigema, "An Online Adaptive Approach to Detecting Zero-day Attacks in IoT and IIoT Systems," *2024 IEEE Globecom Workshops (GC Wkshps)*, Cape Town, South Africa, 2024, pp. 1-7, doi: 10.1109/GCWkshp64532.2024.11101063.

An Online Adaptive Approach to Detecting Zero-day Attacks in IoT and IIoT Systems

Authors: Promise Ricardo Agbedanu, Sanchieh Jay Yang, Richard Musabe, Ignace Gatare, James Rwigema

Abstract: Traditional intrusion detection systems (IDS) struggle to detect zero-day attacks because they rely on pre-defined signatures. This paper proposes an online adaptive machine learning approach, Self-Adjusting Memory K-Nearest Neighbors (Adaptive SAMKNN), to detect zero-day attacks in Internet of Things (IoT) and Industrial IoT (IIoT) environments. The method dynamically adjusts memory allocation, enabling real-time detection with minimal memory usage. Experimental results, conducted using two datasets (NF-BoT-IoT and NF-ToN-IoT), show that Adaptive SAMKNN consistently outperforms traditional offline machine learning models in accuracy and memory efficiency. The system also successfully detects synthetic zero-day attacks created through generative adversarial networks (GANs). This approach demonstrates the feasibility of deploying lightweight, resource-efficient intrusion detection systems in resource-constrained IoT/IIoT environments.

8.1 Introduction

The Internet of Things (IoT) and the Industrial Internet of Things (IIoT) have revolutionized how industrial systems are designed, operated, and managed, increasing efficiency, reliability, and flexibility. However, these systems' increased connectivity has also presented new and complex security challenges, as they are more vulnerable to cyber attacks that can disrupt critical infrastructure and compromise sensitive data [1].

In recent years, there has been a surge in the number of cyber attacks targeting

industrial systems, raising concerns about the security of the IIoT [2], [3]. Cyber attacks pose a significant threat to the IIoT and IoT, leading to several studies trying to solve this problem [4], [5].

A particularly concerning threat is the emergence of zero-day attacks, which exploit previously unknown vulnerabilities before they can be detected and mitigated [6]. Traditional signature-based intrusion detection systems (IDS) struggle to detect such novel attacks effectively, as they rely on prior knowledge of attack patterns [7]. Researchers have explored machine learning (ML) techniques to develop more adaptive and intelligent IDS for IoT environments [8]. Online ML models, in particular, offer a promising approach for detecting zero-day attacks in resource-constrained IoT systems [9]. By continuously learning from network traffic data, these models can adapt to evolving attack patterns without the need for extensive retraining [9].

Most intrusion detection methods are not well-suited to the unique requirements of IoT and IIoT because they are computationally expensive, need to be retrained in an off-production mode to adapt to new attack trends, and cannot detect zero-day attacks. Zero-day attacks exploit unknown vulnerabilities, making traditional signature-based IDS ineffective.

Based on the drawbacks of current systems that detect cyber-attacks in IoT/IIoT, it is important to use a more effective and efficient solution to detect attacks within the IoT/IIoT ecosystem. This solution should be able to detect zero-day attacks and be computationally inexpensive. Online ML algorithms provide real-time adaptability and reduced latency, making them suitable for IoT/IIoT systems and other resource-constrained devices.

This paper proposes an IDS for IoT/IIoT using an online machine learning (ML) technique to develop an adaptive system that identifies previously unseen threats in real-time, enhancing the security of IoT/IIoT networks.

The key contributions of this research are as follows:

- An IDS for IoT and IIoT systems that detects zero-day attacks efficiently, with low memory usage and processing time, using an adaptive memory allocation technique with SAMKNN.
- An optimized memory operation for Short-Term Memory (STM) and Long-Term Memory (LTM) using a Doubly Ended Queue (deque) for efficient

storage and retrieval.

- A label encoding and decoding mechanism that ensures flexible and efficient label handling, optimizing storage and lookup operations during prediction and learning phases.
- Demonstrated improved detection rates for zero-day attacks in IoT and IIoT systems through adaptive online machine learning.

The remainder of this paper is organized as follows: Section II focuses on the related works considered in this study. In Section III, we present our proposed methodology. Section IV presents the experimental design of this study, with the experimental results and discussions presented in Section V. The study is concluded in Section VI.

8.2 Related Work

Detecting zero-day attacks in IoT environments has been a concern for many years. As such, several works have been done to solve this problem. Popoola et al., [10], proposed an optimal deep neural network (DNN) architecture to detect zero-day attacks in IoT systems. The DNN was used with federated ML, where the federated algorithm aggregates local model updates. Due to IoT devices' limited computational power, memory, and battery life, training and updating complex DNN models or performing federated learning on these devices can be resource-intensive.

Federated learning has emerged as a promising approach to enhance detection capabilities against zero-day attacks by sharing attack information among multiple IoT networks [11], [12]. By utilizing federated learning, researchers have developed frameworks that improve the detection of zero-day attacks while preserving user privacy and minimizing communication overhead [13]. To detect zero-day attacks, [14] developed an IDS model based on federated incremental learning that aggregates knowledge from different detectors and then incrementally updates the model. IoT devices often generate heterogeneous data due to differences in device types, manufacturers, and deployment environments. Effectively aggregating models in federated learning to account for this heterogeneity is complex and can lead to sub-optimal performance.

A hybrid approach combining ML and DL techniques has been proposed to combat zero-day attacks for effective detection and mitigation [15], [16]. Saurabh et al., [17] proposed a hybrid IDS using supervised and unsupervised ML approaches to detect known and unknown attacks. Hybrid models combining ML, DL, and DRL are computationally intensive. IoT devices often have limited processing power, memory, and battery life, making it challenging to run these complex models locally.

Research has shown the effectiveness of ensemble learning in detecting zero-day attacks in IoT networks [12], [18], [19]. The use of ensemble learning, particularly with Random Forest (RF) and Extreme Gradient Boosting (XGB) algorithms, has been identified as top-performing in detecting zero-day attacks in IoT systems, outperforming previous methods and enhancing the performance of machine learning models [19]. In an earlier study, [20] developed an ensemble classifier using processed data packets that can detect anomalies and protect IoT systems against zero-day attacks. Additionally, leveraging ensemble learning in a federated framework for IoT networks can achieve superior model aggregation without compromising user privacy, showcasing its efficacy in zero-day botnet attack detection [12]. Ensemble learning models can be computationally intensive, particularly those involving multiple algorithms like RF and XGB. This complexity increases the resource requirements, which can be problematic for resource-constrained IoT devices.

Zero-day attacks targeting IoT devices have become a significant concern due to the vulnerabilities in interconnected devices [21]. ML techniques have also emerged as a powerful tool for detecting zero-day attacks by analyzing patterns in network traffic and user behavior, enhancing cybersecurity defenses against these elusive threats [22]. The computationally expensive nature of most ML techniques makes them unsuitable for use in IoT and IIoT systems.

8.3 Proposed Methodology

The diagram in Figure 8.1 depicts the architecture of our proposed IDS designed for IoT/IIoT environments using SAMKNN with an adaptive STM and LTM. The architecture begins with IoT/IIoT devices feeding data into a preprocessing unit, which filters and prepares the incoming network traffic data for analysis. This

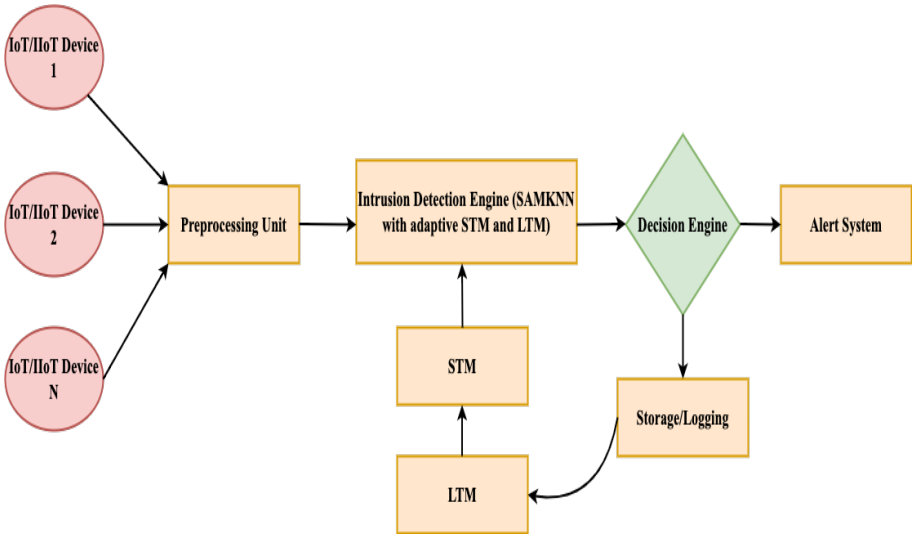


Figure 8.1: The architecture of our proposed zero-day attack intrusion detection system for IoT and IIoT systems

preprocessed data is then passed to the Intrusion Detection Engine, which employs the Adaptive SAMKNN algorithm. The engine utilizes STM and LTM to retain and analyze the historical and real-time patterns in network traffic, ensuring that the system remains adaptable to evolving threats. Once the detection engine processes the data, it sends the results to the decision engine, determining whether the analyzed activity is malicious or benign. If the traffic is flagged as malicious, the decision engine triggers the alert system to notify the user of a potential threat. Simultaneously, the system logs and stores important data for future analysis and auditing purposes. This architecture balances efficiency and adaptability, allowing the IDS to detect zero-day attacks while optimizing memory usage and processing power.

8.3.1 Data Preprocessing

Because the datasets used for validating our experiment were re-created from the pcap files of their original datasets, there were no missing values. The attack label, source, and destination IP addresses were dropped during our data processing procedure. We also scaled the other features before feeding them to the model because distance-based algorithms like KNN are sensitive to the scale of features.

8.3.2 Model training and testing

To train and test our proposed system, we designed an adaptive version of SAMKNN. The original SAMKNN was proposed by [23]. Making SAMKNN adaptive allows it to adjust its memory allocation dynamically. Because Adaptive SAMKNN is

an online ML model, the proposed system trains and tests data incrementally, one data stream at a time. The model updates their parameters and makes predictions as each data point arrives. We used this approach because IoT and IIoT devices do not have the computational capacity to load huge datasets into memory. The proposed system updates its parameters incrementally as each new data sample arrives. Regarding prediction, for each sample, the proposed model makes a prediction, computes the prediction error, and updates the parameters.

8.3.3 Adaptive SAMKNN

In this subsection, we explain how we implemented the proposed Adaptive SAMKNN based on a modification of the original SAMKNN proposed by [23].

The proposed algorithm maintains two types of memories: STM, which stores recent data, and LTM, which retains historical patterns. STM quickly adapts to new trends, while LTM captures long-term behaviors. For each new data sample X, the algorithm encodes the label y and makes a prediction using a distance-weighted K-Nearest Neighbors (KNN) approach. Predictions are based on both STM and LTM when applicable. The algorithm dynamically adjusts the balance between STM and LTM based on model performance. If the accuracy improves, the LTM size increases to retain more historical data. Conversely, if performance drops, the LTM size is reduced, allowing the model to focus on recent data.

The model's performance is tracked through a sliding window of recent predictions, and memory adjustments are made when sufficient data is available, regulated using a cool-down period. The cool-down technique ensures efficient memory usage and maintains accuracy in stationary and non-stationary environments, adapting to concept drift over time. A simplified algorithmic pseudocode of our proposed Adaptive SAMKNN is shown in Algorithm 5.

Algorithm 5 Adaptive SAMKNN Classifier

Input: Feature vector $\mathbf{x}$, label y
Output: Predicted label $\hat{y}$
{— Initialization —}
$STM\_Samples \leftarrow \emptyset$, $STM\_Labels \leftarrow \emptyset$
$LTM\_Samples \leftarrow \emptyset$, $LTM\_Labels \leftarrow \emptyset$
$label\_map \leftarrow \{\}$, $next\_idx \leftarrow 0$
$perf\_window \leftarrow$ deque(maxlen=$perf\_size$)
$best\_acc \leftarrow$ None
$cooldown \leftarrow perf\_cooldown$
{— Training Step —}
if $y \notin label\_map$ **then**
    Assign y a new index in $label\_map$
end if
$\hat{y} \leftarrow$ `PredictOne`($\mathbf{x}$)
Append $\mathbf{x}$ and encoded y to $STM\_Samples$, STM_Labels
if $|STM\_Samples| > STM\_max$ **then**
    Move oldest sample to $LTM\_Samples$ and label to LTM_Labels
end if
Append prediction accuracy to $perf\_window$
{— Adjust Memory Sizes —}
if $|perf\_window| = perf\_size$ and cooldown has passed **then**
    $curr\_acc \leftarrow \frac{\sum perf\_window}{perf\_size}$
    if $best\_acc$ = None **or** $curr\_acc > best\_acc$ **then**
        Increase ltm_size
        $best\_acc \leftarrow curr\_acc$
    else if $curr\_acc < best\_acc$ **then**
        Reduce ltm_size
        $best\_acc \leftarrow curr\_acc$
    end if
    Reset cooldown
else
    Increment cooldown counter
end if
{— Prediction —}
if using both STM and LTM **then**
    Compute distance between $\mathbf{x}$ and samples in both STM and LTM
else
    Compute distance only from STM
end if
Find k nearest neighbors
Perform weighted vote for $\hat{y}$
Return $\hat{y}$
{— Metric Calculation —}
`GetMetrics:` Return accuracy as the fraction of correct predictions

8.4 Experimental Design

8.4.1 Experimental Setup

Python 3.10 was utilized throughout the experimental validation to develop the proposed system using the River online ML framework [24]. We chose the River online ML framework because it can process streaming data. We used a MacBook Pro with an M1 chip running on 16 GB of RAM and 2TB of disk storage to model the proposed system.

8.4.2 Datasets

We used two datasets to build and test our proposed IDS. The datasets are NF-BoT-IoT and NF-ToN-IoT proposed by [25]. The NetFlow records of these datasets were generated from the publicly available pcap files of the respective datasets. Table 8.1 summarizes the dataset used for our experimental validation.

Table 8.1: Summary of the dataset we used for our experimental validation

#Features	#Samples	Benign	Attack	#Attack Category
NF-BoT-IoT				
12	600,100	13,859	586,241	4
NF-ToN-IoT				
12	1,379,274	270,279	1,108,995	6

8.4.3 Evaluation Metrics

The matrices used to evaluate the performance of our proposed system were accuracy, precision, recall, F1 score, model training and testing time, and memory usage.

8.4.4 Experimental Validations

Four experiments were conducted to test and validate our proposed system. The experiments are briefly described below.

8.4.4.1 Baseline Performance Evaluation

The first experiment evaluated the performance of the Adaptive SAMKNN classifier on the two datasets while recording accuracy, F1 score, processing time per sample, and memory usage. As part of the baseline performance evaluation, we

also evaluated the performance of the proposed technique on each dataset's various attack classes. We also compared the performance and memory usage of the proposed Adaptive SAMKNN with the original SAMKNN.

8.4.4.2 Comparison with offline ML classifiers

The second experiment compared the performance of our proposed technique to five popular traditional offline ML algorithms on the two datasets while recording the accuracy, F1 score, processing time per sample, and memory usage. The offline machine algorithms considered are Random Forest (RF), k-nearest Neighbors (KNN), Support Vector Machines (SVM), Decision Tree (DT), and Logistic Regression (LR).

8.4.4.3 Zero-day attack detection

We evaluated our proposed system's ability to detect zero-day attacks during the third experiment. We created zero-day attacks using a generative adversarial network (GAN) to create synthetic attack categories of the respective datasets that differ from the original. We augmented each attack category with synthetic zero days and tested them on the trained model. The prediction performance was then recorded.

8.4.4.4 Statistical Testing

To assess the proposed model's robustness and ensure the results are replicable and not randomized. We compared the performance of the proposed algorithm with the original SAMKNN algorithm using a paired t-test approach, where the mean performance of these two algorithms was compared. During the statistical testing, the experiment was run five times, and the mean accuracy, the standard deviation, t-statistics, and p-values were computed.

8.5 Results

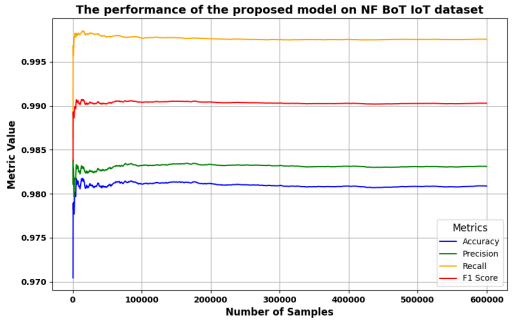
8.5.1 Baseline Performance Evaluation

For the NF-BoT-IoT dataset, as shown in Table 8.2, which consisted of 600,077 samples, the Adaptive SAMKNN algorithm achieved an accuracy of 98.08% and an F1 score of 97.68%. The processing time was remarkably low at 0.02 milliseconds,

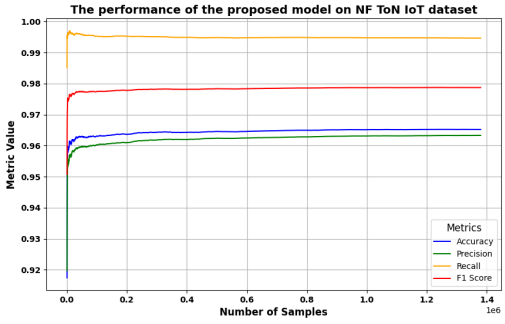
and the memory usage was 64.64 KB. In the case of the NF-ToN-IoT dataset, as shown in Table 8.2, containing 1,379,264 samples, the algorithm recorded an accuracy of 96.56% and an F1 score of 96.47%. The processing time recorded is the same as when the algorithm was executed using the NF-BoT-IoT dataset. However, memory usage was slightly lower than that recorded on the NF-BoT-IoT dataset, which recorded 64.12 KB of memory usage. A plot of the performance of the proposed system on each dataset is shown in 8.2.

Table 8.3 shows Adaptive SAMKNN's performance on each dataset's attack categories. On the NF-BoT-IoT dataset, Adaptive SAMKNN recorded an F1 score above 97% on all the attack categories and the lowest memory usage of 61.34 KB on the Theft attack category. Similarly, on the NF-ToN-IoT dataset shown in 8.3, Adaptive SAMKNN recorded an F1 score over 94% on all attack categories, with the lowest memory usage of 63.03 KB on the Scan attack category.

As shown in Table 8.4, Adaptive SAMKNN recorded an accuracy of 98.09% and 96.56% on NF-BoT-IoT and NF-ToN-IoT datasets, respectively, against 97.73% and 95.67% recorded by SAMKNN. Our proposed algorithm recorded a significantly lower memory usage of 52.54 KB when evaluated with either dataset. SAMKNN recorded a memory usage of 35758.08 KB and 106905.60 KB on the NF-BoT-IoT and NF-ToN-IoT datasets, respectively.



(a) The performance of our proposed model on the NF-BoT-IoT dataset



(b) The performance of our proposed model on the NF-ToN-IoT dataset

Figure 8.2: The performance of our proposed model on the NF-BoT-IoT and NF-ToN-IoT datasets

Table 8.2: The performance of Adaptive SAMKNN on the two datasets

# Samples	Accuracy	F1	Time (ms)	Memory (KB)
NF-BoT-IoT				
600077	98.09%	97.68%	0.02	64.64
NF-ToN-IoT				
1379264	96.56%	96.47%	0.02	64.12

Table 8.3: The performance of Adaptive SAMKNN on each attack category of the two datasets

Attack Category	#Samples	F1	Time(s)	Memory (KB)
NF-BoT-IoT				
DDoS	70702	98.87%	0.02	64.64
DoS	70691	98.89%	0.02	64.62
Recon	484513	97.32%	0.02	64.64
Theft	15767	98.67%	0.02	61.34
NF-ToN-IoT				
DDoS	329268	94.92%	0.02	64.58
DoS	187180	94.67%	0.02	64.58
Injection	859994	96.80%	0.02	63.09
Malware	197036	99.11%	0.02	63.31
MITM	180862	98.99%	0.02	64.58
Scan	192470	98.58%	0.02	63.03

Table 8.4: Comparing the performance of Adaptive SAMKNN with SAMKNN

Algorithm	Accuracy	F1	Memory (KB)
NF-BoT-IoT			
SAMKNN	97.73%	96.91%	35758.08
Adaptive SAMKNN	98.09%	97.68%	52.54
NF-ToN-IoT			
SAMKNN	95.67%	95.54%	106905.60
Adaptive SAMKNN	96.56%	96.47%	52.54

8.5.2 Comparing Adaptive SAMKNN with offline machine learning classifiers

On the NF-BoT-IoT dataset, as shown in Table 8.5, the Adaptive SAMKNN showed an accuracy of 98.09% and an F1 score of 97.68%, with a processing time of 0.02 ms per sample and minimal memory usage of 64.64 KB. RF and DT showed high accuracy and low processing times, but required significantly more memory. SVM had lower F1 scores and much higher processing times and memory usage. On the NF-ToN-IoT dataset, as shown in Table 8.5, Adaptive SAMKNN achieved 96.56% accuracy and 96.47% F1 score, with a processing time of 0.02 ms and low memory usage of 64.12 KB. RF and DT were again top performers in accuracy and F1, but used substantial memory. KNN and SVM exhibited higher processing times, with SVM again having a larger memory footprint.

8.5.3 Zero-day attack detection

The results in Table 8.6 demonstrate the effectiveness of the Adaptive SAMKNN model in detecting zero-day attacks across different datasets (NF-BoT-IoT and NF-

Table 8.5: Comparing the performance of five popular Offline ML algorithms to Adaptive SAMKNN on the two datasets

Algorithm	Accuracy	F1	Time (ms)	Memory (KB)
NF-BoT-IoT				
RF	98.97%	88.36%	0.03	703662.08
KNN	99.16%	89.89%	0.04	703979.52
SVM	98.86%	83.78%	2.16	1193441.28
DT	99.05%	89.53%	0.003	807956.48
LR	98.74%	81.07%	0.007	829204.48
Adaptive SAMKNN	98.09%	97.68%	0.02	64.64
NF-ToN-IoT				
RF	99.95%	99.93%	0.09	1700587.52
KNN	99.57%	99.32%	0.07	1549025.28
SVM	98.40%	97.38%	1.67	1795328
DT	99.94%	99.90%	0.004	744765.44
LR	97.12%	95.31%	0.002	761579.52
Adaptive SAMKNN	96.56%	96.47%	0.02	64.12

ToN-IoT). In presenting the results, accuracy 1 represents the test accuracy when the model is built with the attack categories. In contrast, accuracy 2 represents the model's test accuracy when built with the entire dataset. When trained on individual attack categories, the model achieves high performance, with accuracy and F1 scores consistently exceeding 92% across most attack types. For instance, in the NF-ToN-IoT dataset, the model reaches 99.50% accuracy for GAN Malware detection, while for GAN Injection, it maintains an accuracy of 97.15%. However, when trained on the entire dataset, the model's performance varies significantly depending on the attack type. The accuracy for GAN Theft detection in NF-BoT-IoT drops drastically to 43.57%, while other categories, such as GAN Recon, maintain a high accuracy of 98.03%.

Table 8.6: The performance of Adaptive SAMKNN in detecting zero-day attacks when the model is built with the respective attack category and when the model is built with the respective dataset.

Attack Category	Accuracy 1	Accuracy 2
NF-BoT-IoT		
GAN DDoS	92.43%	86.80%
GAN DoS	98.69%	86.80%
GAN Recon	98.25%	98.03%
GAN Theft	97.02%	43.57%
NF-ToN-IoT		
GAN DDoS	95.56%	88.47%
GAN DoS	94.68%	79.77%
GAN Injection	97.15%	95.58%
GAN Malware	99.50%	80.77%
GAN MITM	98.78%	79.06%
GAN Scanning	98.12%	80.32%

8.5.4 Statistical Testing

From Table 8.7, Adaptive SAMKNN recorded a mean accuracy of 98.15% after five runs when it was statistically evaluated on the NF-BoT-IoT dataset. It recorded a standard deviation of 0.0002 and a confidence interval of 98.13% to 98.17%. On the other hand, SAMKNN recorded a mean accuracy of 97.78%, a standard deviation of 0.0002, and a confidence interval of 97.75% to 97.80%. A paired t-test indicated that the difference in accuracy between the two classifiers was statistically significant, with a t-statistic value of 53.6448 and a p-value less than 0.0001. Comparing the statistical performance of the two algorithms on the NF-ToN-IoT dataset, as shown in Table 8.7, Adaptive SAMKNN recorded a mean accuracy of 96.15% against 94.81% recorded by SAMKNN. The proposed algorithm recorded a standard deviation of 0.0003, while SAMKNN recorded a standard deviation of 0.0007. A confidence interval of 96.38% to 96.44% was recorded by the proposed algorithm. On the other hand, SAMKNN recorded a confidence interval of 94.72% to 94.89%. A paired t-test indicated that the difference in accuracy between the two classifiers was statistically significant, with a t-statistic value of 41.2643 and a p-value less than 0.0001.

Table 8.7: Comparing the statistical performance of SAMKNN and Adaptive SAMKNN using the two datasets

Algorithm	**Mean Accuracy**	**std**	**95% CI**
NF-BoT-IoT			
SAMKNN	97.78%	0.0002	0.9775, 0.9780
Adaptive SAMKNN	98.15%	0.0002	0.9813, 0.9817
NF-ToN-IoT			
SAMKNN	94.81%	0.0007	0.9472, 0.9489
Adaptive SAMKNN	96.41%	0.0003	0.9638, 0.9644

8.5.5 Comparing our proposed model to other state-of-the-art works

Table 8.8 compares the proposed Adaptive SAMKNN model and other state-of-the-art techniques. In the NF-BoT-IoT dataset, RF and MLP achieve near-perfect results, with both F1 scores and accuracy reaching 100%, thereby outperforming Adaptive SAMKNN, which scores 98.09% in accuracy and 97.68% in F1. In the NF-ToN-IoT dataset, while RF demonstrates flawless performance again, the Adaptive SAMKNN model achieves 96.56% accuracy and 96.47% F1, which is

notably lower than RF and Extra Trees but still higher than MLP. These results indicate that although the Adaptive SAMKNN model does not match the perfect accuracy of RF, it remains a strong contender in terms of overall performance, offering competitive results across multiple metrics.

Regarding computational cost, all the state-of-the-art algorithms considered in this work are more computationally expensive than Adaptive SAMKNN.

8.5.6 Discussions

As shown in 8.3, the consistently high F1 scores and low memory usage in the different attack categories show that Adaptive SAMKNN can be effectively deployed in environments with computational constraints.

Table 8.4 shows Adaptive SAMKNN exhibiting slightly higher performance than SAMKNN and recording a significantly lower memory usage. The results show that a practical deployment of SAMKNN in IoT environments will be challenging due to the model's high memory usage. The balance of speed, efficiency, and low memory usage of Adaptive SAMKNN makes it a suitable IDS for use in dynamic environments where quick decision-making is crucial.

Table 8.6 shows the performance of the Adaptive SAMKNN model in detecting zero-day attacks under two different conditions: when the model is trained using a specific attack category and when trained using a dataset as a whole. The accuracy 1 in Table 8.6 shows that the model performs exceptionally well in detecting attacks when trained per attack type. However, accuracy 2 in Table 8.6 shows a performance drop when the model is trained on the entire dataset rather than individual attack categories. For example, in the NF-BoT-IoT dataset, GAN DDoS and GAN DoS attacks significantly decrease accuracy. The GAN Theft category experiences a dramatic drop to 43.57% accuracy. These results indicate that training the model on specific attack types leads to better zero-day attack detection performance than using the entire dataset.

As shown in Table 8.7, the experimental results demonstrate that the Adaptive SAMKNN outperforms the standard SAMKNN on the NF-ToN IoT and NF-BoT-IoT datasets. On the NF-ToN IoT dataset, the Adaptive SAMKNN achieved a mean accuracy improvement of approximately 1.6 percentage points over the SAMKNN. In contrast, on the NF-BoT-IoT dataset, the improvement was about 0.37 percentage points. These findings show that the proposed algorithm is computationally

feasible for IoT/IIoT environments and has improved performance.

Table 8.8: Comparing performance of our proposed model with other state-of-the-art works

Study	Method	Accuracy	F1
NF-BoT-IoT			
[25]	Extra tree	93.82%	97.00%
[26]	RF	100%	100%
[26]	MLP	99.54%	100%
Our work	Adaptive SAMKNN	98.09%	97.68%
NF-ToN-IoT			
[25]	Extra tree	99.66%	100%
[26]	RF	99.66%	100%
[26]	MLP	94.74%	96.00%
Our work	Adaptive SAMKNN	96.56%	96.47%

8.6 Conclusion

This study proposed an adaptive IDS for computational constraint environments such as IoT and IIoT using an Adaptive version of the SAMKNN algorithm. The results of this work show that the proposed algorithm can effectively detect zero-day attacks by recording high detection accuracy while using low memory. Comparing the results of our proposed approach to traditional offline ML algorithms, our method is more efficient in resource-constrained environments. The dynamic memory allocation of the proposed algorithm, coupled with its ability to process data in real-time, makes it suitable for IoT and IIoT domains.

Although the results of this work are promising, there are areas that can be explored further as future work. For instance, deploying the proposed system in real IoT and IIoT environments while measuring its energy consumption will be an interesting area to explore. Additionally, exploring techniques to enhance the proposed system to detect more sophisticated zero-day attacks will also be another interesting area that can be explored in the future.

References

[1] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey", *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[2] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in industrial internet of things: Architecture, advances and challenges", *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2462–2488, 2020.

[3] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (iiot): An analysis framework", *Computers in industry*, vol. 101, pp. 1–12, 2018.

[4] H. Darabian, A. Dehghantanha, S. Hashemi, S. Homayoun, and K.-K. R. Choo, "An opcode-based technique for polymorphic internet of things malware detection", *Concurrency and Computation: Practice and Experience*, vol. 32, no. 6, e5173, 2020.

[5] Z. Moti, S. Hashemi, and A. Namavar, "Discovering future malware variants by generating new malware samples using generative adversarial network", in *2019 9th International conference on computer and knowledge engineering (ICCKE)*, IEEE, 2019, pp. 319–324.

[6] A. P. Ekong, A. Etuk, S. Inyang, and M. Ekere-obong, "Securing against zero-day attacks: A machine learning approach for classification and organizations' perception of its impact", *Journal of Information Systems and Informatics*, vol. 5, no. 3, pp. 1123–1140, 2023.

[7] S. Oluwadare and Z. ElSayed, "A survey of unsupervised learning algorithms for zero-day attacks in intrusion detection systems.", in *The International FLAIRS Conference Proceedings*, vol. 36, 2023.

[8] G. Guo, "An intrusion detection system for the internet of things using machine learning models", in *2022 3rd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, IEEE, 2022, pp. 332–335.

[9] H. Teymourlouei, D. Stone, and L. Jackson, "Identifying zero-day attacks with machine learning and data reduction methods", in *2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE)*, IEEE, 2023, pp. 2285–2290.

[10] S. I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh, and O. Jogunola, "Federated deep learning for zero-day botnet attack detection in iot-edge devices", *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3930–3944, 2021.

[11] T. Ohtani, R. Yamamoto, and S. Ohzahata, "Idac: Federated learning-based intrusion detection using autonomously extracted anomalies in iot", *Sensors*, vol. 24, no. 10, p. 3218, 2024.

[12] J. Zhang, S. Liang, F. Ye, R. Q. Hu, and Y. Qian, "Towards detection of zero-day botnet attack in iot networks using federated learning", in *ICC 2023-IEEE International Conference on Communications*, IEEE, 2023, pp. 7–12.

[13] A. A. Korba, A. Boualouache, B. Brik, R. Rahal, Y. Ghamri-Doudane, and S. M. Senouci, "Federated learning for zero-day attack detection in 5g and beyond v2x networks", in *ICC 2023-IEEE International Conference on Communications*, IEEE, 2023, pp. 1137–1142.

[14] D. Jin, S. Chen, H. He, X. Jiang, S. Cheng, and J. Yang, "Federated incremental learning based evolvable intrusion detection system for zero-day attacks", *Ieee Network*, vol. 37, no. 1, pp. 125–132, 2023.

[15] Z. He and H. Sayadi, "Image-based zero-day malware detection in iomt devices: A hybrid ai-enabled method", in *2023 24th International Symposium on Quality Electronic Design (ISQED)*, IEEE, 2023, pp. 1–8.

[16] M. Asaduzzaman and M. M. Rahman, "An adversarial approach for intrusion detection using hybrid deep learning model", in *2022 International Conference on Information Technology Research and Innovation (ICITRI)*, IEEE, 2022, pp. 18–23.

[17] K. Saurabh, V. Sharma, U. Singh, R. Khondoker, R. Vyas, and O. Vyas, "Hms-ids: Threat intelligence integration for zero-day exploits and advanced persistent threats in iiot", *Arabian Journal for Science and Engineering*, pp. 1–21, 2024.

[18] S. Guo, T. Sivanthi, P. Sommer, *et al.*, "A zero-day container attack detection based on ensemble machine learning", in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2023, pp. 1–8.

[19] M. Nkongolo and M. Tokmak, "Zero-day threats detection for critical infrastructures", in *Annual Conference of South African Institute of Computer Scientists and Information Technologists*, Springer, 2023, pp. 32–47.

[20] P. Verma, A. Dumka, R. Singh, *et al.*, "A novel intrusion detection approach using machine learning ensemble for iot environments", *Applied Sciences*, vol. 11, no. 21, p. 10 268, 2021.

[21] M. Ellouh, M. Ghaleb, and M. Felemban, "Iotzerojar: Towards a honeypot architecture for detection of zero-day attacks in iot", in *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*, IEEE, 2022, pp. 765–771.

[22] A. Armijos and E. Cuenca, "Zero-day attacks: Review of the methods used based on intrusion detection and prevention systems", in *2023 IEEE Colombian Caribbean Conference (C3)*, IEEE, 2023, pp. 1–6.

[23] V. Losing, B. Hammer, and H. Wersing, "Knn classifier with self adjusting memory for heterogeneous concept drift", in *2016 IEEE 16th international conference on data mining (ICDM)*, IEEE, 2016, pp. 291–300.

[24] J. Montiel, M. Halford, S. M. Mastelini, *et al.*, "River: Machine learning for streaming data in python", *J. Mach. Learn. Res.*, vol. 22, no. 1, Jan. 2021, ISSN: 1532-4435.

[25] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets", *Mobile networks and applications*, pp. 1–14, 2022.

[26] M. Sarhan, S. Layeghy, and M. Portmann, "Evaluating standard feature sets towards increased generalisability and explainability of ml-based network intrusion detection. arxiv 2021", *arXiv preprint arXiv:2104.07183*, 2022.

This page has been intentionally left blank.

Chapter 9

Chapter 9 is published as: P. R. Agbedanu, R. Musabe, J. Rwigema, I. Gatare, Y. Pavlidis, "IPCA-SAMKNN: A novel network IDS for resource-constrained devices," In *2022 2nd International Seminar on Machine Learning, Optimization, and Data Science (ISMODE)* (pp. 540-545). IEEE.

IPCA-SAMKNN: A Novel Network IDS for Resource Constrained Devices

Authors: Promise R. Agbedanu, Richard Musabe, Ignace Gatare, James Rwigema, Yanis Pavlidis

Abstract: Intrusion Detection Systems (IDSs) in traditional computing systems have played a significant role in detecting and preventing cyber-attacks. Unsurprisingly, the same technology is used to detect and prevent cyber-attacks in Internet of Things (IoT) environments. However, due to the computational constraints of IoT devices, traditional computing-based IDS is challenging to deploy on IoT devices. Moreover, IDS for IoT environments should have high classification performance, low complexity models, and small model sizes. Despite numerous advances in IoT-based intrusion detection, developing models that achieve high classification performance while being less complex and smaller in size remains difficult. This study proposes a novel IDS for resource-constrained devices like IoT systems by using a blend of incremental principal component analysis (IPCA) and Self-Adjusting Memory KNN (SAM-KNN) to develop a lightweight machine learning model to detect intrusions in IoT systems. The proposed system was deployed on a Raspberry Pi Model B, representing a resource-constrained device, and evaluated using the UNSW-NB15 dataset. The experimental results show a superior accuracy of 98.91%, a memory overhead of 1.4%, 1.6%, and 2% overhead for CPU and energy, respectively.

9.1 Introduction

The Internet of Things (IoT) has become an essential area of computing, with its application felt in areas such as energy [1], transportation [2], healthcare [3], and

even our homes [4]. The Internet of Things connects things by allowing them to communicate with one another. Like in any computing paradigm, security is always one of the fundamental issues that must be addressed in the IoT ecosystem. When it comes to the security of IoT systems, much progress has been made. Several parameters have been considered to make IoT systems more secure, from architectural design, data confidentiality, and privacy to authentication [5]. However, even with these security mechanisms, attackers still find ways to attack IoT systems. A perfect mitigation strategy is to detect these attacks before they are fully exploited. Intrusion Detection Systems (IDSs) can detect anomalous traffic and trigger an alert before an attacker compromises a system. When it comes to detecting intrusion, traditional networks have made significant progress. The situation is, however, different in IoT systems. IDS for traditional networks cannot work efficiently in IoT environments due to the computational constraints of IoT devices [5]. Several works on IoT-based intrusion detection have been published over the years. However, most of those works focus on the proposed methods' accuracy without considering computational overheads or the ability to detect unknown attacks. Chaabouni et al. [6] identified the ability of IDSs to automatically update attack tables as a potential area for future studies. Zhao et al. [7] focused on how it is challenging to deploy most IDS in IoT environments because of the complex nature of the models used in building these IDS. The authors further argued that an IDS for IoT devices should have good classification performance, low model complexity, and a small model size. In light of the issues mentioned earlier, this research investigates the feasibility of developing a lightweight IDS model for computationally restricted devices, with a significant focus on IoT devices. The main contributions of this paper are as follows:

- Proposes a novel IDS model that is a blend of IPCA and SAM-KNN.
- Used a method called "random oversampling examples" to fix the problem of too many or too few classes in the evaluation dataset.
- Deploying our proposed system on a resource-constrained device while measuring parameters such as memory, CPU, and energy usage.

The rest of the paper is organized as follows: Section II describes other works that are closely related to our study. Our proposed system is presented in Section

III. Section V focuses on the experimental validation of the proposed system. The results are presented in Section V. The study concludes in Section VI.

9.2 Related Literature

In this section, we will look at some works that are closely related to our research. Injadat et al. [8] proposed a network intrusion detection system framework for IoT systems based on a multistage optimized machine learning approach. The objective of the proposed framework is to reduce the computational complexity of the framework without negatively impacting the performance of the IDS. The framework is divided into three stages. The first stage handles the pre-processing of data using techniques such as Z-score normalization and the Synthetic Minority Oversampling Technique. The second stage involves a feature selection process, while the third involves hyperparameter optimization. Even though the performance results of the proposed frameworks show a significant improvement in parameters like accuracy, precision, and recall, the experimental validation does not show how the frameworks affect the computational resources of IoT systems.

In [9], the authors also proposed a framework to mitigate security challenges in IoT environments using software-defined networking, network function virtualization, and machine learning. Although the framework proposed by [9] is not exclusive to intrusion detection, it provides security policies that can be enforced from design through deployment and maintenance in IoT environments.

The concept of drift detection and data adaptation has been one of the biggest challenges in network IDS. An adaptive IDS for IoT systems to detect anomalies in network traffic using an optimized version of LightGBM was proposed by [10] to handle the problem of concept drift. The study used a method called Optimized Adaptive and Sliding Windowing to detect changes in the data stream and adapt to them. Although the proposed method showed promising results, the authors did not show how and at which layer of the IoT architecture the proposed model was going to be deployed.

There is always the challenge of analyzing the large volume of data streams generated in industrial control systems, a situation that motivated [11] to use an unsupervised online anomaly detection approach using an ensemble of trees. In concluding their studies, [11] reiterated that the low space and time complexity

of the proposed system makes it suitable for systems with low computational resources. However, they did not deploy the proposed anomaly detection system in any computationally constrained environment.

Detecting botnet attacks has been one of the areas being explored in IoT environments. Traditional botnet detection techniques, on the other hand, are difficult to implement in IoT systems. This is why [12] proposed an adaptive technique to detect botnet attacks in the IoT ecosystem using Hoeffding adaptive trees and adaptive random forests.

The use of blockchain to solve security problems extends beyond authentication to intrusion detection systems. In [13], the authors propose a blockchain framework that thrives on a deep learning approach to offer a secure distributed IDS that ensures privacy. The results after experimental validation show that the proposed framework outperforms other techniques considered in their study in terms of accuracy and detection rate. However, their study did not show how the proposed framework could work in real IoT networks.

To solve the problems affecting the robustness of machine learning models in detecting network intrusions, [14] proposed a multi-level intrusion detection framework that solves the problems of imbalanced network traffic and the non-identical distribution between training and test sets. The framework is made up of four modules, namely: cluster extraction, pattern discovery, fine-grained classification, and model updating. The experimental results of this framework, after evaluating it on the KDDCUP 99 dataset, show that it outperforms existing models in terms of accuracy, F1 score, and the ability to recognize unknown patterns. However, the authors used a non-IoT-based dataset to evaluate their model. Also, the work does not say how the framework will change the way IoT systems work with their limited computing power.

Edge computing and 5G are the catalysts and the infrastructure on which the IoT ecosystem thrives. However, the resource-constrained nature of IoT devices makes them attractive and susceptible to numerous cyber-attacks. To detect intrusions, [15] proposed a framework for 5G-enabled IoT devices using a federated learning approach. The proposed framework aggregates data and builds customized detection models using federated learning. The proposed approach could improve the detection rate of unknown attacks. Experiments have shown that the proposed framework works better than traditional ways of finding intrusions.

Cassales et al. [16] also proposed an architecture that is based on novel detection techniques using the resources at the edge and in the cloud. This architecture allows traffic to be analyzed at the network's edge, thereby reducing latency.

Our Proposed System

In this section, we present our proposed system, explaining the design and the conceptual implementation. The proposed system is based on a machine-learning approach that ensures the design of an IDS model that is suitable for computationally constrained devices. The proposed system uses IPCA as a feature dimension reduction technique and then uses an incremental learning model to train the proposed IDS. A flowchart of the proposed method is shown in Fig. 9.1

9.2.1 Preprocessing

The first stage is to pre-process the data by analyzing, visualizing, and cleaning it with the aim of making it error-free, a situation that might negatively impact the performance of the system. At this stage, we handled problems such as null values, as well as making sure that the various data types are in a form that can be handled by an ML model. The features are also normalized for the feature selection process.

9.2.2 Over Sampling

The next step is to solve the problem of class imbalance. The proposed system uses the random oversampling balance. The proposed system uses the random oversampling technique, popularly known as Random Oversampler, which was proposed by [17] to handle the problem of class imbalance. We decided to adopt this technique because it is fast at generating new samples. Class imbalance, according to [18], [19], jeopardizes the machine learning process because models end up focusing on the dominant class at the expense of the less prevalent ones.

9.2.3 Feature Selection

We then deploy incremental principal component analysis (IPCA) to reduce the dimensionality of the dataset. Reducing the dimensionality of the data will ensure a reduction in the computational cost of our proposed IDS system. Incremental

principal component analysis is usually used as a substitute for principal component analysis (PCA) when the data being used is too large to be read into memory. Our choice of IPCA over PCA is due to the resource constraints of our proposed system.

9.2.4 Model Training - Self-Adjusting Memory KNN

According to [20], the objective of the Self-Adjusting Memory (SAM) is to handle heterogeneous concept drift among streaming data. SAM-KNN uses short-term memory (STM) to handle current concepts, while long-term memory (LTM) is used for retaining information about past concepts. This model also has a cleaning mechanism that controls the size of the STM while ensuring that the information in the LTM is constantly consistent with that of the STM.

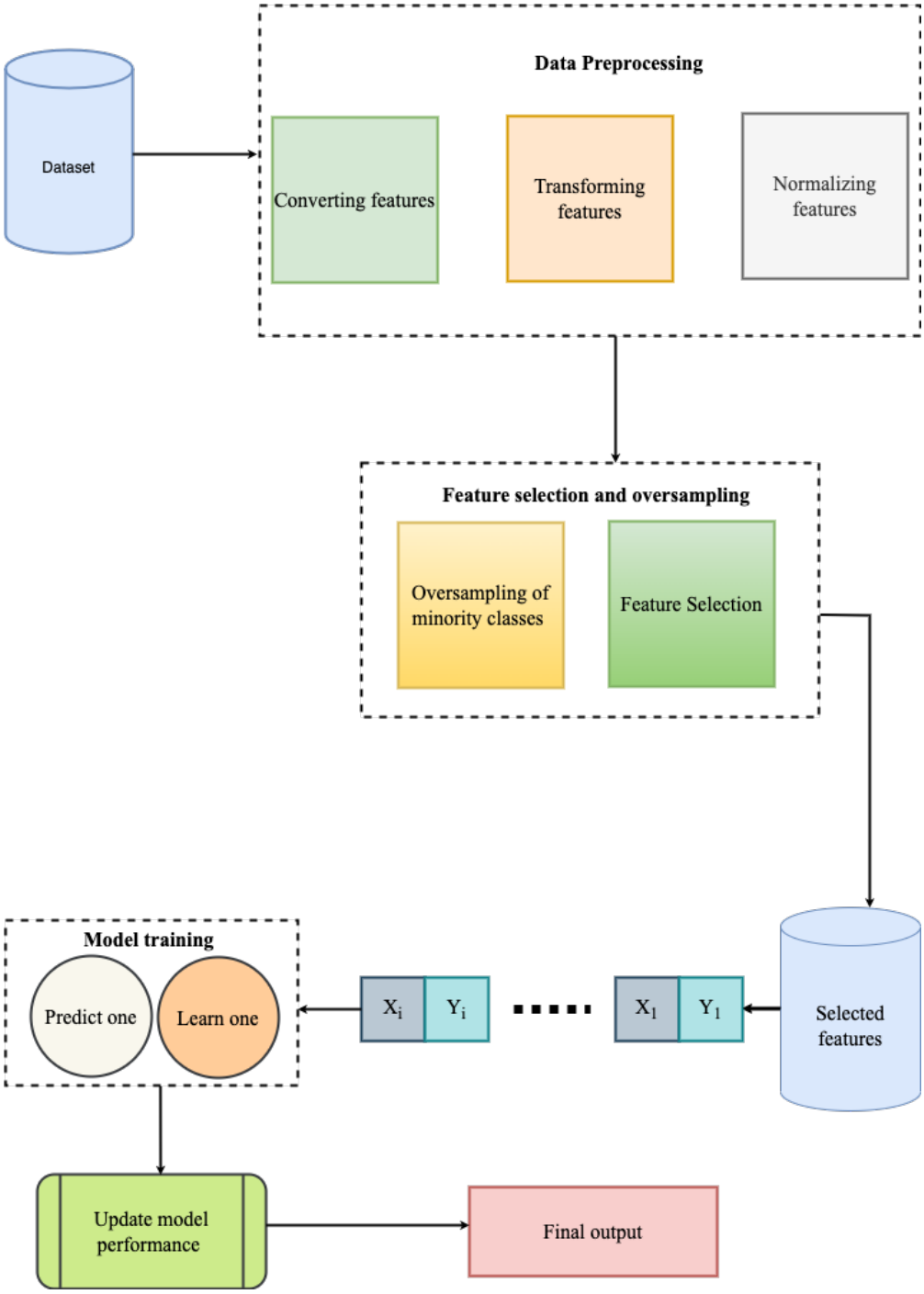


Figure 9.1: Our Proposed IDS based on IPCA-SAMKNN

9.3 Experimental Evaluation

9.3.1 Experimental Environment

This sub-section presents the experimental validation of the model proposed in this study. During the experimental validation, Python 3.8 was used to implement the proposed model while leveraging the Scikit Multiflow framework [21]. In order to prove the efficiency and feasibility of our proposed system, we deployed, trained, and tested it on a Raspberry Pi 2 Model B running on an ARMv7 Quad Core Processor, 1 GB of RAM, a clock speed of 900 MHz, and a 32 GB SD card.

9.3.2 Dataset

We evaluated our proposed system using the UNSW-NB15 dataset. We chose this dataset because it contains contemporary attacks and has a large collection of normal and anomalous traffic. The dataset contains about 100 GB of network packets, which sums up to about 2,540,044 feature vectors categorized into ten different classes, nine of which represent malware activities and one representing normal activities [22]–[25].

9.3.3 Evaluation Metrics

True positive (T_χ): A network traffic that is an attack and has been correctly detected by the model as an attack.

False positive (F_χ): A network traffic that is an attack and has been wrongly detected by the model as normal traffic.

True negative (T_ν): A network traffic that is normal traffic and has been correctly detected by a model as normal traffic.

False negative (F_ν): A network traffic that is normal traffic and has been incorrectly detected by a model as an attack.

9.3.3.1 Accuracy

The accuracy of our proposed system is based on how many attacks and normal network traffic are correctly predicted out of all the network traffic the model has seen.

$$Accuracy = \frac{T_\chi + T_\nu}{T_\chi + T_\nu + F_\chi + F_\nu} \tag{9.1}$$

9.3.3.2 Precision

The precision of our proposed system is calculated by dividing the total predicted network traffic identified by the system by the number of accurately predicted attacks that were detected as attacks.

$$Precision = \frac{T_\chi}{T_\chi + F_\chi} \tag{9.2}$$

9.3.3.3 Recall

The recall of the system is generated by finding the ratio of network traffic that is an attack and has been correctly detected by the model as an attack to the sum of true positives and false negatives.

$$Recall = \frac{T_\chi}{T_\chi + F_\nu} \tag{9.3}$$

9.3.3.4 F1 Score

The proposed IDS's F1 score is the average of precision and recall.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * T_\chi}{2 * T_\chi + F_\chi + F_\nu} \tag{9.4}$$

9.3.3.5 Model Size

Because of the processing limits of IoT devices, it is difficult, if not impossible, to run an IDS designed for regular computers on these IoT devices. This constraint necessitates the development of models that take up as little space as possible.

9.3.3.6 Model Running Time

We measure the overall time it takes the proposed system to learn and predict the state of network traffic.

9.3.3.7 CPU usage

In this evaluation metric, we measure the CPU percentage usage of our proposed system.

9.3.3.8 Energy usage

This criterion for evaluating the system takes into account how much energy our proposed system would use as a percentage of the total energy used by the Raspberry Pi device.

9.4 Results

In this section, we present the experimental validation results of our proposed system and also compare it with other state-of-the-art intrusion detection systems

developed for IoT environments. The original dataset, which is a combination of the training and test sets, contains 164,673 and 93,000 attacks and normal data, respectively. This figure was reduced to 87239 for anomalous data and 29113 for normal data after removing null values. The figure changed to 87239 for both anomalous and normal data after applying the random sampling technique. The parameters of the IPCA used are five components and a batch size of 10. In terms of accuracy, our system achieved an accuracy of 98.91%. Our proposed method also achieved a precision, recall, and F1 score of 98.97%, 99.50%, and 99.23%, respectively. The parameters of the SAM-KNN model used are shown in Table 9.1 while Fig 9.2 shows the performance metrics of the proposed system.

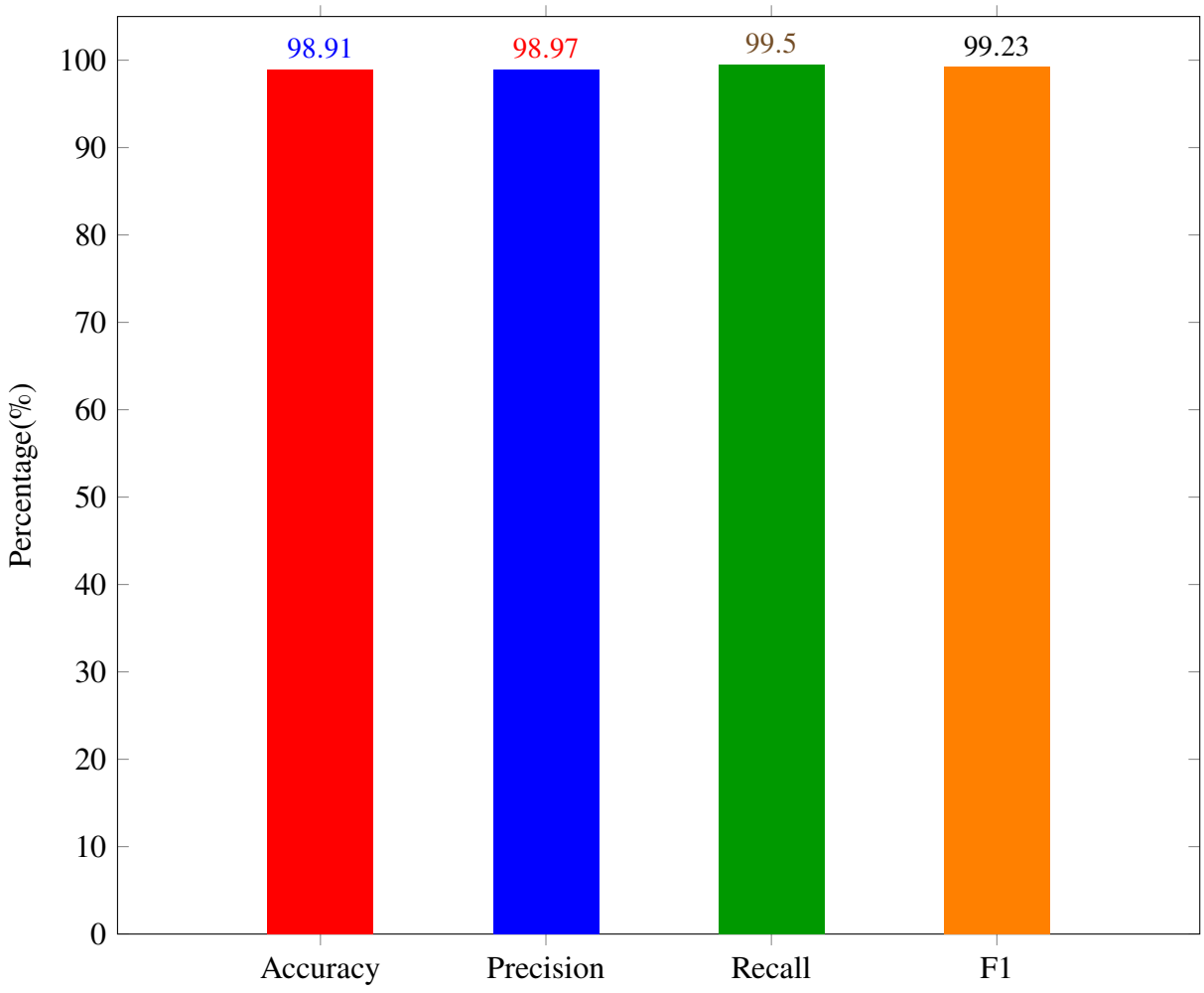


Figure 9.2: Performance metrics of our proposed system

9.4.1 Resource Utilization

Our system was deployed, trained, and tested on this resource-constrained device (Raspberry Pi 2 Model B). During the experimental validation, we measured parameters such as CPU usage, model training time, model size, and energy usage.

Table 9.1: Parameters of the SAM-KNN model

Parameter	Description	Value
n_neighbors	number of nearest neighbors	3
weighting	the category of weighting of the nearest neighbors	uniform
max_window_size	The overall data points	500
stm_size_option	The type of STM adaptation to be used	maxACCApprox
use_ltm	Determines if the use of LTM should be engaged	False

Whereas the model training time and size were measured using metrics in Scikit Multiflow, the CPU and energy usage were measured using PowerJoular, a tool developed by [26] that runs on Linux and can monitor the power consumption of both software and hardware.

The proposed system took up 14.14 MB of space on the hard drive and took 171.41 seconds to process all of the data.

Again, the proposed system used 1.6% of the CPU at peak operation and 1.4% of the memory. It also consumed 2% of the total energy consumed by the Raspberry Pi device. The resource utilization before and after implementing our system is shown in Figure 9.3.

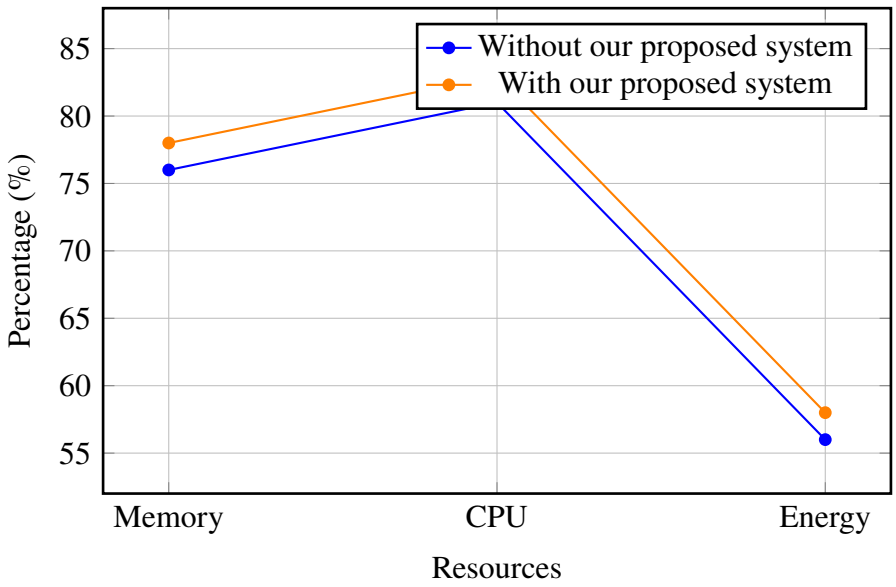


Figure 9.3: Comparison of resource usage with and without the proposed system

9.4.2 Comparison to other studies

In this section, we compare the results of our proposed method to other state-of-the-art methods proposed in other studies.

The problem of detecting minority classes was explored by [27] by combining

a conditional generative adversarial network with a transformer. The proposed approach achieved an accuracy of 89.38% with a precision, recall, and F1 score of 90.12%, 89.38%, and 89.75%, respectively.

The impact of changing the configuration parameters of Spark's ML algorithm was considered by [28]. Evaluating the proposed method with a large dataset (UNSW-NB15) using an optimized decision tree, the results show an accuracy of 98.89%. The proposed method also yielded a precision, recall, and F1 score of 96.58%, 97.10%, and 97.34%, respectively.

Moustapha et al. used a geometric area analysis method based on trapezoid area estimation to detect anomalies in network traffic. To reduce the dimensionality of the data, the authors used PCA. The results showed an accuracy of 91.3% [24].

To combat intrusions in IoT systems, [29] used a combination of deep network beliefs and deep auto-encoders to design an invasive application for IoT systems. The results show an accuracy of 97.86% with an F1 score of 98.17%.

The results show that our proposed system not only shows superiority in terms of accuracy, precision, recall, and F1 score but can also be deployed on devices running on low computational resources.

Table 9.2 shows the results of our proposed system as compared to other state-of-the-art techniques.

Table 9.2: Comparing our proposed system to state-of-the-art techniques on binary classification.

Study	Accuracy	Precision	Recall	F1
[27]	89.38%	90.12%	89.38%	89.75%
[28]	98.89%	96.58%	97.10%	97.34%
[30]	96.30%	NA	NA	NA
[24]	91.30%	NA	NA	NA
[29]	97.86%	NA	NA	98.17%
Our proposed system	98.91%	98.97%	99.50%	99.23%

9.5 Conclusion

The ability of IoT devices to be connected to the internet, especially in this era of 5G technology, not only provides the advantages of automation, access to real-time information, an increase in efficiency, and cost reduction, but also provides disadvantages in the areas of security and privacy. Due to the computational constraints and heterogeneous architectural nature of IoT devices, it is difficult to sufficiently secure them. While using encryption, authentication, access control, and securing the architecture of IoT devices, it is equally imperative to detect attacks

against IoT systems. Even though IDS has come a long way in traditional computing environments, it is almost impossible to use such solutions in IoT environments. This paper proposes a novel IDS for computationally constrained devices like the Internet of Things using a combination of incremental principal component analysis and the SAM-KNN learning technique to build a lightweight IDS that can be trained and deployed on resource-constrained devices. The proposed system was trained and deployed on a Raspberry Pi 2 Model B running on an ARMv7 Quad Core Processor with 1 GB of RAM. The system was evaluated using UNSW-NB15, and the results showed an accuracy of 98.91%. The results also show that the proposed system can be deployed on resource-constrained devices without adding significant computational load, with a model size of 14.14 MB and an overhead of 1.6% for CPU usage, 1.4% of memory size, and 2% of energy. Future work will focus on multi-class classification, testing our model on different datasets, and using different approaches to reduce the computational parameters of our system further.

References

[1] J. Pan, R. Jain, S. Paul, T. Vu, A. Saifullah, and M. Sha, "An internet of things framework for smart energy in buildings: Designs, prototype, and experiments", *IEEE internet of things journal*, vol. 2, no. 6, pp. 527–537, 2015.

[2] P. Saarika, K. Sandhya, and T. Sudha, "Smart transportation system using iot", in *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*, IEEE, 2017, pp. 1104–1107.

[3] M. Islam, A. Rahaman, *et al.*, "Development of smart healthcare monitoring system in iot environment", *SN computer science*, vol. 1, no. 3, pp. 1–11, 2020.

[4] Y. Jie, J. Y. Pei, L. Jun, G. Yun, and X. Wei, "Smart home system based on iot technologies", in *2013 International conference on computational and information sciences*, IEEE, 2013, pp. 1789–1791.

[5] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in internet of things", *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[6] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for iot security based on learning techniques", *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2671–2701, 2019.

[7] R. Zhao, G. Gui, Z. Xue, *et al.*, "A novel intrusion detection method based on lightweight neural network for internet of things", *IEEE Internet of Things Journal*, 2021.

[8] M. Injadat, A. Moubayed, A. B. Nassif, and A. Shami, "Multi-stage optimized machine learning framework for network intrusion detection", *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1803–1816, 2020.

[9] M. Bagaa, T. Taleb, J. B. Bernabe, and A. Skarmeta, "A machine learning security framework for iot systems", *IEEE Access*, vol. 8, pp. 114 066–114 077, 2020.

[10] L. Yang and A. Shami, "A lightweight concept drift detection and adaptation framework for iot data streams", *IEEE Internet of Things Magazine*, vol. 4, no. 2, pp. 96–101, 2021.

[11] L. Liu, M. Hu, C. Kang, and X. Li, "Unsupervised anomaly detection for network data streams in industrial control systems", *Information*, vol. 11, no. 2, p. 105, 2020.

[12] Z. Shao, S. Yuan, and Y. Wang, "Adaptive online learning for iot botnet detection", *Information Sciences*, vol. 574, pp. 84–95, 2021.

[13] O. Alkadi, N. Moustafa, B. Turnbull, and K.-K. R. Choo, "A deep blockchain framework-enabled collaborative intrusion detection for protecting iot and cloud networks", *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9463–9472, 2020.

[14] H. Yao, D. Fu, P. Zhang, M. Li, and Y. Liu, "Msml: A novel multilevel semi-supervised machine learning framework for intrusion detection system", *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1949–1959, 2018.

[15] Y. Fan, Y. Li, M. Zhan, H. Cui, and Y. Zhang, "Iotdefender: A federated transfer learning intrusion detection framework for 5g iot", in *2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE)*, IEEE, 2020, pp. 88–95.

[16] G. W. Cassales, H. Senger, E. R. de Faria, and A. Bifet, "Idsa-iot: An intrusion detection system architecture for iot networks", in *2019 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, 2019, pp. 1–7.

[17] G. Menardi and N. Torelli, "Training and assessing classification rules with imbalanced data", *Data mining and knowledge discovery*, vol. 28, no. 1, pp. 92–122, 2014.

[18] M. Kubat, S. Matwin, *et al.*, "Addressing the curse of imbalanced training sets: One-sided selection", in *Icml*, Nashville, USA, vol. 97, 1997, p. 179.

[19] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study", *Intelligent data analysis*, vol. 6, no. 5, pp. 429–449, 2002.

[20] V. Losing, B. Hammer, and H. Wersing, "Knn classifier with self adjusting memory for heterogeneous concept drift", in *2016 IEEE 16th international conference on data mining (ICDM)*, IEEE, 2016, pp. 291–300.

[21] J. Montiel, J. Read, A. Bifet, and T. Abdessalem, "Scikit-multiflow: A multi-output streaming framework", *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 2915–2914, 2018.

[22] N. Moustafa and J. Slay, "Unsw-nb15: A comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)", in *2015 military communications and information systems conference (MilCIS)*, IEEE, 2015, pp. 1–6.

[23] N. Moustafa, G. Creech, and J. Slay, "Big data analytics for intrusion detection system: Statistical decision-making using finite dirichlet mixture models", in *Data analytics and decision support for cybersecurity*, Springer, 2017, pp. 127–156.

[24] N. Moustafa, J. Slay, and G. Creech, "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks", *IEEE Transactions on Big Data*, vol. 5, no. 4, pp. 481–494, 2017.

[25] A. Mozaffari, M. Vajedi, and N. L. Azad, "A robust safety-oriented autonomous cruise control scheme for electric vehicles based on model predictive control and online sequential extreme learning machine with a hyper-level fault tolerance-based supervisor", *Neurocomputing*, vol. 151, pp. 845–856, 2015.

[26] A. Noureddine, "Powerjoular and joularjx: Multi-platform software power monitoring tools", in *18th International Conference on Intelligent Environments*, 2022.

[27] Y. Yang, C. Yao, J. Yang, and K. Yin, "A network security situation element extraction method based on conditional generative adversarial network and transformer", *IEEE Access*, vol. 10, pp. 107 416–107 430, 2022.

[28] S. Bagui, M. Walauskis, R. DeRush, H. Praviset, and S. Boucugnani, "Spark configurations to optimize decision tree classification on unsw-nb15", *Big Data and Cognitive Computing*, vol. 6, no. 2, p. 38, 2022.

[29] G. Altan, "Securedeepnet-iot: A deep learning application for invasion detection in industrial internet of things sensing systems", *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 4, e4228, 2021.

[30] M. Zeeshan, Q. Riaz, M. A. Bilal, *et al.*, "Protocol-based deep intrusion detection for dos and ddos attacks using unsw-nb15 and bot-iot data-sets", *IEEE Access*, vol. 10, pp. 2269–2283, 2021.

This page has been intentionally left blank.

Chapter 10

Chapter 10 is published as:

P. R. Agbedanu, R. Musabe, J. Rwigema, I. Gatare, Using Incremental Ensemble Learning Techniques to Design Portable Intrusion Detection for Computationally Constrained Systems. *International Journal of Advanced Computer Science and Applications,* vol. 13, no. 11, pp. 33-45, 2022.

Using Incremental Ensemble Learning Techniques to Design Portable Intrusion Detection for Computationally Constrained Systems

Authors: Promise R. Agbedanu, Richard Musabe, Ignace Gatare, James Rwigema

Abstract: Computers have evolved over the years, and as the evolution continues, we have been ushered into an era where high-speed internet has made it possible for devices in our homes, hospitals, energy, and industry to communicate with each other. This era is known as the Internet of Things (IoT). IoT has several benefits in a country's economy, including health, energy, transportation, and agriculture sectors. These enormous benefits, coupled with the computational constraints of IoT devices, make it challenging to deploy enhanced security protocols on them, making IoT devices a target of cyber-attacks. One approach that has been used in traditional computing over the years to fight cyber-attacks is the Intrusion Detection System (IDS). However, it is difficult to deploy IDS meant for traditional computers in IoT environments because of the computational constraints of these devices. This study proposes a lightweight IDS for IoT devices using an incremental ensemble learning technique. We used Gaussian Naive Bayes and Hoeffding trees to build our incremental ensemble model. The model was then evaluated on the TON IoT dataset. Our proposed model was compared with other state-of-the-art methods proposed and evaluated using the same dataset. The experimental results show that the proposed model achieved an average accuracy of 99.98%. We also evaluated the memory consumption of our model, which showed that our model achieved a lightweight model status of 650.11 KB as the highest memory

consumption and 122.38 KB as the lowest memory consumption.

10.1 Introduction

As the evolution of computing technology continues, the ability of things such as fridges, air-conditioners, medical equipment, and meters, among others, to communicate has become a reality due to fast communication technologies. A paradigm popularly known as the Internet of Things (IoT) has not only become a household term with smart homes, but it also has numerous uses in energy, agriculture, manufacturing, healthcare, and transportation. There is no doubt that the IoT has many benefits, which is why the number of IoT devices is growing exponentially. The number of IoT devices is estimated to reach 30.9 billion by 2025, according to [1]. The numerous benefits of the IoT ecosystem make it attractive to cyber-attacks. An attack statistic presented by SAM Seamless Network shows that over 1 billion IoT-based attacks happened in 2021 [2]. Although methodologies such as encryption and secure architecture are progressively being deployed to ensure that IoT devices are secured, the computational constraint of these devices makes it challenging to implement these security measures to their fullest potential. Another approach to securing these devices from cyber-attacks is to detect these attacks before an attacker exploits them. Intrusion Detection Systems have been around for more than four decades, with the development of these IDSs focused on traditional computing systems [3]. They have been among the primary methodologies used to protect computer networks. Vacca [4] defines intrusion detection as the process of detecting activities perpetrated against computer systems by intruders. Over the past forty years, many breakthroughs have been made in intrusion detection. One of the most significant breakthroughs in this area is the use of machine learning in detecting intrusions. However, with all these breakthroughs, it is challenging to deploy traditional computing-based IDS methods in the Internet of Things. The impossibility of deploying these IDSs in IoT systems has arisen because of the computational constraints of IoT devices. The constraints have led to several studies being carried out to design IDSs that can be deployed in IoT systems without significantly affecting the computational resources of these devices. Several approaches have been proposed in designing lightweight IDSs for IoT environments. However, these studies neither report

how these lightweight IDSs are achieved nor specify how much computational resources these proposed approaches consume. For example, [5]–[9] proposed various techniques that are supposed to translate into lightweight IDSs. However, these works either failed to report how these methods translate into lightweight IDS or how much computational resources these proposed methods consume. This study proposes a novel lightweight intrusion detection system using an incremental machine learning approach. The main contributions of this study are as follows

- Using an incremental machine learning approach to design a lightweight IoT intrusion detection system.
- Measuring the memory consumption of our proposed model.
- The study uses an incremental ensemble approach to achieve improved accuracy.
- The study evaluates the proposed IDS model on an IoT dataset.

The remainder of the paper is structured as follows. Section II discusses the study's background. Section III focuses on works relevant to our study, whereas Sections IV, V, and VI focus on the proposed model, experimental evaluation, and conclusion.

10.2 Background

10.2.1 Intrusion Detection System

An intrusion detection system (IDS) is a security device that detects illegal access to data within a networked or computer-based environment in order to threaten the integrity, availability, or confidentiality of the computing device [10], [11]. An IDS's objective is to continuously monitor network traffic and flag any activity that violates the normal usage of the system [12]. According to [13], typically, an IDS consists of sensors, an analysis engine, and some reporting system. Intrusion detection systems can be classified either by how they are deployed or by how they detect illegal activities. From a deployment perspective, an IDS can be classified as distributed, centralized, or hybrid. On the other hand, an IDS can be classified as signature-based, anomaly-based, specification-based, or hybrid. According to [14], signature-based detection is the set of pre-defined rules, such as the sequence

of bytes in network traffic, that are pre-loaded to trigger an alert when a matched sequence is detected. On the other hand, anomaly detection records the normal behavior of a network and then compares it with the system's current behavior. The authors also explained the specification-based detection method as an approach that uses input specifications designed manually. Finally, hybrid detection methods deploy a combination of signature-based, anomaly-based, and specification-based detection methods to improve accuracy and reduce false positive rates.

10.2.1.1 Ensemble Learning

According to [15], ensemble learning is a machine learning technique that combines the strengths of different machine learning algorithms into a single algorithm. The primary goal of ensemble learning is to improve accuracy by leveraging the strengths of the ensemble learners [16]. There are instances where traditional machine learning models do not achieve high accuracy [17]. Several ensemble-based techniques have been developed over time, but the most popular are bagging, boosting, stacking generalization, and expert mixture [16].

In the preceding paragraphs, we briefly explained the three categories of ensemble learning.

10.2.1.2 Bagging-based Learning

Bagging, short for bootstrap aggregation, is an algorithm that is best suited for problems with a small training dataset. Given a training set S with a cardinality n, the bagging algorithm trains several independent classifiers T. Each of these classifiers is trained using a percentage of N [16] sampling. Linear classifiers such as linear SVM, decision stumps, and single-layer perceptrons are excellent candidates for bagging [16]. Classifiers are trained and then combined using simple majority voting in bagging. Bagging, an abbreviation for bootstrap aggregation, is a method that works well with issues that have a limited training dataset. The bagging algorithm learns several independent classifiers T given a training set S with a cardinality of n. Each of these classifiers is trained using a proportion of N sampling. Linear classifiers like linear SVM, decision stumps, and single-layer perceptrons are great candidates for bagging [16]. In bagging, classifiers are trained and then concatenated using simple majority voting.

10.2.1.3 Boosting-based Learning

An iterative approach can be used to generate a robust classifier from a set of weak classifiers. Although boosting also combines a large number of weak learners through simple majority voting, there is one significant difference between boosting and bagging. Every instance in bagging has an equal chance of being in each dataset used in training. In boosting, on the other hand, the dataset used to train each subsequent model focuses on instances misclassified by the previous model. At any given time, a boosting designed for a binary class problem generates a set of three weak classifiers. The first learning classifier is trained on a random subset of the training data available. A different subset of the original training dataset is used to train the second learning classifier [18].

10.2.1.4 Stack Generalization

Non-trainable combiners are used in bagging and boosting methods. The combination weights in non-trainable combiners are determined after the classifiers have been trained. The combination rule used in non-trainable combiners does not allow determining which member classifier learned from which partition of the feature space [16]. Trainable combiners can be used to solve this problem, and individual ensemble members can be combined using a separate classifier in stacked generalization.

10.2.1.5 Mixture of Experts

A sampling technique is used to train an ensemble of classifiers in a mixture of experts. The classifiers are then combined using a weighted combination rule [19]. Furthermore, a mixture of experts can encompass the selection of algorithms, with each classifier trained to become an expert in a different aspect of the feature space. Individual classifiers are usually not weak since they are trained to become experts.

10.2.2 Online/Incremental Machine Learning

Online machine learning, also known as incremental machine learning, is increasingly becoming popular in real-time. According to [20], as reported by [21], in data stream models, an infinite stream of data arrives continuously, and these streams of data are to be processed by systems with resource constraints. earlier because the data will no longer be available for the algorithm. The concept of machine

learning models acquiring knowledge from continuous data without accessing the original data has been applied to domains like intelligent robots, auto-driving, and unmanned aerial vehicles [22]–[24]. According to [20], as reported by [21], in data stream models infinite stream of data arrives continuously, and these streams of data are to be processed by systems with resource constraints. The main restriction of data stream models is that the memory of these systems is usually small and can only hold a minimal portion of the data stream. Regarding data stream models, only a minimal subset of the data can be kept for instant data analysis [25]. Fig. 10.1 shows an online machine-learning model using an offline dataset, while Fig. 10.2 shows the same model using streams of network traffic.

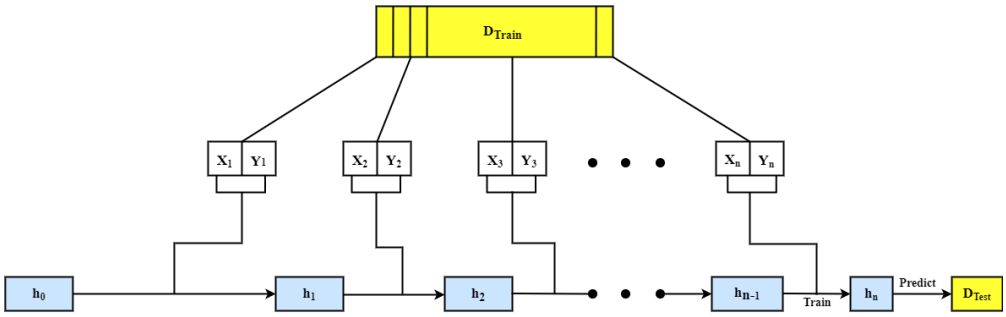


Figure 10.1: Online machine learning using an Offline dataset.

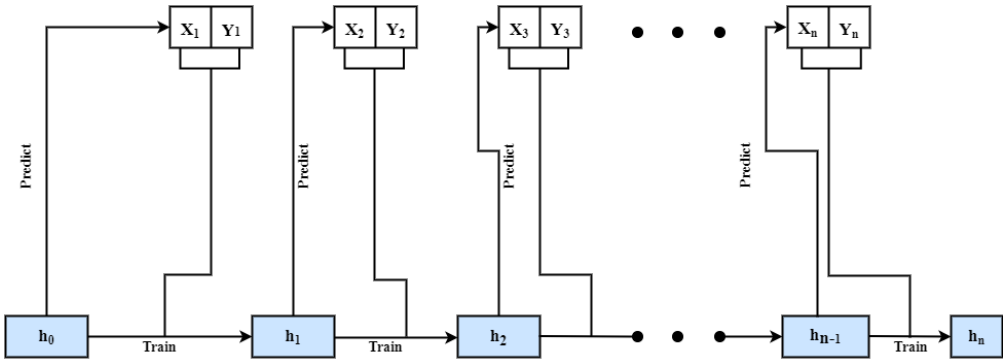


Figure 10.2: Online machine learning using data stream.

10.3 Related Work

Throughout this section, we will look at some studies that are relevant to our work. Yang et al. [26] proposed an ensemble framework for intrusion detection systems (IDSs) in IoT environments, focusing primarily on idea drift adaptability. The suggested framework employs a performance-weighted probability averaging

ensemble to manage concept drift in IoT anomaly detection. When compared to other cutting-edge approaches, the suggested framework performed better. Even though the author's proposed method took less time to run than the other methods they looked at for their study, they did not look into how the proposed model affected other computing parameters, such as memory.

Jan et al. [5] used a supervised Support Vector Machine (SVM) to detect IoT adversarial attacks. The authors used only the sensor node's packet arrival rate to design the proposed IDS. The accuracy of the proposed IDS showed better performance compared to other models like neural networks, KNN, and decision trees. One of the drawbacks of this study is that it only considered DDoS attacks. Additionally, the authors failed to report how the proposed approach leads to a lightweight IDS. Parameters such as memory consumption and model running time were not reported.

In a similar study, [27] proposed a lightweight IDS for IoT ecosystems using a Deep Belief Network and Genetic Algorithm. According to the study, the proposed system was more accurate than other methods that were looked at for the study. However, the study failed to report how the proposed approach translates to a lightweight model. Moreover, the dataset used for the experimental validation is not an IoT-based dataset. Like in other studies, parameters that are supposed to prove the lightweight status of the proposed method were not considered in the study.

Roy et al. [7] also designed a lightweight IDS for IoT systems using a set of optimization techniques. They used multicollinearity, sampling, and dimensionality reduction to reduce the training data, which resulted in a shorter training time. Like other earlier related works considered in this section, although their proposed approach reduces the training time of the model, the study did not report how much memory the model consumes.

Zhao et al. [8] suggested a network intrusion detection method for IoT devices utilizing a lightweight neural network. To minimize the dimensionality of features, the authors employed a principal component analysis approach. The proposed method was tested using the UNSW-NB15 and Bot-IoT datasets. Although the authors determined that both the ultralight feature extraction network and principal component analysis contributed to the suggested model's lightweight performance, they did not report on the computational complexity of their proposed method.

In order to make a lightweight IDS for IoT systems, [9] said that they used a mix of feature selection techniques on different datasets to make a lightweight IDS algorithm for IoT traffic. However, the two datasets used to evaluate their proposed lightweight IDS were non-IoT related. Also, the authors did not talk about how their proposed model would affect the computing power in their experimental environment.

Latif et al. [6] reported using a Dense Random Neural Network to develop a lightweight intrusion detection system for IoT environments. The proposed model was evaluated on the ToN-IoT dataset, and the results show a detection accuracy of 99.14% for binary class classification and 99.05% for multiclass classification. However, Latif et al. did not report on the computational complexity of their proposed model. A parameter is required to measure the lightness of the proposed model.

Pan et al. [28] also suggested a lightweight, intelligent intrusion detection system (IDS) architecture for wireless sensor networks. The authors used KNN and the sine-cosine technique to create their model. The authors reported that combining the above techniques improves classification accuracy and reduces false alarms. However, the authors failed to report how the lightweight model was achieved or what parameters were used to determine the lightweight status of the model.

Reis et al. [29] created an IDS for cyber-physical systems using incremental support vector machines. In their study, a one-class support vector machine was applied to each sensor to retrieve abnormal behaviors. These anomalies are orchestrated as an output of the proposed incremental machine learning model. Although the model proposed by Reis et al. achieved an accuracy higher than 95%, the study did not detail how the proposed method would affect the computational resources of cyber-physical systems.

To reduce the computation overhead, [30] introduced a privacy-preserving pipeline-based intrusion detection for distributed incremental learning that selects unique features using an innovative extraction technique. Current incremental learning techniques are computationally expensive, and the distributed intrusion detection method is used to distribute the load across IoT and edge devices. Theoretical analysis and experiments show that state-of-the-art techniques require less space and time. The study, however, reported on time complexity but not space

complexity. Furthermore, the experimental validation dataset is not an IoT-based dataset.

10.4 Proposed Model

This section details the design and conceptual implementation of our suggested approach. The suggested model is based on a machine learning technique, ensuring the creation of a lightweight IDS model suitable for the Internet of Things environment. The proposed model employs incremental machine learning and data streaming ensemble learning approaches to create a lightweight intrusion detection system for the IoT environment. The proposed model processes network data generated in IoT environments as data streams and trains each data stream. After each iteration, the model is updated. Fig. 10.3 depicts our proposed model.

1. Pre-processing: The dataset used to train our proposed model is cleaned at this stage. The pre-processing data approach used in this study includes imputing missing data values and transforming and selecting important features to train our machine learning model. We employed one-hot encoding as one of the techniques to pre-process our data. A single hot encoding transformer will encode all the features provided to it. If a list or set is supplied, this transformer will encode each item in the list or set by composing it with the compose command in River; the encoding can apply to a subset of features.

2. Model training: This study proposes a novel online stacking ensemble machine learning technique using Gaussian Naive Bayes and Hoeffding Tree Classifier. We chose these two machine learning models to build our ensemble learning because we want to achieve the following three objectives

   - Design a model that consumes a minimal computational resource (lightweight)
   - Building a fast model
   - A model that achieves a high accuracy

   Gaussian Naive Bayes and Hoeffding Tree Classifier are used as the base classifiers of our proposed model, while Hoeffding Tree is used as the meta classifier of the proposed model. Each observation of the dataset is read as a stream and is then used to train the base and meta classifiers. Each base

classifier predicts each stream of data, that is, X_iY_i, which becomes feature input to the meta classifier. The meta classifier (HT) then uses the outputs of the base classifiers to make a better prediction. We chose Hoeffding trees because they learn patterns in data without continuously storing data samples for future reprocessing, and this makes them particularly suitable for use on embedded devices. Similarly, Gaussian NB is quick and flexible and produces highly reliable results. It works well with large amounts of data and requires little training time, and it also improves grading performance by removing insignificant specifications [31], [32].

3. Model evaluation: The final stage of the model is the model evaluation stage. The proposed model's accuracy, precision, recall, F1, model training time, and memory consumption are all evaluated.

We chose incremental and ensemble machine learning to develop our framework in this work because of the following benefits.

1. Network traffic is generated in blocks as a data stream. By using incremental learning on network traffic, models can predict the nature of traffic without having to be trained on large datasets.

2. The computational constraints of IoT devices make loading an entire training dataset into main memory difficult and impractical. Even if the entire training data can fit into the main memory of an IoT device, the device's computational power will be drastically reduced.

3. New data is constantly available because of the sophisticated nature of cyber-attacks. Retraining the model on the entire dataset will be time-consuming and computationally expensive. Because models in online machine learning are trained with data streams, they can quickly learn from new data examples without consuming much computational power.

4. Real-time network traffic is generated, which must be analyzed in real-time to prevent intruders from gaining unauthorized access to devices. Online machine learning has proven to be an effective learning method in real-time environments.

5. Traffic flow is dynamic and constantly changing. Changes in network traffic can impact the predictive performance of machine learning models, referred

to as concept drift in machine learning. Models should be able to self-adapt to changes in the relationship between input and output data to handle concept drifts.

6. Most of the time, ensemble methods have produced higher accuracy than the individual models that were used to make them.

Additionally, the use of the above method poses the following limitation.

Getting a good tradeoff between accuracy, speed, and minimal resource consumption is going to be a challenge because combining models increases the computational consumption of the final output.

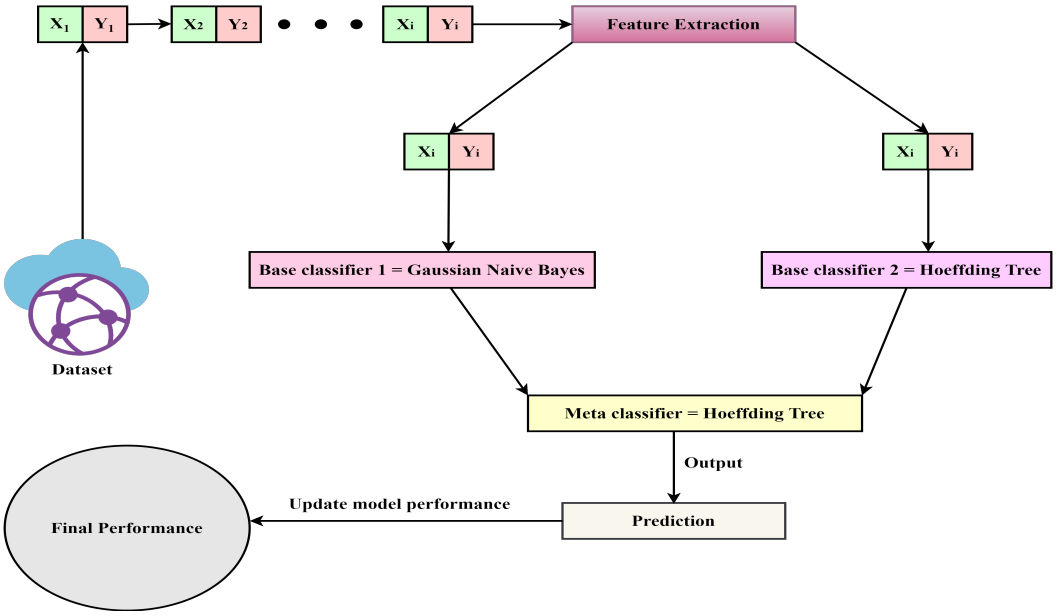


Figure 10.3: Our proposed model

10.4.1 Gaussian Naive Bayes

According to [33], the Naive Bayes algorithm is a typical illustration of how generative hypotheses and parameter guesses can facilitate learning. Consider the problem of predicting a label y ∈ {0,1} from a vector of characteristics **X** = (x_1,..., x_d), where each x_i is in the range of {0,1}. The optimal classifier of Bayes is given below

$$h_{Bayes}(\mathbf{X}) = argmaxP[Y = y|X = x], y \in \{0, 1\} \tag{10.1}$$

We need 2^d parameters to define the probability function P[Y = y|X = x], each of which relates to P[Y = 1|X = x] for a given value of y ∈ $\{0, 1\}^d$. This means that when the number of features increases, so does the number of instances necessary.

In the Naive Bayes technique, we make the generative assumption that, given the label, the features are independent of one another. To put it another way,

$$P[Y = y|X = x] = \prod_{i=1}^{d} P[Y = y|X = x] \tag{10.2}$$

The Bayes optimum classifier can be reduced further using this assumption and the Bayes rule:

$$h_{Bayes}(\mathbf{X}) = argmaxP[Y = y] \prod_{i=1}^{d} P[X_i = x_i|Y = y] \tag{10.3}$$

The formulation in Equations (10.2)–(10.3) reduces the problem to two d-dimensional vectors and one scalar (2d+1). In this situation, the generative assumption we made considerably decreased the number of parameters we needed to learn. When the maximum likelihood principle is used to determine the parameters, the resulting classification model is called the Naive Bayes classifier.

One typical technique to handle continuous attributes in Naive Bayes classification is to use Gaussian distributions to express the probabilities of the features based on the classes. As a result, every attribute is represented as Xi N(, 2) by a Gaussian probability density function (PDF), [34] as reported by [35].

$$X_i \sim N(\mu, \sigma^2) \tag{10.4}$$

The Gaussian PDF is shaped like a bell and is defined by the equation below, where μ is the mean and σ^2 is the variance.

$$N(\mu, \sigma^2)(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{10.5}$$

10.4.2 Hoeffding Tree (HT)

Hulten et al [36] are the first to propose Hoeffding trees. The Hoeffding tree algorithm is a fundamental algorithm for stream data classification. It is an induction of a decision tree algorithm that could learn from enormous data streams if the distributed generating examples remain constant over time. It creates decision trees that are similar to the standard batch learning method. Asymptotically, Hoeffding trees as well as decision trees are connected. The HT technique is based on the

basic premise that a modest sample size can frequently be sufficient to identify an optimal splitting feature. The key point to understand here is that classic batch learning algorithms produce decision trees based on attribute splitting. The HT method is mathematically verified to use the Hoeffding bound. To comprehend the significance of the Hoeffding bound, a few assumptions must be made. Let's say we get N separate samples of a random variable r with a range of R, where r is a measure of attribute selection. In the case of Hoeffding trees, r is information gain, and if we calculate the mean value of r_{mean} for this sample, the Hoeffding limit indicates that the true mean of r is at least 1-δ. The primary benefits of the HT algorithm are as follows:

1. it is incremental in nature
2. it achieves high accuracy with a small sample size.
3. scans on the same data are never performed.

However, the Hoeffding Tree has a few disadvantages. The main disadvantage is that HT cannot handle concept drift because the node cannot be changed once it is created. Wang et al [37] described how to deal with concept drift using classifiers. The algorithm devotes a significant amount of time to attributes with nearly identical splitting quality. Furthermore, memory utilization can be further optimized.

$$\in= \sqrt{\frac{R^2 1 n \frac{1}{\delta}}{2n}} \tag{10.6}$$

The algorithm for the Hoeffding tree is shown in Algorithm **??**.

Algorithm 6 Hoeffding Tree Algorithm [36]

Input:
  S: a sequence of examples
  X: a set of discrete attributes
  $G(\cdot)$: split evaluation function
  δ: one minus the desired probability of choosing the correct attribute
Output: HT: the decision tree
Let HT be a tree with a single leaf l_1 (the root)
Let $X_1 \leftarrow X \cup \{X_\theta\}$
Let $\bar{G}_1(X_\theta)$ be the $\bar{G}$ obtained by predicting the most frequent class in S
for each class Y_k **do**
  for each value X_{ij} of each attribute $X_i \in X$ **do**
    Let $n_{ijk}(l_1) \leftarrow 0$
  end for
end for
for each example $(\mathbf{x}, Y_k)$ in S **do**
  Sort $(\mathbf{x}, Y_k)$ into a leaf l using HT
  for each X_{ij} in $\mathbf{x}$ such that $X_i \in X$ **do**
    Increment $n_{ijk}(l)$
  end for
end for
Label l with the majority class among the examples seen so far at l
if examples at l are not all of the same class **then**
  for each attribute $X_i \in X_l \setminus \{X_\theta\}$ **do**
    Compute $\bar{G}_l(X_i)$ using counts $n_{ijk}(l)$
  end for
  Let X_a be the attribute with highest $\bar{G}_l$
  Let X_b be the attribute with second-highest $\bar{G}_l$
  Compute ϵ using Hoeffding bound (Equation 1)
  if $\bar{G}_l(X_a) - \bar{G}_l(X_b) > \epsilon$ and $X_a \neq X_\theta$ **then**
    Replace l with internal node splitting on X_a
    for each branch of the split **do**
      Add a new leaf l_m and let $X_m \leftarrow X \setminus \{X_a\}$
      Let $G_m(X_\theta)$ be G from most frequent class at l
      for each class Y_k and value X_{ij} of attribute $X_i \in X_m \setminus \{X_\theta\}$ **do**
        Let $n_{ijk}(l_m) \leftarrow 0$
      end for
    end for
  end if
end if
return HT

10.5 Experimental Evaluation

10.5.1 Experimental Environment

The proposed method was implemented using Python 3.8 with River as our framework for online machine learning. The proposed method was implemented on a MacBook Pro with an M1 chip with 16 GB of RAM. The TON-IoT dataset was used to evaluate the proposed framework. There are several incremental or online streaming libraries that provide machine functionalities. Some of these libraries are Creme, Scikit-multiflow, and River. In this study, we choose to build our incremental learning models using River. According to [38], River is a merger of Creme and Scikit-multiflow. River is a library that allows continual learning by handling dynamic data streams. We chose River because it includes data transformation methods, learning algorithms, and optimization algorithms. Its distinct data structure lends itself well to streaming data and web application settings.

10.5.2 Dataset

According to [39], the TON-IoT dataset was built by the Cyber Range and IoT Lab at the University of South Wales. The dataset has nine (9) types of cyber-attacks. These are Denial of Service (DoS), Distributed Denial of Service (DDoS), ransomware, backdoor, data injection, scanning, Cross-site Scripting (XSS), password cracking, and Man-in-the-Middle (MiTM). The generated data were from seven IoT and IIoT devices: fridge, motion light, garage door, GPS tracker, thermostat, and weather. The fridge dataset has a total of 587076 records; the motion light dataset has 452262 records; the garage door has 591446 records; the GPS tracker produced 595686 records, while the thermostat and weather produced 442228 and 650242 records, respectively. The statistics of the dataset used are shown in Tables 10.1 and 10.2 below.

10.5.3 Evaluation Metrics

True positive (V_P): Positive intrusion that is both expected and confirmed.

False positive (U_P): An intrusion that was expected to be positive but ended up turning out to be negative.

True negative (V_N): The intrusion is expected to be negative and confirmed to be negative.

Table 10.1: Statistics of TON IoT dataset [39]

Fridge IoT dataset	
Type of attack	**No of rows**
Backdoor	35568
DDoS	10233
Injection	7079
Normal	500827
Password	28425
Ransomware	2902
XSS	2042
GPS tracker IoT dataset	
Backdoor	35571
DDoS	10226
Injection	6904
Normal	513849
Password	513849
Ransomware	2833
Scanning	550
XSS	577
Motion light IoT dataset	
Backdoor	28209
DDoS	8121
Injection	5595
Normal	388328
Password	17521
Ransomware	2264
Scanning	1775
XSS	449
Weather IoT dataset	
Backdoor	35641
DDoS	15182
Injection	9726
Normal	559718
Password	25715
Ransomware	2865
Scanning	529
XSS	866
Garage IoT dataset	
Backdoor	35568
DDoS	10230
Injection	6331
Normal	515443
Password	19287
Ransomware	2902
Scanning	529
XSS	1156

False negative (U_N): The intrusion was expected to be negative, but it turned out to be positive.

Table 10.2: Statistics of TON IoT dataset continuation [39]

Modus IoT dataset	
Backdoor	40035
Injection	7079
Normal	405904
Password	24269
Scanning	529
XSS	577
Thermostat IoT dataset	
Backdoor	35568
DDoS	10230
Injection	6331
Normal	515443
Password	19287
Ransomware	2902
Scanning	529
XSS	1156

10.5.3.1 Accuracy

A model's overall accuracy can be measured by the number of correctly predicted events made by the given model. The formula below computes the total accuracy of the model.

$$Accuracy = \frac{V_P + V_N}{V_P + V_N + U_P + U_N}$$

10.5.3.2 Precision

Precision is found by dividing the total number of positive detections by the number of positive detections that were correctly identified as positive.

$$Precision = \frac{V_P}{V_P + U_P}$$

10.5.3.3 Recall

The recall is defined as the ratio of true positive detections to the number of real abnormal samples.

$$Recall = \frac{V_P}{V_P + U_N}$$

10.5.3.4 F1 Score

The F1 score is the average of precision and recall. The F1-score is determined as the weighted average of precision and recall, taking both the U_P and U_N into consideration.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * V_P}{2 * V_P + U_P + U_N}$$

10.5.3.5 Memory

The computational constraint of IoT devices makes it difficult and sometimes impossible to run an IDS meant for traditional computers on these IoT devices. This calls for developing models that consume minimal memory (lightweight models).

10.5.3.6 Model Running Time

In this evaluation metric, we measure the total time it takes for the proposed model to run.

10.5.4 Results

In this section, we present the proposed results and compare them with other state-of-the-art techniques. To begin with, this study compares the results of the proposed model with other state-of-the-art IDS models proposed and evaluated with the TON IoT dataset. We decided to limit the state-of-the-art studies that used the ToN IoT dataset because we wanted to eliminate biases. In comparing these studies, we considered the category of the TON IoT dataset used in each study, the method proposed by each of the works under consideration, the highest accuracy recorded, and whether the study records the time used to build the model, as well as the amount of memory the model consumes. The comparison of our approach with other state-of-the-art IDS for IoT systems is presented in Table 10.3. Where the authors did not report a parameter, we indicate it as not available (N/A). Table 10.3 shows that out of the five state-of-the-art IDSs considered in the study, none of them reports on the model or the memory consumption of their proposed technique. Although [40], [41] both report 100% accuracy, our proposed model outperforms the methods proposed in those studies for the following reasons:

1. The accuracy reported in our study is the average accuracy of our proposed model, whereas [40], [41] report total accuracy.

2. Our study focused on multi-class classification, whereas [40], [41] focused on binary-class classification.

Table 10.4 compares the accuracy, time, and memory consumption of the models used to build our incremental ensemble technique with our proposed model. Although the time and memory consumption of the individual models are lower than our proposed model, we wanted to propose a model that achieves a trade-off between accuracy, time, and memory consumption. Our proposed model achieved a higher accuracy without significantly increasing the time and memory consumption. The time and memory consumption of the proposed model show that it can run on computationally constrained devices without negatively impacting the computational resources of these devices.

Table 10.3: Using the ToN IoT dataset, we compared our proposed model to state-of-the-art models that had been tested using the same dataset

Study	Year of the study	Method used	Highest accuracy	Model training time (S)	Memory consumption (KB)
[6]	2021	Dense Random Neural Network	99.14%	N/A	N/A
[40]	2022	Optimized decision tree	100	N/A	N/A
[41]	2022	Ensemble based voting	100	N/A	N/A
[42]	2022	Graph Neural Network	97.87	N/A	N/A
[43]	2021	Synthetic minority oversampling technique	99.0	N/A	N/A
Our proposed model	2022	Stack-based Incremental ensemble (HT and Gaussian NB)	99.98	71	122.38

Fig. 10.4 below shows the output of our proposed model in terms of memory usage. The results show that our proposed model recorded the highest memory usage as well as accuracy. The results show a good trade-off between detection accuracy and memory usage. Although the highest memory recorded is 650 KB, this model can efficiently run on most edge devices used as IoT gateways without negatively impacting the computational resources of such devices.

When tested on the modus dataset, our proposed model achieved a superior average accuracy of 96.81%, with precision, recall, and F1 scores of 97.23%, 96.81%, and 96.92%, respectively. The Hoeffding tree had an average accuracy

Table 10.4: Comparing the accuracy and memory usage of the base classifiers against our model on the different datasets

Fridge IoT dataset					
Gaussian NB		**Hoeffding Tree**		**Our proposed model**	
Accuracy	**Memory (KB)**	**Accuracy**	**Memory (KB)**	**Accuracy**	**Memory (KB)**
85.31%	15.85	98.68%	532.71	99.98%	650.11
Modus IoT dataset					
Gaussian NB		**Hoeffding Tree**		**Our proposed model**	
Acc	**Memory (KB)**	**Acc**	**Memory (KB)**	**Acc**	**Memory (KB)**
77.60%	19.2	92.36%	124.58	96.81%	495.25
Garage IoT dataset					
Gaussian NB		**Hoeffding Tree**		**Our proposed model**	
Acc	**Memory (KB)**	**Acc**	**Memory (KB)**	**Acc**	**Memory (KB)**
85.96%	27.05	95.70%	75.85	99.96%	394.95
Motion light IoT dataset					
Gaussian NB		**Hoeffding Tree**		**Our proposed model**	
Acc	**Memory (KB)**	**Acc**	**Memory (KB)**	**Acc**	**Memory (KB)**
85.86%	20.6	92.06%	33.56	99.98%	219.58
GPS Tracker IoT dataset					
Gaussian NB		**Hoeffding Tree**		**Our proposed model**	
Acc	**Memory (KB)**	**Acc**	**Memory (KB)**	**Acc**	**Memory (KB)**
85.20%	10.56	98.29%	120.36	99.97%	281.94
Weather IoT dataset					
Gaussian NB		**Hoeffding Tree**		**Our proposed model**	
Acc	**Memory (KB)**	**Acc**	**Memory (KB)**	**Acc**	**Memory (KB)**
86.08%	20.58	98.37%	314.91	99.93%	627.81
Thermostat IoT dataset					
Gaussian NB		**Hoeffding Tree**		**Our proposed model**	
Acc	**Memory (KB)**	**Acc**	**Memory (KB)**	**Acc**	**Memory (KB)**
87.27%	6.85	99.12%	55.07	99.94%	122.38

of 92.96%, precision, recall, and F1 scores of 92.36%, 92.36%, and 92.36%, respectively. Using the GPS IoT dataset, the Gaussian NB had an average accuracy of 77.60% and precision, recall, and F1 scores of 60.21%, 77.60%, and 67.81%, respectively. Fig. 10.5 shows the results of our model when evaluated using the modus IoT dataset.

Similarly, our proposed model has a superior average accuracy of 99.98% when it was evaluated using the fridge IoT dataset. The proposed model also recorded the same precision, recall, and F1 score value using the same dataset. Hoeffding tree algorithm recorded 98.63%, 98.68%, 98.52%, and 98.52% for precision, recall, F1 score, and average accuracy, respectively. Gaussian NB recorded the lowest average accuracy, an average accuracy of 85.31%. Gaussian NB also recorded the least values for precision, recall, and F1 scores, with 72.8%, 85.31%, and 78.55%,

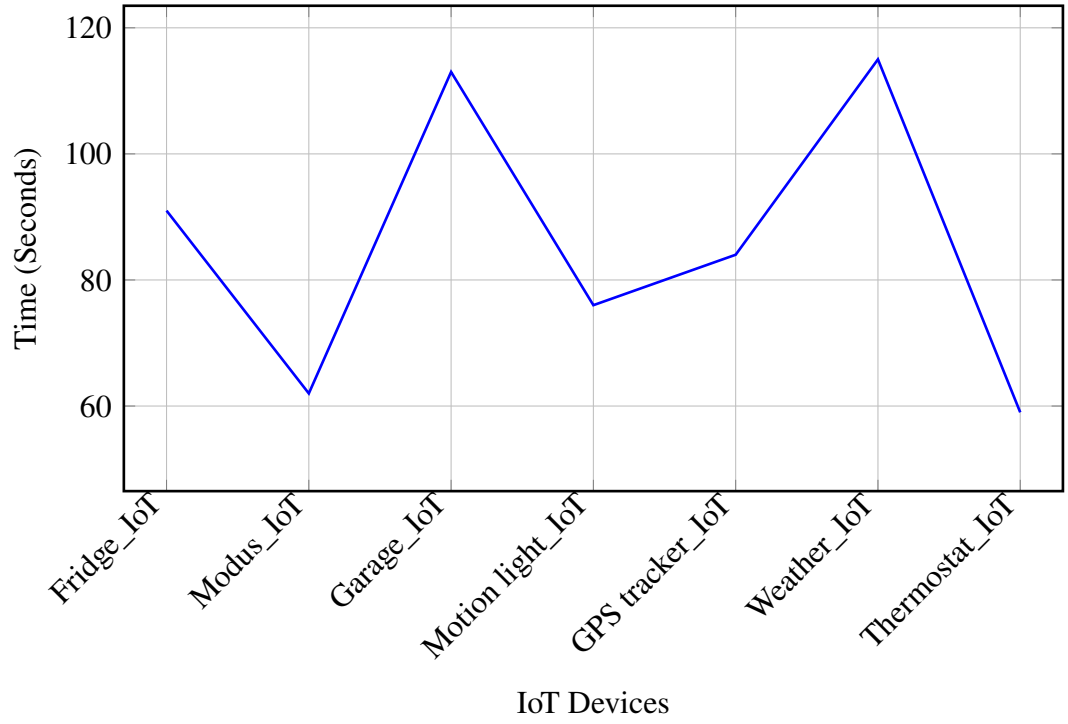


Figure 10.4: Model training time of our model using the TON IoT dataset

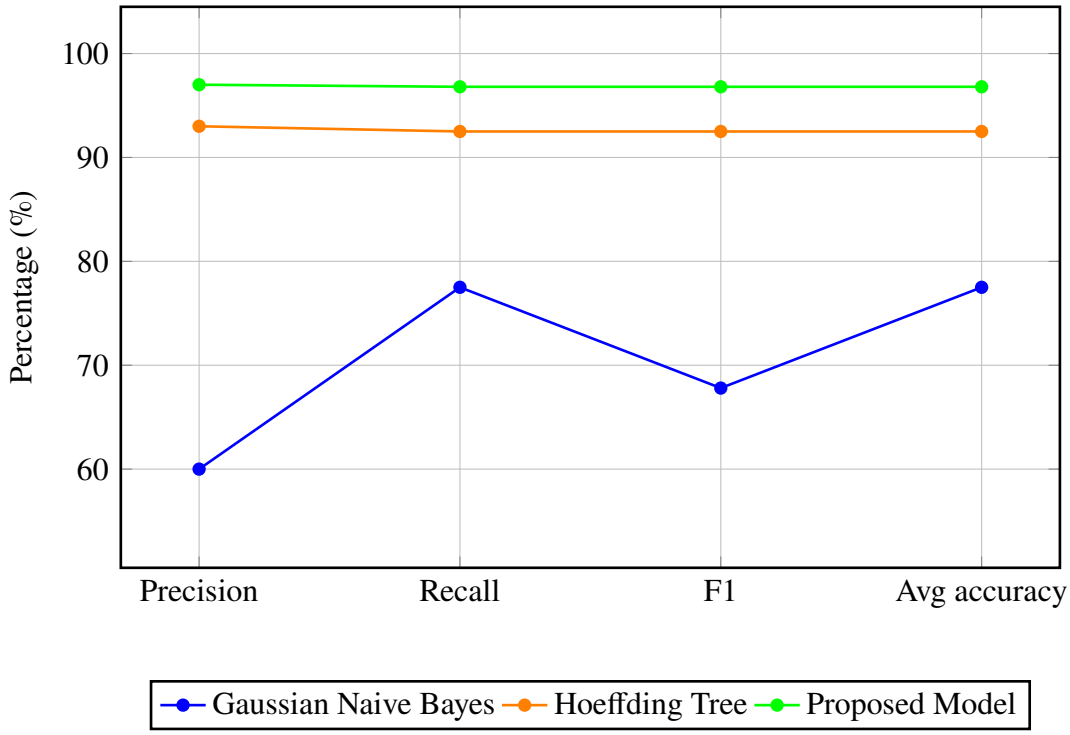


Figure 10.5: Accuracy of Gaussian NB, HT, and proposed model on Modus dataset

respectively. Fig. 10.6 illustrates the outcomes of our model when tested against the fridge IoT dataset.

The experimental findings show that our proposed method performed better when tested using the motion IoT dataset. The proposed ensemble model achieved an average accuracy of 99.98% with precision and recall of the same value while recording 99.97% for the F1 score. The same dataset revealed that the Hoeffding tree recorded an average accuracy of 92.06% while recording a precision, recall,

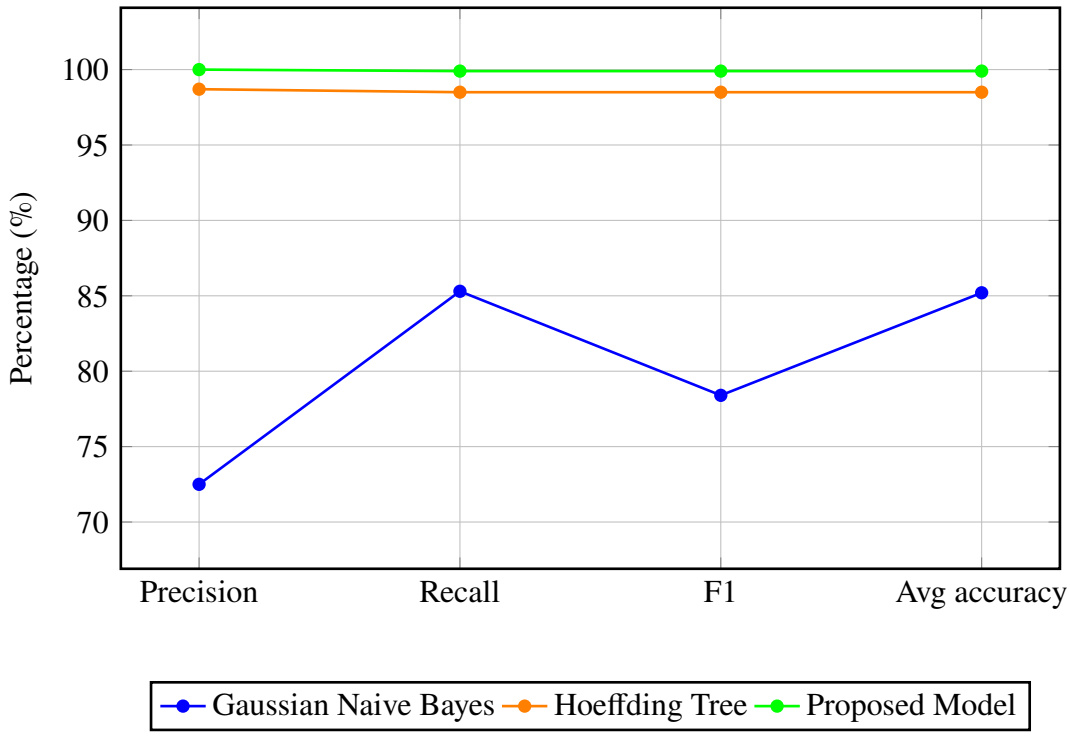


Figure 10.6: Accuracy of Gaussian NB, HT, and proposed model on Fridge dataset.

and F1 score of 88.56%, 92.06%, and 89.64%, respectively. The average accuracy, precision, recall, and F1 score recorded by Gaussian NB are 85.86%, 73.73%, 85.86%, and 79.33%, respectively. Fig. 10.7 depicts the results of our model when tested against the motion IoT dataset.

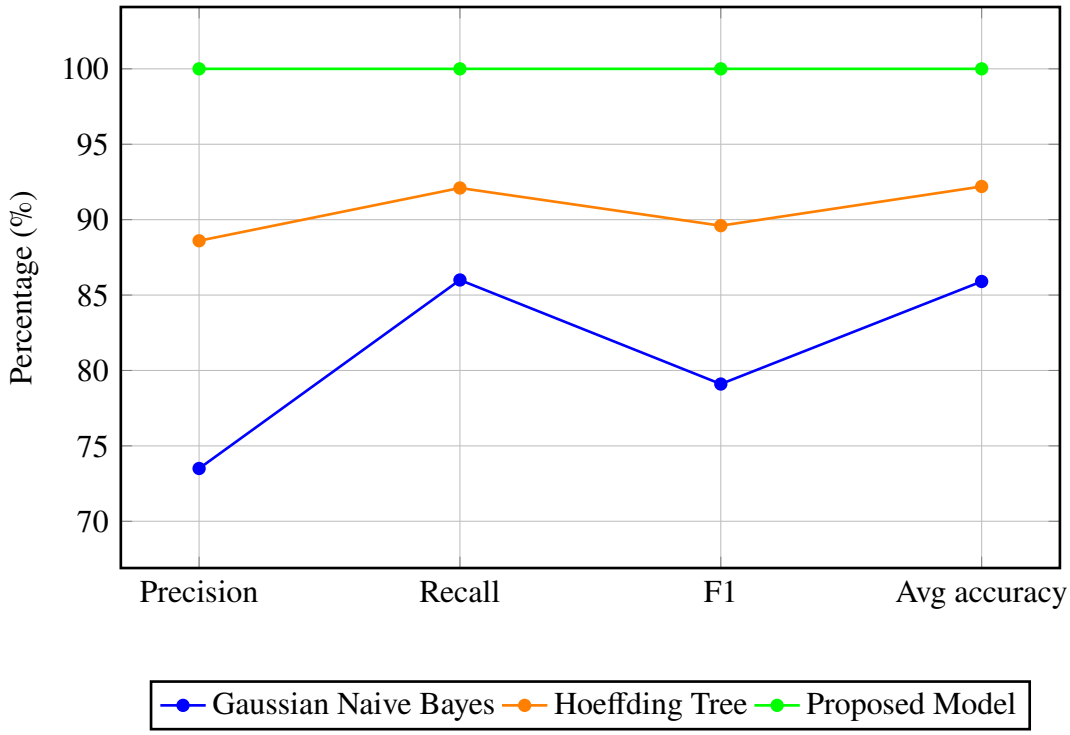


Figure 10.7: Accuracy of Gaussian NB, HT, and proposed model on Motion dataset.

When tested on the garage IoT dataset, our proposed ensemble model again had the highest precision, recall, F1 score, and average accuracy. Our proposed model

recorded an average accuracy of 99.96%, with the same value recorded for precision, recall, and F1 score. Hoeffding tree, on the other hand, recorded a precision, recall, F1 score, and average accuracy of 95.52%, 95.70%, 95.26%, and 95.70%, respectively. Gaussian Naive Bayes recorded a precision, recall, F1 score, and average accuracy of 73.88%, 85.96%, 79.46%, and 85.96%, respectively, when it was evaluated using the garage IoT dataset. Figure 10.8 shows the results of our model when tested against the garage IoT dataset.

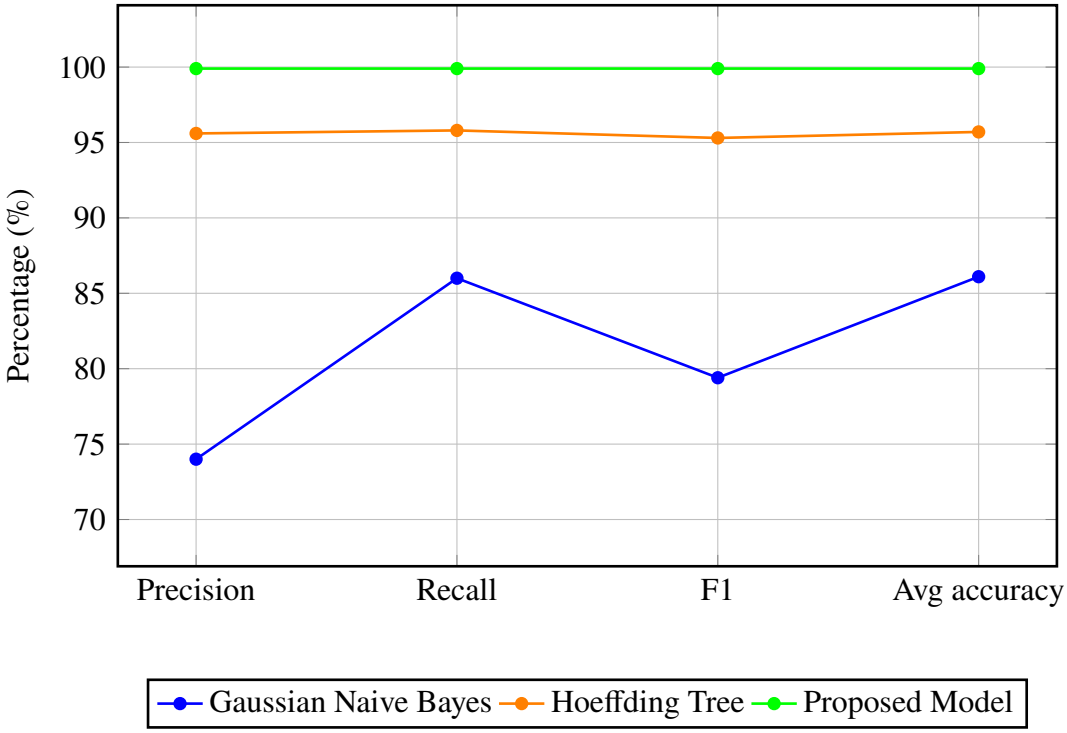


Figure 10.8: Accuracy of Gaussian NB, HT, and proposed model on Garage dataset.

Evaluating our proposed model on the GPS tracker dataset, our model achieved a superior average accuracy of 99.97% with precision, recall, and F1 score of 99.97% each. Hoeffding tree recorded an average accuracy of 98.29% while recording a precision, recall, and F1 score of 98.36%, 98.29%, and 98.08%, respectively. Evaluating the Gaussian NB using the The GPS IoT dataset revealed an average accuracy of 85.20% with a precision, recall, and F1 score of 88.26%, 85.20%, and 82.02%, respectively. Fig. 10.9 shows the results of our model when evaluated using the GPS tracker IoT dataset.

The experimental result shows that when our proposed model is evaluated using the thermostat IoT dataset, the model achieved an average accuracy of 99.94% with precision, recall, and F1 score of 99.94% for each of them, respectively. Gaussian NB showed an average accuracy of 87.27% while recording a precision, recall, and

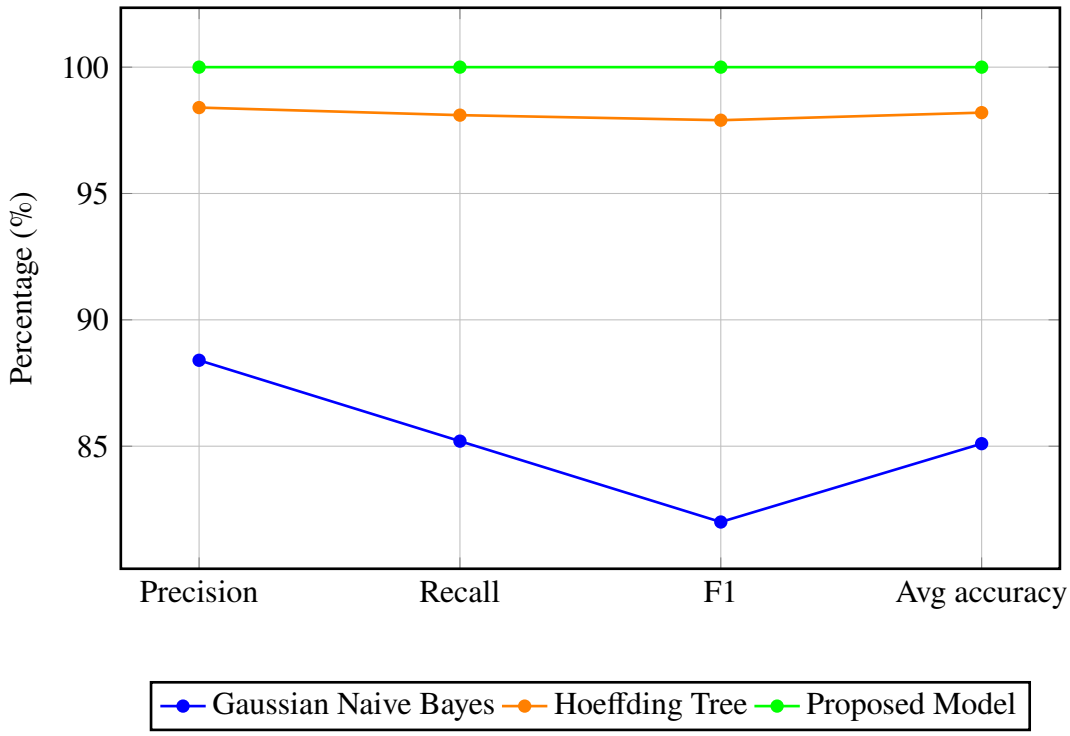


Figure 10.9: Accuracy of Gaussian NB, HT, and proposed model on GPS Tracker dataset.

F1 score of 76.17%, 87.27%, and 81.34%, respectively. Hoeffding tree showed an average accuracy of 99.12% with precision, recall, and F1 score of 99.07%, 99.12% and 99.03% respectively. Fig. 10.10 shows the results of our model when tested against the thermostat IoT dataset.

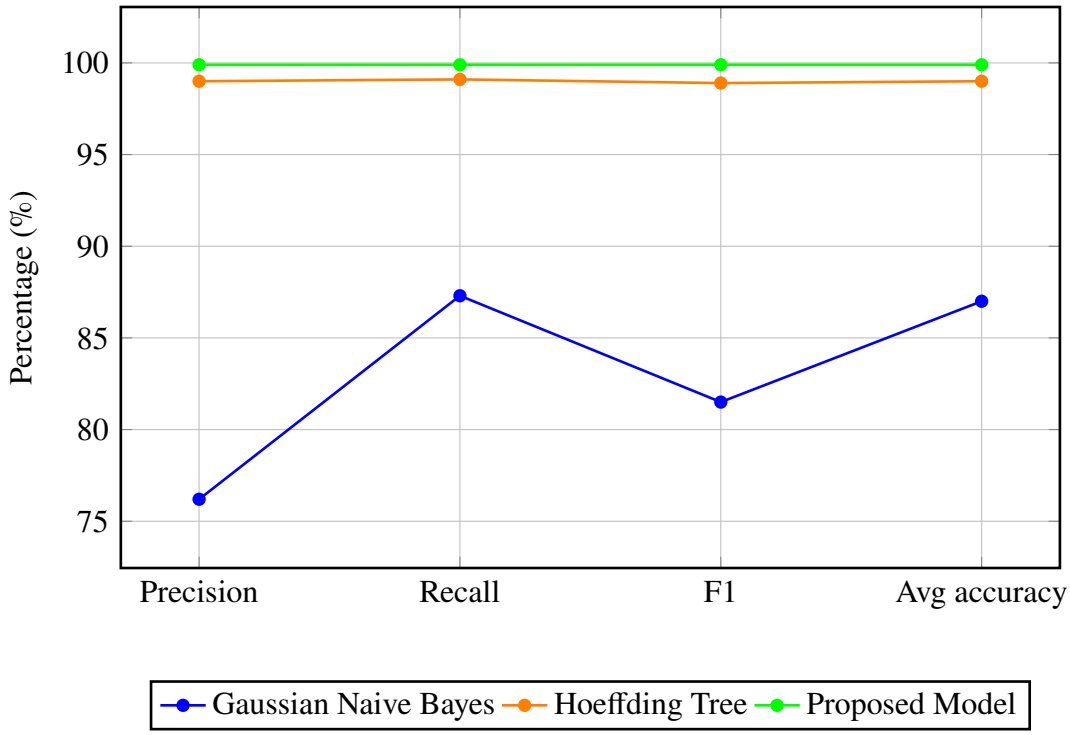


Figure 10.10: Accuracy of Gaussian NB, HT, and proposed model on the Thermostat dataset.

We also evaluated our model on the weather IoT dataset, one of the datasets found in the TON IoT dataset. The results show that Gaussian NB recorded preci-

sion, recall, F1, and average accuracy of 80.02%, 86.08%, 76.65%, and 86.08%, respectively. On the other hand, the Hoeffding tree recorded precision, recall, F1, and average accuracy of 98.47%, 98.37%, 98.30%, and 98.37%, respectively. However, our proposed ensemble technique recorded an average accuracy of 99.93% with precision, recall, and F1 score of 99.93%, respectively. Fig. 10.11 illustrates the outcomes of our model when tested against the weather IoT dataset.

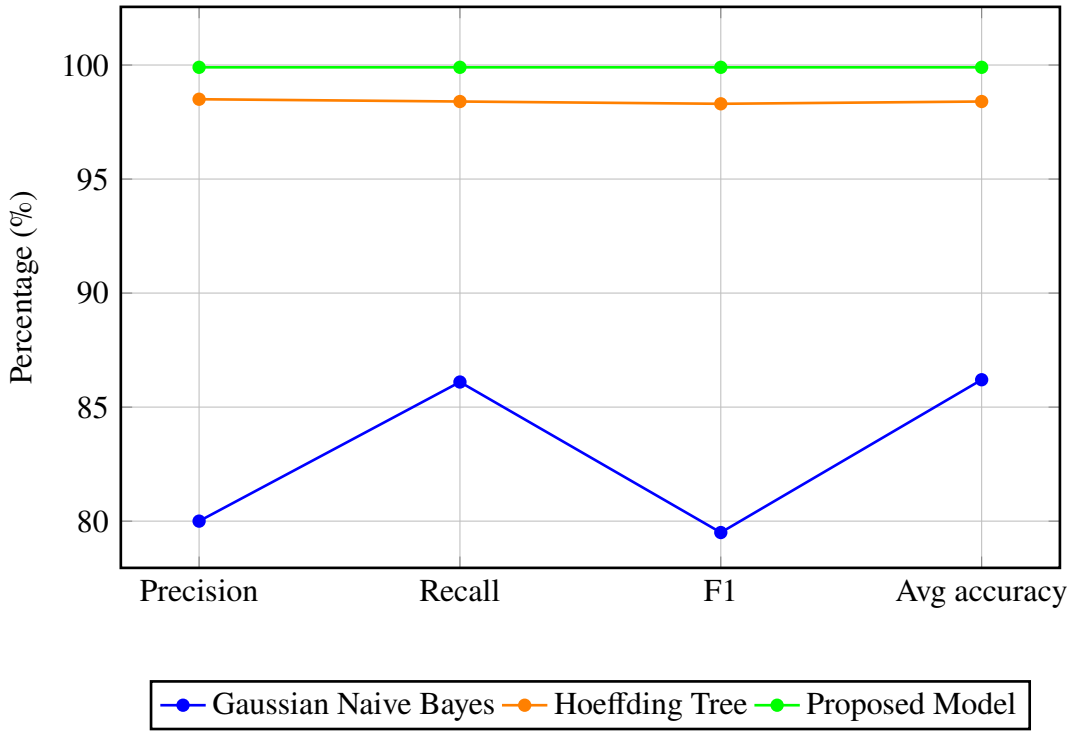


Figure 10.11: Accuracy of Gaussian NB, HT, and proposed model on Weather dataset.

The Fridge IoT dataset recorded the highest consumption of 650.11 KB, while the weather IoT dataset recorded the lowest memory consumption of 122.38 KB. The results of the memory consumption of the model proposed in this study show that even at the highest memory consumption, the proposed IDS achieves a lightweight status and can potentially run on IoT devices without significantly affecting the available memory of these devices. The memory consumption of our proposed model is shown in Fig. 10.12.

The F1 scores of Gaussian Naive Bayes, Hoeffding tree, and our proposed model on different attack categories are shown in Tables 10.5 and 10.6 below.

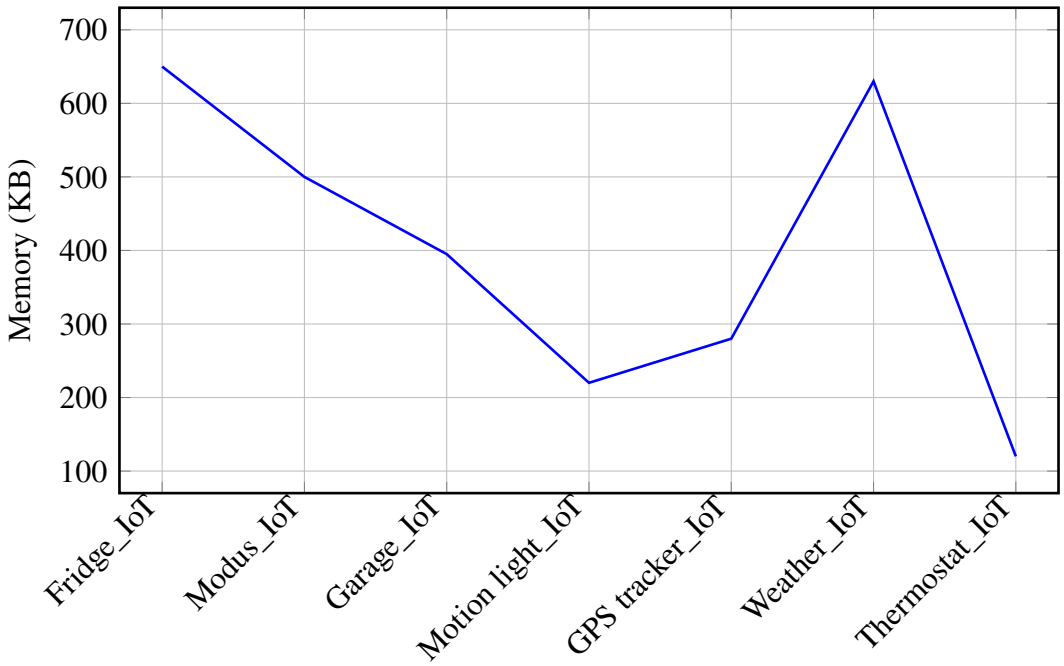


Figure 10.12: Memory consumption of our proposed model on various sub-datasets of the TON IoT dataset

Table 10.5: Comparing the F1 score of each model on each attack category

Fridge IoT dataset			
Gaussian NB		**Hoeffding Tree**	**Our proposed model**
Attack	**F1 Score (%)**	**F1 Score (%)**	**F1 Score (%)**
Backdoor	0.00	99.06	99.86
DDoS	0.00	91.99	99.97
Injection	0.00	79.12	99.92
Normal	92.07	99.98	99.99
Password	0.00	91.52	99.97
Ransomware	0.00	33.64	99.64
XSS	0.00	20.33	99.71
Modus IoT dataset			
Backdoor	0.00	76.97	92.10
Injection	0.00	25.33	72.63
Normal	87.39	99.99	99.99
Password	0.00	55.43	79.36
Scanning	0.00	54.31	79.62
XSS	0.00	0.40	20.63
Garage IoT dataset			
Backdoor	0.00	96.86	99.79
DDoS	0.00	52.06	99.32
Injection	0.00	0.00	99.79
Normal	92.45	99.99	100.00
Password	0.00	55.23	99.87
Ransomware	0.00	0.00	99.86
Scanning	0.00	0.00	99.24
XSS	0.00	0.00	99.57
Motion light IoT dataset			
Backdoor	0.00	60.92	99.83
DDoS	0.00	0.00	99.95
Injection	0.00	0.00	99.94
Normal	92.39	99.97	99.99
Password	0.00	0.00	99.98
Ransomware	0.00	0.00	99.82
Scanning	0.00	0.00	99.77
XSS	0.00	0.00	99.55
GPS Tracker IoT dataset			
Backdoor	13.90	98.66	99.61
DDoS	15.91	79.47	99.92
Injection	24.58	79.40	99.86
Normal	92.31	99.98	99.99
Password	20.00	84.32	99.95
Ransomware	29.94	14.29	99.37
Scanning	0.36	33.84	99.82
XSS	12.08	0.00	97.88

Table 10.6: Comparing the F1 score of each model on each attack category continuation

Gaussian NB		**Hoeffding Tree**	**Our proposed model**
Attack	**F1 Score(%)**	**F1 Score (%)**	**F1 Score (%)**
Weather IoT dataset			
Backdoor	0.10	97.92	99.94
DDoS	0.00	88.28	99.87
Injection	0.00	78.18	98.62
Normal	92.52	99.98	100.00
Password	0.00	83.45	99.45
Ransomware	0.63	49.56	98.11
Scanning	0.00	55.66	90.37
XSS	0.00	56.98	96.97
Thermostat IoT dataset			
Backdoor	0.00	98.44	99.84
Injection	0.00	90.94	99.45
Normal	93.20	99.97	99.99
Password	0.00	85.00	99.26
Ransomware	0.00	55.27	98.89
Scanning	0.00	0.00	76.47
XSS	0.00	2.64	97.25

10.6 Limitations of the Study

The main limitation of the study is how the proposed method can be used to achieve good accuracy, higher detection speed, and lower resource consumption at the same time. One approach that can be used to overcome this limitation is to deploy the proposed model on a resource-constrained device while fine-tuning the model to achieve a good tradeoff among the parameters mentioned above.

10.7 Conclusion

The security of the Internet of Things ecosystems is increasingly gaining significant importance due to its numerous applications. The security of IoT systems has gone beyond encryption, authentication, and secure architecture. Recently, much security-based research in IoT systems has been focused on detecting attacks and anomalies in network traffic. However, because of the computational constraints of IoT devices, IDS developed for traditional computing systems cannot be deployed in IoT environments. It is, therefore, expedient to design a lightweight IDS that can be deployed on IoT devices. In this view, we used the incremental machine learning technique to design a lightweight IDS for IoT systems using incremental ensemble machine learning algorithms. Our proposed model was evaluated using the TON IoT dataset. The results show that our proposed model achieved a high average accuracy rate of 99.98%. The experimental results show the highest memory consumption at 650.11 KB and the lowest at 122.38 KB. The experimental result shows that our approach has led to the design of an IDS with a high accuracy rate and a lightweight model that can potentially run on IoT devices. In the future, we plan to evaluate our approach on other IoT-based datasets and deploy our model on an IoT device to evaluate parameters such as CPU usage, memory, and energy consumption. Additionally, future work could consider exploring how concept drift in these datasets could be handled.

References

[1] Statista, • *Global IoT and non-IoT connections 2010-2025 | Statista*. [Online]. Available: https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/ (visited on 07/05/2022).

[2] SAM Seamless Network, *2021 IoT Security Landscape - SAM Seamless Network*. [Online]. Available: `https://securingsam.com/2021-iot-security-landscape/` (visited on 07/13/2022).

[3] J. P. Anderson, "Computer security threat monitoring and surveillance", *Technical Report, James P. Anderson Company*, 1980.

[4] J. R. Vacca, *Computer and information security handbook*. Newnes, 2012.

[5] S. U. Jan, S. Ahmed, V. Shakhov, and I. Koo, "Toward a lightweight intrusion detection system for the internet of things", *IEEE Access*, vol. 7, pp. 42 450–42 471, 2019.

[6] S. Latif, Z. e Huma, S. S. Jamal, *et al.*, "Intrusion detection framework for the internet of things using a dense random neural network", *IEEE Transactions on Industrial Informatics*, 2021.

[7] S. Roy, J. Li, B.-J. Choi, and Y. Bai, "A lightweight supervised intrusion detection mechanism for iot networks", *Future Generation Computer Systems*, vol. 127, pp. 276–285, 2022.

[8] R. Zhao, G. Gui, Z. Xue, *et al.*, "A novel intrusion detection method based on lightweight neural network for internet of things", *IEEE Internet of Things Journal*, 2021.

[9] T. D. Diwan, S. Choubey, H. Hota, *et al.*, "Feature entropy estimation (fee) for malicious iot traffic and detection using machine learning", *Mobile Information Systems*, vol. 2021, 2021.

[10] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines", in *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, IEEE, vol. 2, 2002, pp. 1702–1707.

[11] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges", *computers & security*, vol. 28, no. 1-2, pp. 18–28, 2009.

[12] D. E. Denning, "An intrusion-detection model", *IEEE Transactions on software engineering*, no. 2, pp. 222–232, 1987.

[13] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in internet of things", *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[14] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for iot security based on learning techniques", *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2671–2701, 2019.

[15] S. Otoum, B. Kantarci, and H. T. Mouftah, "A novel ensemble method for advanced intrusion detection in wireless sensor networks", in *Icc 2020-2020 ieee international conference on communications (icc)*, IEEE, 2020, pp. 1–6.

[16] C. Zhang and Y. Ma, *Ensemble machine learning: methods and applications*. Springer, 2012.

[17] T. T. Khoei, G. Aissou, W. C. Hu, and N. Kaabouch, "Ensemble learning methods for anomaly intrusion detection system in smart grid", in *2021 IEEE International Conference on Electro Information Technology (EIT)*, IEEE, 2021, pp. 129–135.

[18] R. E. Schapire, "The strength of weak learnability", *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[19] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts", *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[20] S. Muthukrishnan *et al.*, "Data streams: Algorithms and applications", *Foundations and Trends® in Theoretical Computer Science*, vol. 1, no. 2, pp. 117–236, 2005.

[21] A. A. Benczúr, L. Kocsis, and R. Pálovics, "Online machine learning in big data streams", *arXiv preprint arXiv:1802.05872*, 2018.

[22] A. Khannoussi, A.-L. Olteanu, C. Labreuche, *et al.*, "Integrating operators' preferences into decisions of unmanned aerial vehicles: Multi-layer decision engine and incremental preference elicitation", in *International Conference on Algorithmic Decision Theory*, Springer, 2019, pp. 49–64.

[23] A. Mozaffari, M. Vajedi, and N. L. Azad, "A robust safety-oriented autonomous cruise control scheme for electric vehicles based on model predictive control and online sequential extreme learning machine with a hyper-level fault tolerance-based supervisor", *Neurocomputing*, vol. 151, pp. 845–856, 2015.

[24] F. Feng, R. H. Chan, X. Shi, Y. Zhang, and Q. She, "Challenges in task incremental learning for assistive robotics", *IEEE Access*, vol. 8, pp. 3434–3441, 2019.

[25] M. R. Henzinger, P. Raghavan, and S. Rajagopalan, "Computing on data streams.", *External memory algorithms*, vol. 50, pp. 107–118, 1998.

[26] L. Yang, D. M. Manias, and A. Shami, "Pwpae: An ensemble framework for concept drift adaptation in iot data streams", in *2021 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2021, pp. 01–06.

[27] V. Shakhov, S. U. Jan, S. Ahmed, and I. Koo, "On lightweight method for intrusions detection in the internet of things", in *2019 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, IEEE, 2019, pp. 1–5.

[28] J.-S. Pan, F. Fan, S.-C. Chu, H.-Q. Zhao, and G.-Y. Liu, "A lightweight intelligent intrusion detection model for wireless sensor networks", *Security and Communication Networks*, vol. 2021, 2021.

[29] L. H. A. Reis, A. Murillo Piedrahita, S. Rueda, *et al.*, "Unsupervised and incremental learning orchestration for cyber-physical security", *Transactions on emerging telecommunications technologies*, vol. 31, no. 7, e4011, 2020.

[30] A. Tabassum, A. Erbad, A. Mohamed, and M. Guizani, "Privacy-preserving distributed ids using incremental learning for iot health systems", *IEEE Access*, vol. 9, pp. 14 271–14 283, 2021.

[31] S. D. Jadhav and H. Channe, "Comparative study of k-nn, naive bayes and decision tree classification techniques", *International Journal of Science and Research (IJSR)*, vol. 5, no. 1, pp. 1842–1845, 2016.

[32] A. McCallum, K. Nigam, *et al.*, "A comparison of event models for naive bayes text classification", in *AAAI-98 workshop on learning for text categorization*, Citeseer, vol. 752, 1998, pp. 41–48.

[33] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[34] T. M. Mitchell and T. M. Mitchell, *Machine learning*. McGraw-hill New York, 1997, vol. 1.

[35] C. Bustamante, L. Garrido, and R. Soto, "Comparing fuzzy naive bayes and gaussian naive bayes for decision making in robocup 3d", in *Mexican International Conference on Artificial Intelligence*, Springer, 2006, pp. 237–247.

[36] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams", in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001, pp. 97–106.

[37] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers", in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 226–235.

[38] J. Montiel, M. Halford, S. M. Mastelini, *et al.*, "River: Machine learning for streaming data in python", *J. Mach. Learn. Res.*, vol. 22, no. 1, Jan. 2021, ISSN: 1532-4435.

[39] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "Ton_iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems", *Ieee Access*, vol. 8, pp. 165 130–165 150, 2020.

[40] Q. Abu Al-Haija, A. Al Badawi, and G. R. Bojja, "Boost-defence for resilient iot networks: A head-to-toe approach", *Expert Systems*, e12934, 2022.

[41] M. A. Khan, M. A. Khan Khattk, S. Latif, *et al.*, "Voting classifier-based intrusion detection for iot networks", in *Advances on Smart and Soft Computing*, Springer, 2022, pp. 313–328.

[42] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, "E-graphsage: A graph neural network based intrusion detection system for iot", in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2022, pp. 1–9.

[43] A. R. Gad, A. A. Nashat, and T. M. Barkat, "Intrusion detection system using machine learning for vehicular ad hoc networks based on ton-iot dataset", *IEEE Access*, vol. 9, pp. 142 206–142 217, 2021.

Chapter 11

General conclusion and future work

This chapter summarizes the thesis, highlights the contribution of this work, and recommends some aspects of this work that could be explored as future work.

11.1 Summary

In this thesis, we designed an IoT-based IDS that dynamically adapts to changes in network traffic without having to retrain the model from scratch using an online ML algorithm. Considering the heterogeneity of the IoT ecosystem and its resource-constrained nature, we designed an IDS that is adaptive, lightweight, able to handle drift, and can detect unknown attacks.

- **Question 1: Given the dynamic and heterogeneous nature of IoT environments, which machine learning algorithms are most applicable for achieving effective intrusion detection?**
  Chapters 4, 5, 6, 7, 8, 9, and 10 show that the online ML algorithms developed in this thesis are adaptive, making them suitable for designing IoT-based IDS.

- **Question 2: How can intrusion detection models be designed to detect intrusions in real-time, adapt to evolving attack patterns, and adapt to concept drift in IoT network traffic?**
  In chapter 4, we proposed an online deep learning algorithm with dynamic quantization that not only reduces the model size but is also able to recover quickly in the presence of concept drift. In Chapters 7 and 8, we modified the SAMKNN algorithm to handle drifts in streaming environments. Concept drifts are predominant in IoT environments because of their heterogeneous nature of IoT devices. The results show that the proposed adaptive model is able to perform comparably well in the midst of gradual, sudden, recurring, and incremental drifts.

- **Question 3: What strategies can be applied to reduce computational and memory overhead in IDS models, enabling their deployment on resource-constrained IoT and edge devices without significantly compromising detection accuracy?**
  The technique used to design an efficient and lightweight IDS for IoT environments is described in Chapter 4. In chapter 4, we were able to use an online deep learning algorithm with a dynamic quantization technique. This led to the proposed IDS reducing about 96% of the model size without negatively affecting its detection accuracy. The techniques used in chapters 9 and 10 also led to producing lightweight models.

- **Question 4: To what extent do the proposed IDS approaches maintain effectiveness when validated on real-world IoT datasets and deployed in embedded platform scenarios?**
  In Chapters 4, 5, and 6, we validated the proposed models across multiple datasets. In chapters 4 and 9, we deployed the proposed IDS model on a Raspberry Pi and measured metrics such as CPU and energy usage.

11.2 Limitation

Despite the strengths of this study, the proposed approach had some limitations.

1. Limited Dataset: The evaluation of the system was limited to a number of datasets from different IoT domains. These datasets may not fully represent all the variants and complexities that exist in heterogeneous real-world IoT networks.

2. Adversarial Scope: The adversarial evaluation adopted basic perturbation strategies. More complex attack vectors, such as adaptive adversaries and transfer-based attacks, were not explored.

11.3 Future work

Although the objectives of this work have been achieved, coupled with promising results, there are still unexplored areas that can be considered as future work. One of the areas that will be considered for feature work is identifying attacks in encrypted network traffic. This is a challenging area that has attracted a lot of

research in recent years. Using online ML and other techniques to detect anomalies within encrypted traffic is a viable and interesting problem that should be explored as future work. Additionally, the use of unsupervised online ML approaches to detect intrusions in IoT environments is another interesting area we will explore as future work.

Appendix

11.4 Code of the algorithm used in Chapter 4

The codes used for our experimental validation for Chapter 4 are available in the GitHub repository.
https://github.com/v-pragbe/Adaptive_Online_Deep_Quantization

11.5 Code of the algorithm used in Chapter 5

The codes used for our experimental validation for Chapter 5 are available in the GitHub repository.
https://github.com/v-pragbe/IoT_ALMANET

11.6 Code of the algorithm used in Chapter 6

The codes used for our experimental validation for Chapter 6 are available in the GitHub repository.
https://github.com/v-pragbe/Online_Multi_Label_IDS

11.7 Code of the algorithm used in Chapters 7 and 8

The codes used for our experimental validation for Chapters 7 and 8 are available in the GitHub repository.
https://github.com/v-pragbe/Adaptive_Online_Attack_Detection