UNIVERSITY OF KWAZULU-NATAL

Investigating Machine- and Deep-Learning Model Combinations for a Two-Stage IDS for IoT Networks

by

André van der Walt

Supervised by:

Dr Tahmid Quazi

Dr Brett van Niekerk

A dissertation submitted in fulfilment for the degree of

Master of Science in Computer Engineering

In the

Discipline of Electrical, Electronic and Computer Engineering,

College of Agriculture, Engineering and Science,

University of KwaZulu-Natal, Durban, South Africa

December 2021

Declaration 1 - Plagiarism

I, **André van der Walt**, declare that this dissertation titled, '**Investigating Machine- and Deep-Learning Model Combinations for a Two-Stage IDS for IoT Networks.**' and the work presented in it are my own. I confirm that:

1. The research reported in this dissertation, except where otherwise indicated, is my original work.
2. This dissertation has not been submitted for any degree or examination at any other university.
3. This dissertation does not contain other person's data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
4. This dissertation does not contain other persons writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
    - i.) Their words have been re-written, but the general information attributed to them has been referenced and;
    - ii.) Where their exact words have been used, their writing has been placed inside quotation marks, and referenced.
5. Where I have reproduced a publication of which I am an author, co-author or editor, I have indicated in detail which part of the publication was written by myself alone and have fully referenced such publications.
6. This dissertation does not contain text, graphics or tables copied and pasted from the internet, unless specifically acknowledged, and the source being detailed in the dissertation and in the References sections.

Signed:

Date:

03 December 2021

As the candidate's supervisor, I, **Tahmid Quazi**, agree to the submission of this dissertation.

Signed:

Date:

03 December 2021

As the candidate's supervisor, I, **Brett van Niekerk**, agree to the submission of this dissertation.

Signed:

Date:

03 December 2021

Declaration 2 - Publications

I **André van der Walt,** declare that the following publications were produced as a result of the work reported in this dissertation.

Article 1

Title	Two-Stage IDS for IoT Using Layered Machine- and Deep-Learning Models
Authors	André van der Walt, Tahmid Quazi and Brett van Niekerk
Type	Journal Article
Status	Under Review at *Cyber-Physical Systems* (ISSN: 2333-5785)

Article 2

Title	Investigating Machine- and Deep-Learning Model Combinations for a Two-Stage IDS for IoT Networks
Authors	André van der Walt, Tahmid Quazi and Brett van Niekerk
Type	Journal Article
Status	Under Review at *Sensors* (ISSN: 1424-8220)

I declare that the experimental work contained in these publications was performed as a joint effort by me, Dr Tahmid Quazi and Dr Brett van Niekerk, and the writing of each publication was done by me with assistance from Dr Tahmid Quazi and Dr Brett van Niekerk, and the work contained in these publications has been reproduced in this dissertation.

Signed: ___________________________

Acknowledgements

Thank you to Dr Tahmid Quazi and Dr Brett van Niekerk for guiding me and believing in my ability even when I did not. To my parents, brother, and grandmother: Thank you. I cannot, in this single paragraph, explain how much your support, teachings, willingness to listen and guidance have helped me. This work is the culmination of years of belief, trust and support given to me by my family, friends, mentors, and teachers. Without them, this would not have been possible, and I would not be the person I am today.

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveller, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same,

And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I—
I took the one less travelled by,
And that has made all the difference.

~ ‘The Road Not Taken’ by Robert Frost

Abstract

Master of Science in Computer Engineering

Investigating Machine- and Deep-Learning Model Combinations for a Two-Stage IDS for IoT Networks.

by André van der Walt

By 2025, there will be upwards of 75 billion IoT devices connected to the internet. Notable security incidents have shown that many IoT devices are insecure or misconfigured, leaving them vulnerable, often with devastating results. AI's learning, adaptable and flexible nature can be leveraged to provide networking monitoring for IoT networks.

This work proposes a novel two-stage IDS, using layered machine- and deep-learning models. The applicability of seven algorithms is investigated using the BoT-IoT dataset. After replicating four algorithms from literature, modifications to these algorithms' application are then explored along with their ability to classify in three scenarios: 1) binary attack/benign, 2) multi-class attack with benign and 3) multi-class attack only. Three additional algorithms are also considered. The modifications are shown to achieve higher F1-scores by 22.75% and shorter training times by 35.68 seconds on average than the four replicated algorithms. Potential benefits of the proposed two-stage system are examined, showing a reduction of threat detection/identification time by 0.51s on average and an increase of threat classification F1-score by 0.05 on average. In the second half of the dissertation, algorithm combinations, layered in the two-stage system, are investigated. To facilitate comparison of time metrics, the classification scenarios from the first half of the dissertation are re-evaluated on the test PC CPU. All two-stage combinations are then tested. The results show a CNN binary classifier at stage one and a KNN 4-Class model at stage two performs best, outperforming the 5-Class (attack and benign) system of either algorithm. This system's first stage improves upon the 5-Class system's classification time by 0.25 seconds. The benign class F1-score is improved by 0.23, indicating a significant improvement in false positive rate. The system achieves an overall F1-score of 0.94. This shows the two-stage system would perform well as an IDS. Additionally, investigations arising from findings during the evaluation of the two-stage system are presented, namely GPU data-transfer overhead, the effect of data scaling and the effect of benign samples on stage two, giving a better understanding of how the dataset interacts with AI models and how they may be improved in future work.

Contents

List of Figures

List of Tables

Abbreviations and Definitions

Abbreviation	**Definition**
6LoWPAN	The IPv6 version of a low power wireless personal area network
AE	Autoencoder
AI	Artificial Intelligence
AIDS	Anomaly-based Intrusion Detection System
ANN	Artificial Neural Network
C5	A widely used decision tree method
CIC	Canadian Institute for Cybersecurity
CICFlowMeter	A network flow generation tool developed by the CIC
CL16	Referring to computer Random Access Memory (RAM) CAS Latency, where CAS stands for Column Address Strobe
CNN	Convolutional Neural Network
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
DAE	Deep Autoencoder
DBN	Deep Belief Network
DDoS	Distributed Denial-of-Service
DDR4	A shortening for DDR4-SDRAM which stands for Double Data Rate 4 Synchronous Dynamic Random-Access Memory
DL	Deep Learning
DNN	Deep Neural Network
DoS,	Denial-of-Service
DTLS	Datagram Transport Layer Security
FN	False Negative
FP	False Positive
FTP	File Transfer Protocol
GB	Gigabyte
GPU	Graphics Processing Unit
HIDS	Host Intrusion Detection System
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDPS	Intrusion Detection and Prevention System
IDS	Intrusion Detection System
IEEE 802.15.4	IEEE Standard for Low-Rate Wireless Networks
IoT	Internet-of-Things
IP	Internet Protocol
IPS	Intrusion Prevention System
Ipv6	Version 6 of the Internet Protocol
J48	A decision tree algorithm

KNN	K-Nearest Neighbours
LSTM	Long-Short Term Memory
ML	Machine Learning
MLP	Multi-Layer Perceptron
MQTT	Message Queueing Telemetry Transport
NB	Naïve Bayes
NIDS	Network Intrusion Detection System
Nmap	A Network Mapping Tool
NN	Neural Network
OCSVM	One-Class Support Vector Machine
OS	Operating System
PC	Personal Computer
PCAP	Packet Capture
PCIE	Peripheral Component Interconnect Express
R2L	Remote to Local
ReLU	Rectified Linear Unit
RF	Random Forest
RNN	Recurrent Neural Network
RPL	Routing Protocol for Low-Power and Lossy Networks
SIDS	Signature-based Intrusion Detection System
SMOTE	Synthetic Minority Oversampling Technique
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Sockets Layer
SVM	Support Vector Machine
SYN-ACK	Part of the three-way TCP handshake
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TN	True Negative
TP	True Positive
U2R	User to Root
UDP	User Datagram Protocol
UML	Unified Modelling Language

Chapter 1

Research Background

1.1 Introduction

Mass adoption of Internet of Things (IoT) devices has resulted in anywhere between 25 and 50 billion IoT devices currently in operation [1]. An estimated 79.4 Zettabytes of data is expected to be generated, by upwards of 75 billion devices, by 2025 [2], [3]. This trend is expected to proceed into the future and for IoT to become an increasingly pervasive part of daily life, business, health etc.

As we become increasingly reliant on the services and convenience offered by these devices, more of our data is shared and collected by them and their creators. With use-cases in healthcare, personal fitness, private document storage, financial administration etc., the list of areas in which IoT brings value is rapidly expanding. The modern-day adage of "information is the new currency", is now truer than ever.

IoT devices' wealth of information has made them prime targets for attackers and bad actors. Palo Alto Unit 42 [4] in its 2020 IoT Threat Report identify that 98% of IoT device traffic is unencrypted, with 57% of devices vulnerable to severe attacks. Unit 42 also notes that significant emphasis is placed on securing traditional computer systems, resulting in a lack of protection for custom operating system (OS) devices, such as IoT devices.

IoT devices come in many shapes and sizes with a plethora of varying goals, capabilities, and limitations. This inherently presents an issue in creating standard procedures for these devices. What may solve a set of problems for one group of devices, is completely infeasible for millions of others. This rings true specifically for security. The emphasis, in many cases, on 'lightweight' devices means many of these devices are created with low-compute ability and cheap components. This opens the door for attackers to exploit these characteristics and often result in catastrophic breaches in security.

Attacks based on the infamous Mirai Botnet are a perfect example of this, capable of generating Distributed Denial of Service (DDoS) events, generating 1.1Tbps of traffic and sustained attacks for days on end [5]. This is a singular example of the plethora of threats faced by IoT devices and network administrators. Palo Alto [4] identifies exploit attacks (scans, zero-days, SQL injection etc.) as the main threat to IoT devices currently. It also warns against a move towards 'worms' that can propagate through a network as the new preferred method of attack.

Network administrators face two major problems: first is the large number of false positives generated by traditional intrusion detection systems (IDS) and, second, the rapid evolution of the threat landscape, generating unique attacks at a pace that is difficult to keep up with [6], [7]. The learning nature of artificial intelligence (AI) can be leveraged to solve these problems where traditional systems may not, reducing the number of false positives while being able to detect unique, novel threats to the system.

The need to use artificial intelligence in IoT for various applications has been recognised for some time now [8]–[12]. This must be applied to the security of these devices as well [13]. The design of an IDS for these devices must consider the extreme variation found in the implementation of IoT devices as well as the variety in network conditions, all while remaining a system capable of identifying and preventing unique threats.

1.1.1 Background

This section covers necessary background information relating to the security of IoT devices, including some of the attacks faced by these systems, methods of securing them and the role of AI in their security.

1.1.1.1 IoT Architecture

All IoT devices perform the following four broad tasks: receive data, collect and transmit this data, process gathered data and use this data to perform a task such as activating a connected device or providing information to the user. This is accomplished through sensors, control units, communication modules and power sources [13]. The model of IoT architecture has taken many forms over the years as research developed the topic further. One of these models is the 5-layered model [14].

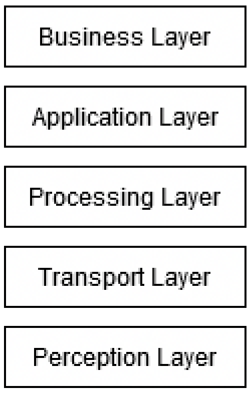


Figure 1.1: The 5-Layer IoT Architecture Model

1. **Perception Layer**: This is the physical layer that interacts and perceives the outside world, using sensors and actuators to gather information about the environment and effect change in it. Hardware level trojans have been reported to be an effective attack vector at this layer.

2. **Transport Layer**: In this layer, data is taken from the perception layer to the processing layer over protocols such as Wi-Fi, 3G/4G, Bluetooth etc. The wide variety of technologies utilised at this layer all introduce their own unique attack vectors.

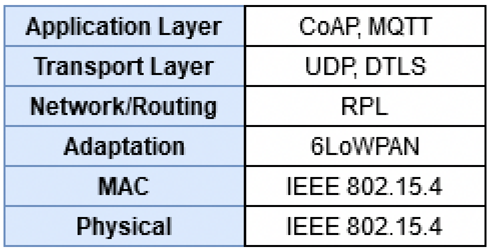
3. **Processing Layer**: This layer stores, prepares and analyses the data delivered to it as well as delivering services to the lower layers.

4. **Application Layer**: This layer gives effect to the application-specific services of the device. This is what "is sold on the box", so to speak. Examples include smart-home functionality, wearables etc.

5. **Business Layer**: This is the management layer that oversees the IoT system as a whole. It is used to monitor and control applications, user information and privacy, and business interests.

1.1.2 IoT Networking and Security

Understanding the network activity and protocols of IoT devices is greatly beneficial in understanding why and how these protocols and procedures could be, and are, abused to allow malicious actors undue influence over these devices. Various protocols are used at the different layers of the IoT device.



Application Layer	CoAP, MQTT
Transport Layer	UDP, DTLS
Network/Routing	RPL
Adaptation	6LoWPAN
MAC	IEEE 802.15.4
Physical	IEEE 802.15.4

Figure 1.2: IoT Stack [13]

1. **IEEE 802.15.4**: Approved in May 2020, this standard focusses on flexibility, low cost, very low power consumption and low data rate [15]. The transceiver spends most of its time in sleep mode and activates when communication is sensed. IEEE 802.15.4 provides link-layer security, meaning four basic security services are rendered: access control, message integrity, message confidentiality and replay protection [16]. Despite this, Raza et al. [17] show that there is a need to supplement the security provided by the IEEE 802.15.4 protocol.

2. **6LoWPAN**: The IPv6 version of a low power wireless personal area network (LoWPAN) operating over IEEE 802.15.4, it is a communication network designed for applications in which power is a restriction. It places an adaptation layer above the 802.15.4 link layer to allow for IP communication [18]. Raza et al. [19] emphasise the need for an IDS as these networks are vulnerable to attacks from both inside and outside the network.

3. **RPL**: Routing Protocol for Low-Power and Lossy Networks (RPL) is designed for use in environments similar to the IoT environment, where resources are limited. Gothawal and Nagaraj [20] note that an IDS designed specifically for the IoT environment is required to prevent attacks, such as sinkhole attacks or disabling attacks that deplete system resources.

4. **CoAP**: The Constrained Application Protocol (CoAP) provides some HTTP functions in a constrained environment such as IoT. Security at this layer is provided by the UDP variant of TLS (Datagram Transport Layer Security - DTLS) [13] for which, similar to TLS, it is beneficial to supplement defence by means of an IDS or IPS.

5. **MQTT**: The message Queue Telemetry Transport (MQTT) is a messaging protocol that was designed to allow low-powered processing and storage-limited devices to communicate over low-bandwidth connections. Aziz [21] shows that this protocol is vulnerable to attack, identifying Denial of Service as the largest threat. Aziz notes the use

of encryption algorithms like SSL would compromise the lightweight nature of the protocol. A supplementary IDS may then be a possible solution.

A clear trend emerges when the protocols at the different layers are considered in terms of their security. Significant security is either impossible to introduce or doing so would compromise the lightweight nature of these protocols. This then points to an auxiliary solution such as an IDS, applied supplementary to these devices that would allow for the stack to be sustained while simultaneously removing many of the security risks faces by IoT devices.

1.1.3 Threats

The set of threats faced not only by IoT devices but computer systems in general, is vast. Their origins, methods and objectives result in far too many attacks to account for individually. Understanding these threats is paramount to creating adequate defence systems.

1.1.3.1 External Threats

External threats to a system consist of a wide range of attack methods, objectives and damage that could be caused. Generally, intrusion refers to the act of accessing digital resources/data without the requisite authorisation. The internet is commonly used to intrude. Some common attack types are:

1.1.3.1.1 Denial of Service (Dos) Attack

The premise of this attack is that by forcing a computing system to become too busy, it can be prevented from performing its intended tasks. Flooding and flaw exploitation are ways in which a DoS attack can be achieved [22].

Flooding abuses the ability of a system to respond to external communications such as servicing requests, responding to ICMP Packets or attempting to establish a secure connection using the SYN-ACK handshake. Flaw exploitation leverages a vulnerability in the target system that can be abused to cause a system crash or for the system to otherwise hang [22].

1.1.3.1.2 User-to-Root (U2R) Attack

U2R attacks are a privilege escalation attack. A threat actor obtains the authentication details of a non-admin user and then, using flaws and vulnerabilities, increases the security level of that user to have administrator (root) privileges. This allows unfettered access to the system [22].

1.1.3.1.3 Port Scanning Attack

Port scanning uses tools such as Nmap to query the network for information regarding the ports used for communication. This is used primarily for reconnaissance of a target and allows for an attack strategy to be formulated by exposing the running services on a network, potentially vulnerable hosts and, sometimes, open and unprotected ports [22].

1.1.3.1.4 Man-in-the-Middle Attack

In this scenario, an attacker intercepts the communication between two parties, simultaneously impersonating them. The attacker is then able to receive the authentic information and replace it with modified, potentially malicious information. The communicating parties may be completely unaware of the attack's occurrence [22].

1.1.3.2 Insider Threats

Insider threats originate from trusted points inside the system. An example would be an employee of a company inadvertently allowing access to a malicious-intent threat. In some cases, insider threats are more dangerous than external threats as there may be very little to alert to a breach caused from within [23].

The 2020 Insider Threat Report compiled by Cybersecurity Insiders [24] shows that 68% of organisations surveyed feel vulnerable to insider attacks and believe that insider threats are becoming more frequent. Of the surveyed organisations, 53% also believe detecting insider threats are becoming more difficult. Furthermore, 63% believe that privileged IT professionals pose the highest risk. This also indicates the probability of a more sophisticated attack being employed. Interestingly, only 52% identify internal threats as more difficult to detect and prevent. This is important as it shows that, in the general sense, companies believe internal and external threats pose a similar level of danger.

1.1.4 Intrusion Detection Systems (IDS)

Intrusion Detection Systems are the systems put in place to defend against these threats. Intrusion Detection Systems come in many forms and are implemented a variety of ways:

1. **Host IDS**: Host Intrusion Detection Systems (HIDS) are used to protect a single system ('Host') from intrusions. Often this is accomplished by the IDS continually examining the log file of that system and then determining if any action or set of actions are potentially

dangerous or illegal. If such activity is found, the system is alerted, the activity is blocked, and the system administrator is alerted [22].

2. **Network IDS**: A Network Intrusion Detection System (NIDS) is installed at the network level and monitors for any unauthorized access or suspicious activity. These are the IDSs used to detect events such as DoS attacks, port scans etc [22]. Unfortunately, NIDSs suffer from a high false detection rate, both positive and negative. This makes some solutions unsuitable for full implementation [25].

3. **Active and Passive IDS**: NIDS is an active monitoring system as network activity is monitored in real time, whereas HIDS is passive, looking at the logs of a system which are generated after the fact. It is thus common practice to combine both these systems to give a more complete layer of security [22].

4. **Signature-based**: Signature-based IDS (SIDS) methods use data gathered from previous attacks to determine if current behaviour is malicious. The 'signature' behaviour of an attack is stored and used for later evaluation of new threats. This information can be updated with the signatures of new attacks as they become available [22]. While this method is extremely accurate in determining whether behaviour is malicious if it follows a previously known attack pattern, new attacks would not be detected [26]. This is obviously a huge vulnerability when using signature-based detection.

5. **Anomaly-based**: An anomaly-based IDS (AIDS) method compares detected behaviour to recorded normal behaviour for a specific network or host. When an action deviates far enough from this measure of normalcy, an anomaly is declared and an alarm raised. This method has a substantial benefit and an equally substantial disadvantage. It can detect threats that may be completely new, not only in terms of the system, but also globally. Unfortunately, this method may also produce many false positives when detecting intrusions [22].

6. **Placement**: This refers to where the IDS exists on the network, being either centralised or distributed. A centralised IDS sits at a border node (router) and inspects traffic between the node and the internet. A distributed IDS is placed at the individual nodes that constitute the network [13].

7. **Hybrid Methods**: Hybrid Detection is a combination of Signature- and Anomaly-based methods, as well as mixed placement strategies, to counteract each other’s weaknesses

and create a stronger overall system. A good example of this is when an anomaly is detected, the pattern of the detected intrusion can be included into the dataset of a signature-based system [26]. Tabassum et al. [13] note that deep learning is a valuable tool in creating Hybrid Systems for use in IDS as it would otherwise be impractical to cater to all scenarios individually.

1.1.5 AI, ML and DL

A clear distinction should be made between the terms artificial intelligence, machine learning and deep learning as they are often used interchangeably but comprise different concepts. Figure 3 provides a graphical overview of these concepts.

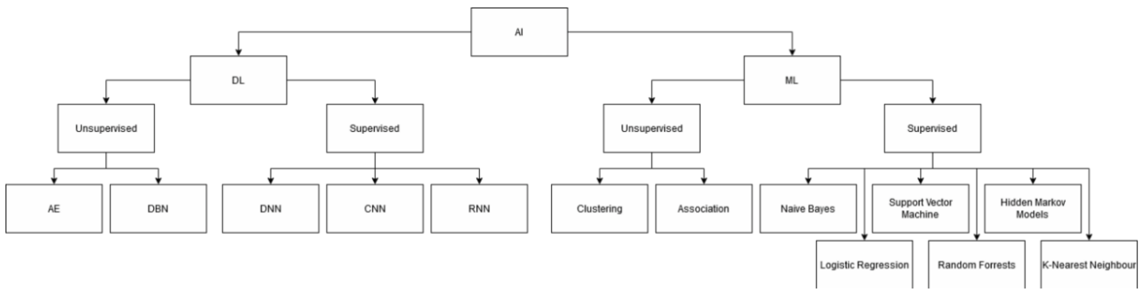


Figure 1.3: Machine- and Deep-Learning Algorithms

1. **Artificial Intelligence (AI)**: The exact definition of artificial intelligence has been a source of debate since nearly the inception of the term. A common definition is 'to make machines/computers think as humans do'. Even this is an extremely wide net cast over the field which has diverged into a great multitude of disciplines each with a different focus. The term AI will not be used in any specific capacity to refer to a set of algorithms or methods in this work. Instead, machine learning and deep learning will be considered, both of which are branches of AI.

2. **Machine Learning (ML)**: When referring to ML in this work, it will refer specifically to traditional, statistics-based algorithms such as Naïve Bayesian Classifiers (NB), Random Forest (RF), Support Vector Machines (SVM) etc. In other words, algorithms explicitly concerned with classification based on known features learned from training data [26]. ML is also referred to as shallow learning, in some cases to show the contrast between it and deep learning [27]. More examples of shallow learning algorithms can be seen in Figure 3.

3. **Deep Learning (DL)**: Deep Learning, as opposed to ML, refers specifically to the use of a neural network used to simulate the human brain. Examples of these algorithms include

Deep Belief Networks (DBN), Convolutional Neural Networks (CNN) and Long-Short Term Memory (LSTM). More examples of deep learning algorithms can be seen in Figure 3. Typically, deep learning models require a large set of data, does not require as much data pre-processing as ML but requires significantly more processing power, usually from a Graphics Processing Unit (GPU), and will take longer than most ML algorithms [26].

1.1.5.1 Supervised and Unsupervised Learning

Supervised learning refers to the use of labelled data to train models to classify data into defined groups or to accurately predict outcomes of given input. Significant pre-processing is usually required when implementing supervised learning as an appropriate 'feature vector' should be utilised to obtain the best results. Feature vectors are often generated through a 'feature selection' process or algorithm.

Unsupervised learning utilises unlabelled data to cluster similar datapoints, effectively generating its own classes. As there is less human intervention in the training process due to no labels being attached to the data, it is considered 'unsupervised'. These algorithms are often capable of finding correlations between data that may not be apparent to humans.

1.1.5.2 Application of Machine- and Deep-Learning in Security

From the previous sections, it is easy to envision where the evolving nature and flexibility provided by AI may be useful in cybersecurity applications. The dynamic nature of the field demands a system that can adapt in order for it to be acceptably secure. The adaptability of AI also allows for good integration with standard IDSs. For example, an AI decision engine can be used to determine if an anomaly in an IDS is malicious and if it is, add it to the data for a signature-based IDS. As such, most implementations of machine/deep learning are hybrid IDS’ [26].

1.1.6 Datasets

Along with the ever-increasing need for enhanced cybersecurity comes the need for increased research conducted in the field. For this, data is needed. As collecting data on a per-study basis is infeasible, the sheer number of datasets (and in some cases, their respective sizes) are astounding. Table 1.1 identifies some of the common datasets used in literature:

Table 1.1: Common Datasets

Dataset	**Year**	**Attack Types**	**Details**
NSL-KDD	2009	DoS, Probe, R2L and U2R	Created to solve some of the shortcomings of the KDD cup 99 set, namely: removing redundant and duplicate records and making the dataset more reasonable in terms of size [25], [28].
UNSW-NB15	2015	Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms	Contains real and synthetic network activity and threats [29].
CICDDoS-2019	2019	DDoS	Contains very recent attack methodologies [30].
CICIDS-2017	2017	Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS	Data was collected over a five-day period, generating 2,800,000 datapoints with 85 features [31].
CSE-CIC-IDS2018	2018	Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and insider infiltration	Created to solve many of the problems that pervade the field of IDS research (low dataset availability, out-of-date attack scenarios, privacy concerns etc.) [31].
Bot-IoT	2018	DDos, Dos, OS and Service Scan, Keylogging and Data exfiltration	Contains normal and bot-net traffic [32].

1.1.6.1 NSL-KDD Dataset

This dataset builds on the KDD Cup 99 dataset. It was created to solve some of the shortcomings of the KDD Cup 99 set, namely: removing redundant and duplicate records and making the dataset more reasonable in terms of size [25], [28]. Despite being an improved set, it is common for both the KDD Cup 99 and NSL-KDD sets to be used together when evaluating performance [33].

1.1.6.2 UNSW-NB15 Dataset

Created by the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS), this dataset contains real and synthetic network activity and threats. The 100 GB of captured data contains nine attack types (Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms) [29].

1.1.6.3 CICDDoS2019 Dataset

Created by the Canadian Institute for Cybersecurity (CIC), this dataset aims to address the shortcomings of other DDoS datasets as identified by the authors. A large focus of this dataset is that it contains very recent attack methodologies in an attempt to allow for the creation of equally modern detection algorithms [30].

1.1.6.4 CICIDS2017 Dataset

The Canadian Institute for Cybersecurity released this dataset in 2017. Focusing on network intrusion detection, the data was collected over a five-day period, generating 2,800,000 datapoints with 85 features from attacks such as Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS [31], [34].

1.1.6.5 CSE-CIC-IDS2018 Dataset

Created through a collaboration between the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC), this dataset was created to solve many of the problems pervading the field of IDS research (low dataset availability, out-of-date attack scenarios, privacy concerns etc.). The dataset (over 400 GB in size) contains seven attack scenarios (Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and insider infiltration) with over 80 features extracted [31].

1.1.6.6 Bot-IoT Dataset

This dataset was created by the Cyber Range Lab of The Center of UNSW Canberra Cyber. Containing normal and botnet traffic, the dataset is created from 69.3 GB of captured network traffic with the focus on IoT traffic. The types of attacks included in the dataset are: DDos, Dos, OS and Service Scan, Keylogging and Data exfiltration [32].

1.2 Research Objectives

This section details the current state of research available in literature where the use of AI-based Intrusion Detection Systems is considered for use in IoT networks and systems. The shortcomings are identified, and proposed work and improvements to address these shortcomings are introduced.

1.2.1 Current State of IoT Intrusion Detection Systems using AI

The use of AI for creating IDS' designed specifically for use in IoT networks is a field that is not as well explored as that of IDS' for traditional computer systems [35]. A key reason for this is the poor utilisation of appropriate datasets when designing IDS' for IoT [36]. Many authors prefer to use older, inappropriate datasets that often do not contain modern attack prototypes or IoT traffic as these older datasets are more widely studied and provide a 'benchmark'. As pointed out in earlier sections, the rapid evolution of the threat landscape makes the use of an appropriate dataset, such as the BoT-IoT dataset, an absolute necessity. Furthermore, it is reasonable to assume that even this dataset's worth will diminish over time and revisions and/or newer datasets will have to be generated.

The work in which appropriate, recent datasets are utilised is limited. Considering work in which the BoT-IoT dataset is used, it is commonly the case that the dataset is modified using oversampling techniques such as Synthetic Minority Oversampling Technique (SMOTE) [9] or extremely small subsets are used to balance the classes. There is very little work that considers the data as it was presented by its authors, Koroniotis et al. [32].

Literature regarding the use of AI-based IDS' mostly focusses on finding a single, 'best' model to detect and classify attacks [8]–[11], [32], [33], [35], [37]. Often this work is also limited in the range of algorithms considered. However, it is common to see that in many of these works, deep learning algorithms provide substantial time and classification metric benefits over machine learning methods. Guizani and Ghafoor [37] note the flexibility of neural networks and propose their use over traditional machine learning methods.

Clearly, the use of AI to defend and protect against threats in IoT networks is well supported in literature. Furthermore, the need for more advanced approaches in the form of distributed and combination-based systems has also been identified as a possibly advantageous solution [13]. An important observation of the work performed by Ferrag et al. [33] is that some algorithms are able to detect different attacks with better accuracy than others. This points to a possible layered solution, in which various algorithms can be leveraged in specific scenarios to provide optimal results. A similar conclusion is reached by Susilo and Sari [11], proposing a hybrid ML/DL system.

Very little work investigates layered approaches to this problem where the additional restrictions of the dataset and IoT environment are also considered. Khraisat et al. [38] use a C5 SIDS to separate known and unknown attacks. Unknown attacks are then classified as attack or normal traffic by a One Class SVM (OCSVM) at the second stage. Bovenzi et al. [39] instead uses their first stage to detect attack traffic and then classify the attack at the second stage. A modified Deep Autoencoder (DAE) is used at the first stage with a RF classifier at the second. Ullah and Mahmoud [40] follow a similar methodology to Bovenzi et al. but use a Decision Tree at the first stage. In all cases, the authors show that the proposed layered system provides significant benefit over a single-stage system.

The focus of the research in this dissertation is to propose and investigate a system that addresses the need for security in IoT networks while being applicable in resource-constrained environments. This system leverages the learning and adaptive nature of machine- and deep-learning methods to identify and classify malicious traffic in an IoT network. Specifically, seven (three machine learning and four deep learning) algorithms are evaluated for, and in, a two-stage system using the BoT-IoT dataset.

1.2.2 Proposed Work and Improvements

The work performed in this dissertation can be divided into three broad sections. The first is to investigate and compare the performance of various machine- and deep-learning models in different malicious traffic classification scenarios. Using the BoT-IoT dataset, the performance of algorithms and their relative strengths are investigated to gauge their suitability for the subsequent sections of work to be performed. This work places emphasis on deep learning methods due to their identified benefits over traditional machine learning methods.

Secondly, this work will design and propose a two-layer intrusion detection system to provide benefits in terms of time and classification metrics. The objective of this system is to meet the evolving security requirements of IoT networks and systems while remaining aware of the unique conditions and constraints of these networks and systems. This means identifying areas where redundancy can be reduced and considering the requirements at different stages of the system.

Finally, this work will investigate and evaluate the proposed system in terms of its efficacy and performance, using relevant metrics, such as time, F1-score, and false positive rate. Using the design from section two and the models implemented in section one, the proposed system can be fully realised and interrogated. This will explore the data flow through the system, the intricacies of its operation and the interactions of the stages with each other. The performance benefit over single-stage systems can then be examined in greater detail.

1.3 Contributions

The research conducted in the preparation of this dissertation has contributed directly to two research articles. The title, authors, article type, publication status and a brief summary of each article is provided.

Article 1

Title	Two-Stage IDS for IoT Using Layered Machine- and Deep-Learning Models
Authors	André van der Walt, Tahmid Quazi and Brett van Niekerk
Type	Journal Article
Status	Under Review at *Cyber-Physical Systems* (ISSN: 2333-5785)
Summary	An investigation of the applicability of various algorithms for use in a proposed two-stage system is presented. Using the BoT-IoT dataset, seven different (machine learning and deep learning) algorithms are applied. Similar works in literature have explored the application of augmentation techniques to balance the dataset as well as advanced feature selection techniques before applying the classification algorithms. Unlike such investigations, this work applies classification algorithms to an unmodified dataset and feature vector as proposed by the dataset authors. Furthermore, modifications of these algorithms' application, with the objective of obtaining improved efficiency, are explored along with their ability to classify in three scenarios: 1) binary attack/benign, 2) multi-class attack type with benign samples and 3) multi-class attack only. The modifications made to the application of the algorithms (where it varies from literature) are shown to have a positive effect on the results, achieving higher F1-scores than the four original Neural Network algorithms by an average of 22.75% and shorter training times relative to the base system by an average of 35.68 seconds. Furthermore, the potential benefits of an implemented two-stage system are examined, showing a potential reduction of threat detection/identification time of 0.51s on average and a potential increase of threat classification F1-score by 0.05 on average.

Article 2

Title	Investigating Machine- and Deep-Learning Model Combinations for a Two-Stage IDS for IoT Networks
Authors	André van der Walt, Tahmid Quazi and Brett van Niekerk
Type	Journal Article
Status	Under Review at *Sensors* (ISSN: 1424-8220)
Summary	Using the BoT-IoT dataset, the work in this paper investigates the performance of the algorithm combinations, layered in the two-stage system, as proposed in the previous work. Stage one determines if a network flow is malicious or benign. Malicious flows are then classified at stage two into DDoS, DoS, Scan or Theft attack categories. The first stage, binary classification performance of the algorithms is explored and compared in terms of classification and time metrics. The algorithms are then evaluated for multi-class classification of the attack types, and all two-stage combinations are tested. The results show a system consisting of a CNN binary classifier at stage one and a KNN 4-Class model at stage two performs best, beating the 5-Class system of either algorithm that classifies attack and benign classes. This system's first stage improves upon the 5-Class system's classification time by 0.25 seconds. Importantly, the benign class F1-score is improved by 0.23, indicating a significant improvement in false positive rate. The two-stage system achieves an overall F1-score of 0.94. The observed performance gains provided by using a two-stage system align broadly with the proposed benefits predicted in the previous work, showing the two-stage system would perform well as an IDS. Additionally, investigations arising from findings during the evaluation of the two-stage system are also discussed, namely GPU data-transfer overhead, the effect of data scaling and the effect of benign samples on stage two, providing a better understanding of how the BoT-IoT dataset interacts with AI models and how these models may be improved in future work.

References

[1] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, ‘Machine learning for Internet of Things data analysis: A survey’, *Digit. Commun. Netw.*, vol. 4, no. 3, pp. 161–175, Aug. 2018, doi: 10.1016/j.dcan.2017.10.002.

[2] M. R. Shahid, G. Blanc, Z. Zhang, and H. Debar, ‘IoT Devices Recognition Through Network Traffic Analysis’, in *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA, Dec. 2018, pp. 5187–5192. doi: 10.1109/BigData.2018.8622243.

[3] H. Nguyen-An, T. Silverston, T. Yamazaki, and T. Miyoshi, ‘IoT Traffic: Modeling and Measurement Experiments’, *IoT*, vol. 2, no. 1, pp. 140–162, Feb. 2021, doi: 10.3390/iot2010008.

[4] Palo Alto, ‘2020 Unit 42 IoT Threat Report’. Palo Alto, 2020. Accessed: Sep. 17, 2021. [Online]. Available: https://unit42.paloaltonetworks.com/iot-threat-report-2020/

[5] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, ‘DDoS in the IoT: Mirai and Other Botnets’, *Computer*, vol. 50, no. 7, pp. 80–84, 2017, doi: 10.1109/MC.2017.201.

[6] R. Calderon, ‘The Benefits of Artificial Intelligence in Cybersecurity’, *Econ. Crime Forensics Capstones*, Jan. 2019, [Online]. Available: https://digitalcommons.lasalle.edu/ecf_capstones/36

[7] G. Kumar and K. Kumar, ‘The Use of Artificial-Intelligence-Based Ensembles for Intrusion Detection: A Review’, *Appl. Comput. Intell. Soft Comput.*, vol. 2012, pp. 1–20, 2012, doi: 10.1155/2012/850160.

[8] C. Liang, B. Shanmugam, S. Azam, M. Jonkman, F. D. Boer, and G. Narayansamy, ‘Intrusion Detection System for Internet of Things based on a Machine Learning approach’, in *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, Mar. 2019, pp. 1–6. doi: 10.1109/ViTECoN.2019.8899448.

[9] Y. N. Soe, P. I. Santosa, and R. Hartanto, ‘DDoS Attack Detection Based on Simple ANN with SMOTE for IoT Environment’, in *2019 Fourth International Conference on Informatics and Computing (ICIC)*, Oct. 2019, pp. 1–5. doi: 10.1109/ICIC47613.2019.8985853.

[10] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, ‘Deep Learning-Based Intrusion Detection for IoT Networks’, in *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)*, Kyoto, Japan, Dec. 2019, pp. 256–25609. doi: 10.1109/PRDC47002.2019.00056.

[11] B. Susilo and R. F. Sari, ‘Intrusion Detection in IoT Networks Using Deep Learning Algorithm’, *Information*, vol. 11, no. 5, p. 279, May 2020, doi: 10.3390/info11050279.

[12] M. Shafiq, Z. Tian, Y. Sun, X. Du, and M. Guizani, ‘Selection of effective machine learning algorithm and Bot-IoT attacks traffic identification for internet of things in smart city’, *Future Gener. Comput. Syst.*, vol. 107, pp. 433–442, Jun. 2020, doi: 10.1016/j.future.2020.02.017.

[13] A. Tabassum, A. Erbad, and M. Guizani, ‘A Survey on Recent Approaches in Intrusion Detection System in IoTs’, in *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, Jun. 2019, pp. 1190–1197. doi: 10.1109/IWCMC.2019.8766455.

[14] P. Sethi and S. R. Sarangi, ‘Internet of Things: Architectures, Protocols, and Applications’, *J. Electr. Comput. Eng.*, vol. 2017, pp. 1–25, 2017, doi: 10.1155/2017/9324035.

[15] S. C. Ergen, ‘ZigBee/IEEE 802.15.4 Summary’, p. 37.

[16] N. Sastry and D. Wagner, ‘Security considerations for IEEE 802.15.4 networks’, in *Proceedings of the 2004 ACM workshop on Wireless security - WiSe ’04*, Philadelphia, PA, USA, 2004, p. 32. doi: 10.1145/1023646.1023654.

[17] S. Raza, S. Duquennoy, J. Höglund, U. Roedig, and T. Voigt, ‘Secure communication for the Internet of Things-a comparison of link-layer security and IPsec for 6LoWPAN: Secure communication for the Internet of Things’, *Secur. Commun. Netw.*, vol. 7, no. 12, pp. 2654–2668, Dec. 2014, doi: 10.1002/sec.406.

[18] G. Ee, C. K. Ng, N. Noordin, B. Mohd Ali, and Ali, ‘A Review of 6LoWPAN Routing Protocols’, *Proc. Asia-Pac. Adv. Netw.*, vol. 30, Aug. 2010, doi: 10.7125/APAN.30.11.

[19] S. Raza, L. Wallgren, and T. Voigt, ‘SVELTE: Real-time intrusion detection in the Internet of Things’, *Ad Hoc Netw.*, vol. 11, no. 8, pp. 2661–2674, Nov. 2013, doi: 10.1016/j.adhoc.2013.04.014.

[20] D. Gothawal and S. V. Nagaraj, ‘Intrusion Detection for Enhancing RPL Security’, *Procedia Comput. Sci.*, vol. 165, pp. 565–572, Jan. 2019, doi: 10.1016/j.procs.2020.01.051.

[21] B. Aziz, ‘On the Security of the MQTT Protocol’, 2016, p. 22.

[22] S. N. Sheela Evangelin Prasad, M. V. Srinath, and M. Saadique Basha, ‘Intrusion Detection Systems, Tools and Techniques – An Overview’, *Indian J. Sci. Technol.*, vol. 8, no. 35, Jan. 2015, doi: 10.17485/ijst/2015/v8i35/80108.

[23] ‘Insider threats and Insider Intrusion Detection’, *ijrte*, vol. 8, no. 2S5, pp. 158–166, Aug. 2019, doi: 10.35940/ijrte.B1033.0782S519.

[24] *2020 Insider Threat Report*. Accessed: Mar. 13, 2020. [Online]. Available: https://www.cybersecurity-insiders.com/portfolio/2020-insider-threat-report-gurucul/

[25] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, ‘Deep Learning Approach for Intelligent Intrusion Detection System’, *IEEE Access*, vol. 7, pp. 41525–41550, 2019, doi: 10.1109/ACCESS.2019.2895334.

[26] Y. Xin *et al.*, ‘Machine Learning and Deep Learning Methods for Cybersecurity’, *IEEE Access*, vol. 6, pp. 35365–35381, 2018, doi: 10.1109/ACCESS.2018.2836950.

[27] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, and M. Marchetti, ‘On the effectiveness of machine and deep learning for cyber security’, in *2018 10th International Conference on Cyber Conflict (CyCon)*, Tallinn, May 2018, pp. 371–390. doi: 10.23919/CYCON.2018.8405026.

[28] 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*, Ottawa, ON, Canada, Jul. 2009, pp. 1–6. doi: 10.1109/CISDA.2009.5356528.

[29] 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)*, Canberra, Australia, Nov. 2015, pp. 1–6. doi: 10.1109/MilCIS.2015.7348942.

[30] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, ‘Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy’, in *2019 International Carnahan Conference on Security Technology (ICCST)*, CHENNAI, India, Oct. 2019, pp. 1–8. doi: 10.1109/CCST.2019.8888419.

[31] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, ‘Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization’:, in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, Funchal, Madeira, Portugal, 2018, pp. 108–116. doi: 10.5220/0006639801080116.

[32] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, ‘Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset’, *ArXiv181100701 Cs*, Nov. 2018, Accessed: Sep. 30, 2020. [Online]. Available: http://arxiv.org/abs/1811.00701

[33] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, ‘Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study’, *J. Inf. Secur. Appl.*, vol. 50, p. 102419, Feb. 2020, doi: 10.1016/j.jisa.2019.102419.

[34] J. Lee, J. Kim, I. Kim, and K. Han, ‘Cyber Threat Detection Based on Artificial Neural Networks Using Event Profiles’, *IEEE Access*, vol. 7, pp. 165607–165626, 2019, doi: 10.1109/ACCESS.2019.2953095.

[35] J. Alsamiri and K. Alsubhi, ‘Internet of Things Cyber Attacks Detection using Machine Learning’, *Int. J. Adv. Comput. Sci. Appl. IJACSA*, vol. 10, no. 12, Art. no. 12, Jun. 2019, doi: 10.14569/IJACSA.2019.0101280.

[36] T. Singh and N. Kumar, ‘Machine learning models for intrusion detection in IoT environment: A comprehensive review’, *Comput. Commun.*, Feb. 2020, doi: 10.1016/j.comcom.2020.02.001.

[37] N. Guizani and A. Ghafoor, ‘A Network Function Virtualization System for Detecting Malware in Large IoT Based Networks’, *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1218–1228, Jun. 2020, doi: 10.1109/JSAC.2020.2986618.

[38] Khraisat, Gondal, Vamplew, Kamruzzaman, and Alazab, ‘A novel Ensemble of Hybrid Intrusion Detection System for Detecting Internet of Things Attacks’, *Electronics*, vol. 8, no. 11, p. 1210, Oct. 2019, doi: 10.3390/electronics8111210.

[39] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescape, ‘A Hierarchical Hybrid Intrusion Detection Approach in IoT Scenarios’, in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, Taipei, Taiwan, Dec. 2020, pp. 1–7. doi: 10.1109/GLOBECOM42002.2020.9348167.

[40] I. Ullah and Q. H. Mahmoud, ‘A Two-Level Flow-Based Anomalous Activity Detection System for IoT Networks’, *Electronics*, vol. 9, no. 3, p. 530, Mar. 2020, doi: 10.3390/electronics9030530.

Chapter 2

Journal Article 1

Two-Stage IDS for IoT Using Layered Machine- and Deep-Learning Models

André van der Walt, Tahmid Quazi, Brett van Niekerk

Under Review at *Cyber-Physical Systems*

2.1 Abstract

The ever-growing integration of Internet-of-Things (IoT) devices into our daily lives provides us with a level of convenience never seen before. However, as with most computer systems that deal with sensitive information, as IoT devices do, they must be secure. With billions of devices forming the IoT network, many low-powered and incorrectly configured, they are vulnerable to a myriad of attacks. The effects of attacks on these devices can be devastating. The discrete, low-powered nature of IoT devices makes their security a difficult problem to solve. The objective is to detect these threats before they can cause damage. To do this, we turn to machine-learning-based Intrusion Detection Systems (IDS) that can detect malicious traffic. This work proposes a two-stage detection and classification system using layered machine- and deep-learning models. An investigation of the applicability of various algorithms for this purpose is presented. Using the BoT-IoT dataset, seven different (machine learning and deep learning) algorithms are applied. Similar works in literature have explored the application of augmentation techniques to balance the dataset as well as advanced feature selection techniques before applying the classification algorithms. Unlike such investigations, this work applies classification algorithms to an unmodified dataset and feature vector as proposed by the dataset authors. Furthermore, modifications of these algorithms' application, with the objective of obtaining improved efficiency, are explored along with their ability to classify in three scenarios: 1) binary attack/benign, 2) multi-class attack type with benign samples and 3) multi-class attack only. The modifications made to the application of the algorithms (where it varies from literature) are shown to have a positive effect on the results, achieving higher F1-scores than the four original Neural Network algorithms by an average of 22.75% and shorter training times relative to the base system by an average of 35.68 seconds. Furthermore, the potential benefits of an implemented two-stage system are examined, showing a potential reduction of threat detection/identification time of 0.51s on average and a potential increase of threat classification F1-score by 0.05 on average. It is thus evident that a two-stage system such as the one proposed has potential benefit in the IoT context and is a viable tool to be explored further.

2.2 Introduction

It is estimated that there is anywhere between 25 and 50 billion IoT devices currently in operation [1]. As we introduce more and more devices into this connected web and become more reliant on them, we are entrusting these devices, and their creators, with increasingly concerning amounts of data. Ranging from personal health and well-being to administrative and financial data, we are becoming more connected not just with each other, but with our devices. Many say information is the new currency, usually followed by the phrase "if it's free, you are the product". Data theft, infiltration and disabling (or otherwise preventing these devices from functioning) can all have devastating consequences for individuals and companies using IoT devices.

A key aspect of IoT devices in many cases is to make them as low-powered and cheap as possible. This unfortunately also becomes one of the largest drawbacks of these devices in terms of security. With many security solutions being too 'heavy' for the IoT space, exploring any and all ways of improving computing efficiency of security solutions is paramount to ensuring viability. Lax security configuration, little-to-no standardisation and constant internet connectivity make IoT devices easy targets for threat actors. A proven favourite of these attackers are Distributed Denial-of-Service (DDoS) attacks. Attacks such as those based on the Mirai Botnet can be especially difficult to counter and have devastating consequences, being able to hit up to 1.1Tbps of traffic and sustained attacks for days on end [2]. Attacks based on the Mirai botnet are just one facet of a slew of challenges faced by these systems daily. These include information theft, keylogging, zero-days, etc.

Previous work has clearly shown the benefit of using artificial intelligence to detect these attacks [3]–[7]. The dynamic nature of the field demands a system that can adapt for it to be acceptably secure. The adaptability of AI also allows for good integration with standard IDSs. The application of machine learning/deep learning (ML/DL) in an IDS for the IoT environment is not as well explored as IoT's more fully featured computer counterparts [8].

2.3 Related Works

Calderon [9] explores the benefits of AI in cybersecurity, specifically in terms of IDPS (Intrusion Detection and Prevention Systems). Noting two specific problems, the use of Botnets to launch DDoS attacks and the large number of false positives generated by IDS's, Calderon shows that the adaptable, 'learning' nature of ML/DL solves these problems.

Kumar and Kumar [10] note that the rapid evolution of attacks leads to a lack of signatures of novel attacks to use with traditional Signature-based IDS's. This, along with the thousands of false positive alarms daily, present a massive challenge for network administrators. AI-based IDS techniques are proposed as a solution due to their "flexibility, adaptability, new pattern recognition, fault tolerance, learning capabilities, high computational speed, and error resilience for noisy data."

Lazic [11] looks at the real-world implications of using AI in IDS's. 64% of organisations reported a reduction in costs due to the use of AI to detect threats, with an average saving of 12%. Furthermore, time taken to detect threats and breaches was reduced by 12%. This shows the benefits of AI in threat detection extend beyond the academic, hypothetical realm.

IoT security is not a new topic and thus has several pieces of work dedicated to moving the field forward from which knowledge can be garnered. Tabassum et al. [12] investigate the recent approaches in creating appropriate IoT Threat Detection Systems. They note specifically the threat posed by DDoS attacks due to the large number of IoT Devices used today. They draw several very pertinent conclusions from their review. Chief among these is the need for an advanced, combinatorial, and distributed approach. They identify the use of ML/DL nearing a point of absolute necessity, however bringing with it a major challenge due to the resource intensive nature of its operation. Very importantly, they note that ML is unsuitable in some cases for detecting variations of attacks that DL may be able to detect, pointing out that DL may be the best direction in the future.

Vinayakumar et al. [13] explored the accuracy of a Deep Neural Network (DNN) on several datasets (KDDCup99, NSL-KDD, UNSW-NB15, WSN-DS, CICIDS2017, Kyoto), as well as comparing this to several ML algorithms (Logistic Regression, Naive Bayes, K-Nearest Neighbours, Decision Tree, AB, RF, SVM-rbf). The results show that different algorithms detect different attack methods with varying accuracy. Importantly, some cases of classical ML outperforming the DNN were observed. Again, this reinforces the idea that a one 'catch-all' solution does not exist, and that the optimal solution may rather be found by combining several algorithms. In general, it was found that the DNN outperformed the classical ML algorithms. They note the limitations found in many datasets which do not represent real-world traffic.

Liang et al. [3] look at the various hyperparameters of a Deep Neural Network (DNN) model used to classify IoT traffic. Although using the NSL-KDD dataset (which does not contain IoT Traces), the researchers found that an Adamax Optimizer and ReLU activations perform best.

Looking more specifically at work conducted with the BoT-IoT dataset: Ferrag et al. [14] utilised both the CSE-CIC-IDS2018 and Bot-IoT datasets. using multiple deep learning models. They classified these into two sets of models: deep discriminative models (Deep Neural Network, Recurrent NN, Convolutional NN) and generative/unsupervised models (Restricted Boltzmann Machine, Deep Belief Network, Deep Boltzmann Machine, Deep Autoencoder). The study expanded beyond just testing the overall accuracy of each dataset and included the accuracy of each model in detecting different attack types. This provides the very interesting conclusion, similar to [13], that different models are able to detect various attacks with different accuracy. This indicates that no one optimal algorithm will outperform all others. Rather, multiple layers of detection may be necessary to achieve the best results. They found that, generally, the Deep Autoencoder performed best in the IDS2018 dataset, and the CNN performed best in the BoT-IoT dataset.

Susilo and Sari [6] use the BoT-IoT dataset to examine the efficacy of various learning algorithms (Random Forest, Convolutional Neural Network, Multi-Layer Perceptron) and varying hyper-parameters (Batch size and epochs). The authors found that RF performed best overall and that, for large batch sizes, accuracy increases with epochs for the CNN and MLP. Noting this speed-up, they propose a hybrid system of ML and DL techniques.

Alsamiri and Alsubhi [8] investigate the implementation of machine learning algorithms for detecting cyber-attacks in an IoT context, explicitly. They use the BoT-IoT dataset's raw traffic files with features extracted using the CICFlowMeter tool and curated using a random forest regressor. Seven different ML algorithms are applied to classify the ten different attacks contained in the dataset. The results show that KNN performs best of all the algorithms with an F1-score of 0.99 but has a significantly longer training time than the other algorithms (nearly double that of the next slowest).

Guizani and Ghafoor, [15] noting the flexibility provided by neural networks over traditional machine learning algorithms, propose a Recurrent Neural Network Long-Short Term Memory (RNN-LSTM) Model for threat detection. Using the BoT-IoT dataset and the Keras Python Library, they perform dimensionality reduction on the feature vector and hyper-parameter tuning. The highest accuracy reported is 85% when classifying 10 classes.

Ge et al. [5] perform a comparison between a ML (Support-Vector Classifier) and DL (Feed-forward Neural Network) algorithm applied to the BoT-IoT Dataset. The authors note that using a sufficiently large sample size, the ML algorithm can outperform the DL algorithm in terms of accuracy metrics but does so at the cost of being "more than an order of magnitude slower" than the DL algorithm. They conclude that it is thus more efficient to make use of a DL algorithm.

Soe et al. [4] look at the Bot-IoT dataset's DDoS category. Noting the massive class support disparity, they use SMOTE to bolster the Benign Class to equal weight of the DDoS Class (477 Records to 1.9 million Records). They conclude that the imbalance in the dataset is a large problem in detection accuracy and F1-score, and that using SMOTE successfully reduces this problem, moving the F1-score from 0.99 to 1.00 on a simple Neural Network.

Koroniotis et al. [16] authored the BoT-IoT dataset and details some experiments on it in the paper accompanying the dataset. First identifying the shortcomings (chiefly the lack of IoT traces in most other datasets), the authors describe the data capture procedure, the features extracted by the Argus client program and generated features. The authors also explore the importance of dimensionality reduction. The ten most valuable features are identified, and they propose a 5% subset of the data to be used for training and testing purposes. Using the proposed 5% subset, the authors applied SVM, RNN and Long-Short Term Memory models for classification of the data. The authors achieve very good results with all three models (88% accuracy - worst case (SVM), 99% accuracy - best case (RNN and LSTM)). These models will serve as the base systems for the work performed in this paper.

Multi-stage AI systems used to detect attacks are not common, especially in IoT. Khraisat et al. [17] propose a two-stage model to detect threats using the BoT-IoT dataset. The first stage is a C5 SIDS phase used to identify known attacks, and to separate known attacks from unknown attacks. The second stage is a One-class Support-Vector Machine trained on benign samples to detect abnormal behaviour. Using Information Gain to select the feature set they obtained the following results: 0.957 F-measure and 94% accuracy in stage one, classifying the 5 attack classes. 0.927 F-measure and 92.5% accuracy in stage two, classifying benign and malicious traffic. The combined result gives an overall F-measure of 0.957 and 99.97% accuracy. It should be noted that the authors used a modified subset of the data to create more balanced classes, using only a few thousand entries as opposed to the few million contained in the dataset.

Ullah and Mahmoud [18] propose a two-level model to detect anomalous traffic. First detecting anomalous traffic and then determining the type of attack in levels one and two, respectively. Using the CICIDS2017 and UNSW-NB15 datasets as inputs to the model, and Recurrent Feature Elimination and Synthetic Minority Oversampling Technique to bolster the minor classes, the authors achieve a perfect F1-score in both binary datasets (using a decision tree) barring a 0.99 F1-score for UNSW-NB15 anomaly traffic. For multi-class (an Edited Nearest Neighbours algorithm), 1.00 and 0.97 F1-scores were achieved for CICIDS2017 and UNSW-NB15 respectively.

The same authors [19] later propose another two-layer system and evaluate it with the BoT-IoT dataset. The first layer classifies an input flow as anomalous or normal, while the second layer classifies the category or subcategory of a flow labelled as anomalous by the first layer. Notably, the second layer is trained to also classify benign flows. This is repeated work as benign and malicious flows were already classified in the first layer. A decision tree was used for layer one and a RF classifier for layer two. The authors achieve a 99.99 F1-score for the Binary classification and an average of 99.80 and 98.80 for 5-Class (Category) and 11-Class (subcategory) F1-score. The work performed by these authors does not look at any other algorithms or possible combinations. No deep learning models are considered either. Thus, one of the aims of this work will be to examine a wider variety of algorithms, both machine- and deep-learning.

As pointed out by Singh and Kumar [20], there is a tendency in the field to use outdated and unrealistic datasets in the name of 'benchmarking'. Considering the rate of progress in this and related fields, this is untenable. It is for this reason that datasets such as BoT-IoT are important to this and future work, especially with respect to IoT threat detection. This work utilises the BoT-IoT dataset to ensure that it is representative of modern attack vectors and traffic signatures.

The work of Koroniotis et al. [16] in which the dataset is described, and from which the base system of this work is derived, is limited in terms of its application of ML/DL models to the dataset and can be expanded upon. Considering the work performed with the BoT-IoT dataset in literature, several authors make use of re-sampling or other augmentation techniques to balance out the classes of the dataset instead of using the original class weights of the dataset.

It is uncommon to see work in literature that focusses on the applicability of DL methods when investigating IDS’ for IoT networks. Many papers will identify them as important avenues for exploration in identifying means of securing systems, but they are rarely investigated in the IoT context. This is even more so the case when two-stage systems are considered literature [17]–[19]. This work includes four deep learning models in its investigation.

The performance metrics used to measure the efficacy of such an unbalanced dataset are also extremely important. For example, the 'accuracy' metric becomes much less important if the classes are extremely imbalanced as is the case with the BoT-IoT dataset. The 'recall', 'precision' and 'F1-score' metrics would in this case be much more representative of performance.

Additionally, understanding the computational requirements of a system designed for potentially resource-constrained devices is also important. This work evaluates the performance of the investigated algorithms not only on their F1-score but also considers the associated time metrics to give insight into the relative computational performance of the considered algorithms. Table 2.1 summarises the limitations of closely related works in literature, namely those that look at multi-stage IDS’ for use in IoT.

Table 2.1: Limitations of Related Work

Related Work	**Summary**	**Limitations**
Khraisat et al. [17]	This work proposes a two-stage system with a C5 SIDS at the first	The chief limitation of this work stems from its lack of consideration

	stage followed by a OCSVM at the second. Attacks are first separated into known and unknown signatures and then classified further at the second stage. The system achieves a 0.957 F-Score.	of other algorithms for use in the two-stage system. No deep learning models are considered, and significant emphasis is placed on the 'accuracy' metric. No computational or time metrics are considered when judging performance.
Ullah and Mahmoud [18]	The authors propose a two-layer system in which traffic is first determined to be benign or malicious after which malicious traffic is classified into attack types. The authors do not use the BoT-IoT dataset but achieve a 1.00 and 0.97 F-score in two other recent datasets.	A major shortcoming of this work is not using a dataset containing IoT traces despite aiming to develop a system for IoT networks. Furthermore, only two algorithms are considered. Time or computation metrics are also not considered.
Ullah and Mahmoud [19]	The authors here extend their idea from the previous work, utilising the BoT-IoT dataset. A DT is used at the first stage with a RF classifier at stage 2. The system achieves 0.9999, 0.9980 and 0.9880 F-scores for binary, 5-class and 11-class classification respectively.	While an appropriate dataset is used here, as opposed to [19], it is again the case that only two algorithms are investigated, neither of which are deep learning models. Similar to [19], time and computation metrics are again not considered.

This work contributes to the literature by investigating the applicability of seven different machine- and deep-learning algorithms in a proposed two-stage system for detecting and classifying attacks in the IoT environment. The algorithms are applied to the unmodified and unaugmented BoT-IoT dataset as described by the dataset authors to maintain the original class weights. The ability of these algorithms to classify the binary attack/benign traffic as well as 5-Class attack and benign traffic and 4-Class attack-only traffic is also considered. Modifications are made to the application of the algorithms to the dataset and their effects are interrogated, seeking performance improvements in terms of time and classification metrics. This investigation will provide insight to the feasibility of a proposed, two-stage system.

The contribution of this paper is therefore as follows:

- Proposing a two-stage intrusion detection system is for an IoT network.

- Evaluating the applicability of various algorithms (ML and DL) for a two-stage detection system.
- Expanding the work performed with the BoT-IoT dataset by evaluating various ML/DL algorithms on an unmodified and unaugmented BoT-IoT dataset in different classification scenarios.

The paper, following this point, is organised as follows: Section 3 presents the proposed system. Section 4 details the experimental setup, dataset and tools used as well as a discussion of some choices made before commencing with the work. Section 5 documents the experimental work performed, and the results obtained at the various stages of the experiment, showing a logical flow between the sections as the work progressed. Section 6 is used to discuss the results obtained in the previous section. Section 7 is the conclusion, followed by recommendations for future work.

2.4 Proposed System

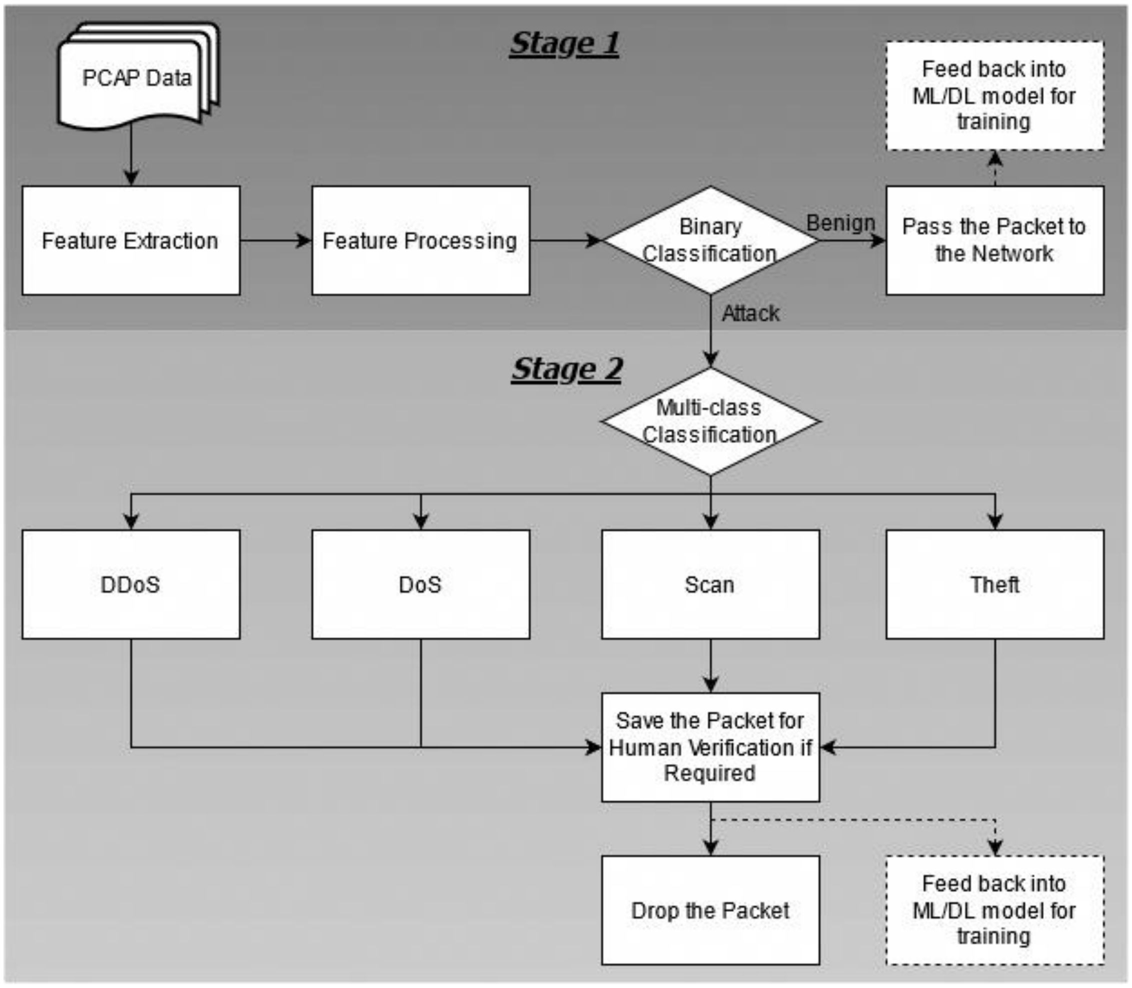

Stage 1
PCAP Data
Feature Extraction
Feature Processing
Binary Classification
Benign
Pass the Packet to the Network
Feed back into ML/DL model for training
Attack
Stage 2
Multi-class Classification
DDoS
DoS
Scan
Theft
Save the Packet for Human Verification if Required
Drop the Packet
Feed back into ML/DL model for training

Figure 2.1: Proposed Two-Stage System UML Diagram

The proposed system in Figure 2.1 shows the flow of data through the system. The first stage performs the data clean-up and preparation of the packet capture data for the binary classification. The output of this stage is a binary classification of 'Benign' or 'Attack'. If the traffic is 'Benign', the packet is permitted and passed on to the next step in the network. If the traffic is an 'Attack', it is passed on to the second stage of the system.

At the second stage, a multi-class classification is performed. Here the flow can be classified as DDoS, DoS, Scan or Theft attacks. Since the flow is already known to be malicious at this stage, the system does not have the possibility to classify traffic as 'Benign' as the system proposed by [19] does. This should result in a more efficient system as work is not repeated. After the classification, the packet can be saved for later review and training and dropped from the network.

Beyond the operation of the system, the continuous learning of such a system would be paramount to its success and maintained reliability. The dashed lines and boxes in Figure 2.1 show branches where flows may be fed back into the underlying models to train on new data. This can be based on the labels assigned at stage one and two for unsupervised learning or admin assigned/approved labels from review of the packets for supervised learning.

2.4.1 Stage Positions

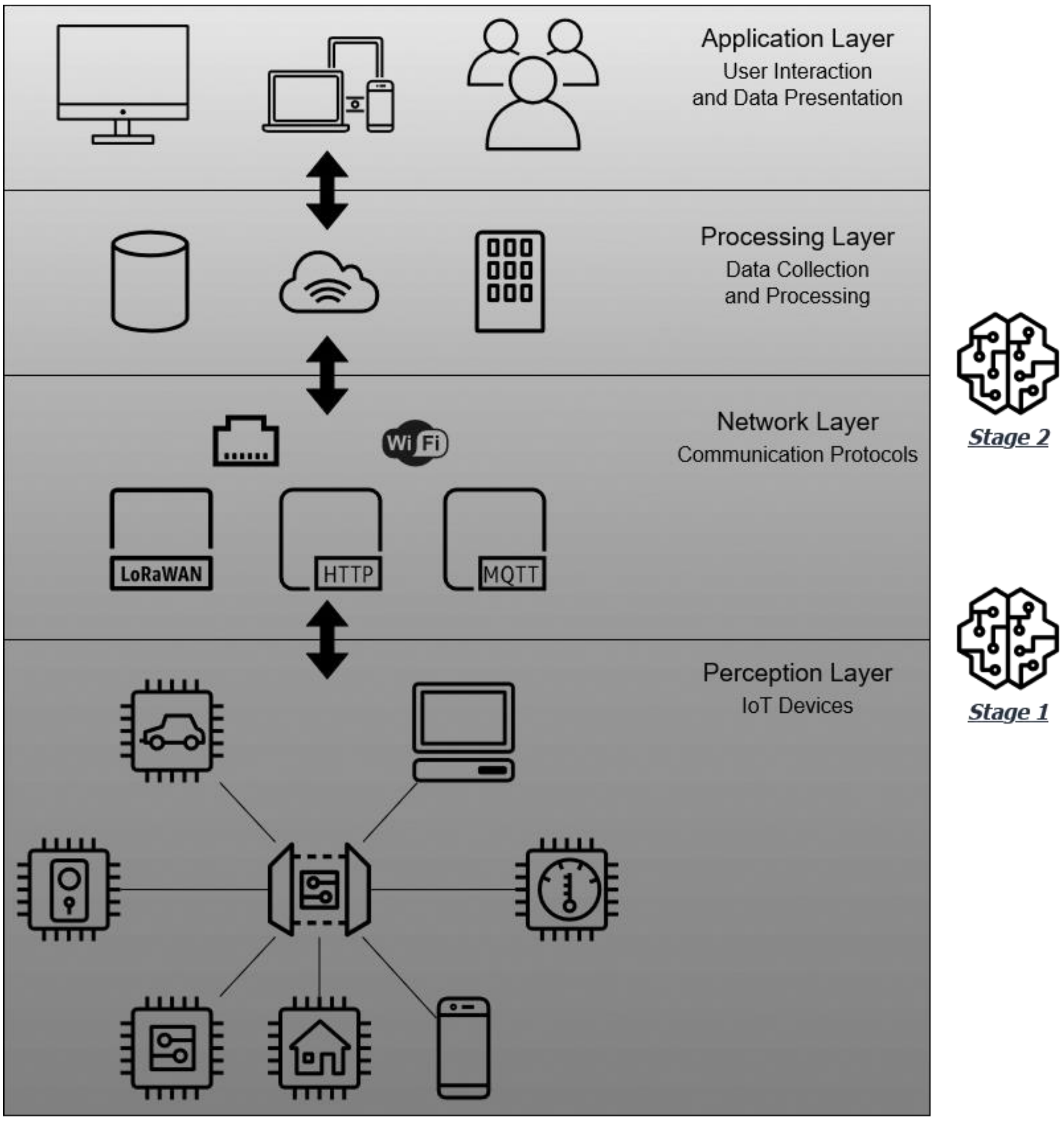


Figure 2.2: Proposed Two-Stage System in IoT Network

Figure 2.2 illustrates the location of the stages of the proposed system in the IoT stack. The first stage is placed at the network layer as the incoming flows are inspected by the system before being passed or dropped. This stage can either be physically located on a host IoT device or a gateway device depending on the network conditions and compute capability of the devices in the network. The second stage exists further up the stack, where the data collected by the system can be further classified. This second stage would likely be implemented on a dedicated IDS server or a more powerful gateway node.

2.5 Experimental Setup

2.5.1 BoT-IoT Dataset

For this work, the BoT-IoT dataset was selected as it is both very recent and contains the IoT traces necessary to create a system designed specifically for the IoT environment. Kororniotis et al., the authors of the dataset, further propose a 5% (3 million records, 1.07 GB) subset of the data to allow for easier handling [16]. Further to this, the ten most important features are selected by the dataset authors.

Table 2.2: BoT-IoT 10 Best Features [16]

Feature	**Description**
Seq	Argus sequence number
Stddev	Standard deviation of aggregated records
N_IN_Conn_P_SrcIP	Number of inbound connections per source IP
Min	Minimum duration of aggregated records
State_number	Numerical representation of feature state
Mean	Average duration of aggregated records
N_IN_Conn_P_DstIP	Number of inbound connections per destination IP
Drate	Destination-to-source packets per second
Srate	Source-to-destination packets per second
max	Maximum duration of aggregated records

Table 2.3: Description of the 5% BoT-IoT Subset

Class		**Count**		**Sum**
		Train	**Test**	
Benign		370	107	477
Attack	DDoS	1 541 315	385 309	1 926 624

	DoS	1 320 148	330 112	1 650 260
	Reconnaissance	72 919	18 163	91 082
	Theft	65	14	79
Total		2 934 817	733 705	3 668 522

This 5% subset, using the proposed 10-best features, is used in this paper.

2.5.1.1 Imbalance

A crucially important feature of the dataset, paramount to its usage, is that it is severely imbalanced. Attack traffic constitutes 99.99% of the dataset and of this attack traffic, DDoS and DoS attacks comprise a high percentage of this attack traffic. This imbalance makes the dataset difficult to utilise without balancing during pre-processing.

2.5.1.2 Pre-processing

The pre-processing of the 5% subset of data was very light as it merely required some data cleaning and enumeration. However, an important decision taken in this work is to not augment the dataset as many other researchers have done. While the appeal of a balanced dataset is clear, and in many cases beneficial, the objective of this work is to examine the capability of the various algorithms on the dataset as it was presented. Thus, no balancing techniques such as SMOTE, were used.

2.5.2 Tools

All experiments were conducted on a Windows 10 PC. The PC has the following specifications:

- CPU: Intel Core i7-10700k (8C/16T) @ 4.9GHz
- Memory: 64GB DDR4-3200 CL16
- GPU: Nvidia GeForce RTX 2070 Super

To create the experimental environment and models, the Anaconda Python 3 Distribution was used. For all deep-learning models, Keras (with a TensorFlow 2 backend) was used. For machine-learning, Scikit-Learn was used.

2.5.3 Performance Evaluation

Due to the imbalance of the dataset, F1-score will be used as the main evaluation metric derived from the Confusion Matrices. Generally, this metric is most important in cybersecurity applications [21].

Table 2.4: Confusion Matrix

	Predicted Positive	**Predicted Negative**
Labelled Positive	True Positive (TP)	False Negative (FN)
Labelled Negative	False Positive (FP)	True Negative (TN)

F1-score is given by $2 * TP/(2 * TP + FN + FP)$, this is the harmonic mean of precision and recall.

Additionally, the performance of each system must be evaluated in terms of its computational requirements. As these models are implemented on a test system and considering the wide variety of devices for which such a system may be used, computational requirements may be evaluated through examination of time metrics. By comparing time metrics when different models are run on identical hardware, we gain an insight into relative performance of the models to one another.

2.6 Comparison and Investigation of Algorithms for the Two-Stage System

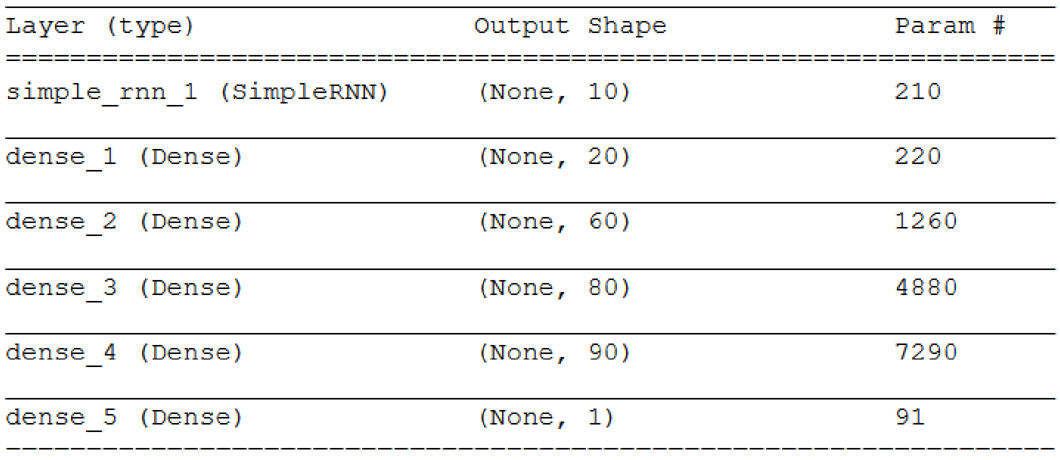
The objective of this section is to determine the viability of, and compare, various machine- and deep-learning algorithms for the two-stage system to detect attacks from a dataset with IoT traces proposed in Section 3. The work of Alsamiri and Alsubhi [8] and the work of Koroniotis et al. [16] are of particular interest in this case. From the results of these papers, specific algorithms and implementation steps were selected to be replicated, as well as additional algorithms to be introduced to expand upon this work.

Koroniotis et al. [16] discuss the intricacies of their proposed BoT-IoT dataset, exploring the significant features from the extracted data, a proposed standard training and testing subset and three algorithms used to evaluate the threat detection of each, using the dataset. The pre-processing steps, model architectures and parameters are also presented. Alsamiri and Alsubhi [8] similarly perform this work, focussing on different machine learning algorithms.

The proposed system consists of binary classification followed by a more granular multi-class classification. As such, the models are tested in three classification scenarios. The binary scenario looks only at attack or benign classification. The 4-Class seeks only to classify the attack types and is not classifying any benign traffic. The 5-Class model discriminates between DDoS, DoS, Scans, Theft and Benign Traffic. The 5-Class model serves as the base from which the other models are built as it allows for comparisons to be made to related work as well as a comparison to the two-stage system.

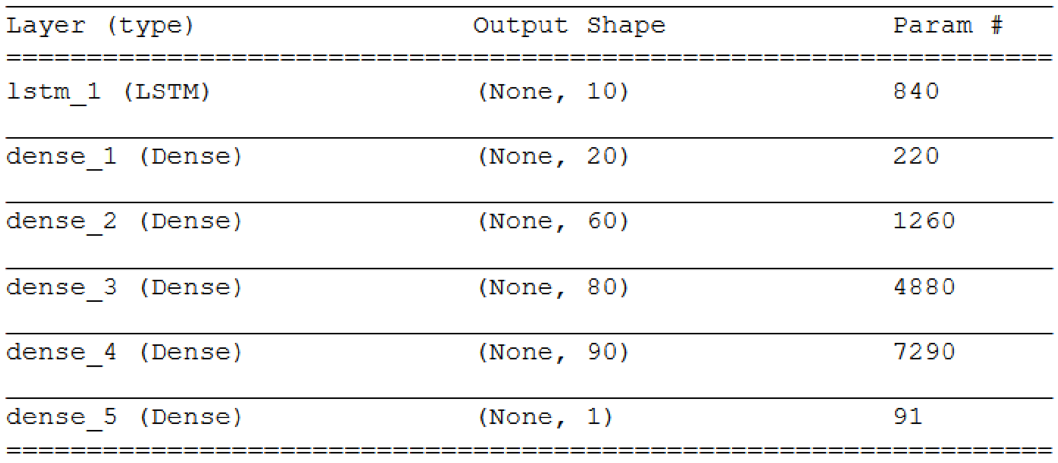
2.6.1 Replication and Verification

As a starting point to the planned experiment, it was decided to replicate the implementations of Koroniotis et al [16]. Using the proposed 5% subset and the identified 10 best features, the SVM, RNN and LSTM networks were replicated.



```
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_1 (SimpleRNN)     (None, 10)                210
_________________________________________________________________
dense_1 (Dense)              (None, 20)                220
_________________________________________________________________
dense_2 (Dense)              (None, 60)                1260
_________________________________________________________________
dense_3 (Dense)              (None, 80)                4880
_________________________________________________________________
dense_4 (Dense)              (None, 90)                7290
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 91
-----------------------------------------------------------------
```

Figure 2.3: RNN Binary Classification Network



```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 10)                840
_________________________________________________________________
dense_1 (Dense)              (None, 20)                220
_________________________________________________________________
dense_2 (Dense)              (None, 60)                1260
_________________________________________________________________
dense_3 (Dense)              (None, 80)                4880
_________________________________________________________________
dense_4 (Dense)              (None, 90)                7290
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 91
=================================================================
```

Figure 2.4: LSTM Binary Classification Network

Figures 2.3 and 2.4 show the replicated architecture implementation of the Binary Classifiers. The results obtained by running these models and the SVM are documented along with their multi-class counterparts. The confusion matrices of the Binary Classifiers are also given. This can be seen in Tables 2.5 and 2.6(a-c):

Table 2.5: Replication (Epochs = 4, Batch = 100) Results

Algorithms	Accuracy	F1-score	Weighted F1-score	Training Time	Classification Time
RNN Bin	99.9850%	0.61	1.00	331.26s	11.22s
LSTM Bin	99.9879%	0.72	1.00	363.31s	12.82s
SVM Bin	99.9862%	0.55	1.00	2.93s	0.05s
RNN 5-Class	97.3418%	0.81	0.97	326.85s	11.44s
LSTM 5-Class	97.3538%	0.75	0.97	358.56s	12.94s
SVM 5-Class	83.6295%	0.42	0.83	138.31s	0.04s

Table 2.6: Replication Binary Classification Confusion Matrices

True\Predict	Normal (0)	Attack (1)
Normal (0)	15	92
Attack (1)	18	733580

(a) RNN Binary Classification Confusion Matrix

True\Predict	Normal (0)	Attack (1)
Normal (0)	35	72
Attack (1)	17	733581

(b) LSTM Binary Classification Confusion Matrix

True\Predict	Normal (0)	Attack (1)
Normal (0)	6	101
Attack (1)	0	733598

(c) SVM Binary Classification Confusion Matrix

The model architectures described by Koroniotis et al. [16] for the neural nets are well documented. However, some choices such as the activation functions for the hidden and output layers are not discussed and seem arbitrarily chosen. Interestingly, the confusion matrices used to represent the classification performance of the models show 477 benign samples in the test set. This implies that the training samples were used in testing as well as the total of the benign samples in both the training and testing 5% subsets is 477. This would significantly skew the results and invalidate them. As can be seen from Tables 2.5 and 2.6(a-c), the results obtained through replication of the work performed by the dataset authors yield significantly different results but does show a similar trend in terms of model performance relative to the other algorithms implemented.

Note in Table 2.5 the 'F1-score', 'Training Time' and 'Classification Time' columns as these will be the main points of comparison used throughout the paper. Table 2.6(a-c) shows the confusion matrices of the different algorithms applied to the binary classification scenario. The binary classification confusion matrices will also be used for comparison throughout the work performed. It is clear that LSTM performs best here with the highest true positive rate for normal traffic and smallest false positive rate for attack traffic. This is also seen in the work of Koroniotis et al. [16] where their implementation of an LSTM network also outperformed (in terms of F1-score using the 10-best feature vector) the other models, with SVM performing the worst.

2.6.2 Modification

Following the results obtained during the replication stage, it was decided to investigate the discrepancy by means of finding the parameters needed to attain the accuracy reported by the authors [16].

2.6.2.1 Modification: Categorical Target Vector (Epochs = 4, Batch = 100)

The first attempt was to experiment specifically with the way the target vector was handled by the binary models. This was done by changing the class vector of integers into a binary class matrix. The results of these changes are shown in Tables 2.7 and 2.8(a-b).

Table 2.7: Categorical Target Vector Modification (Epochs = 4, Batch = 100) Results

Algorithm	**Accuracy**	**F1-score**	**Weighted F1-score**	**Training Time**	**Classification Time**
RNN (Bin)	99.9848%	0.61	1.00	324.23s	11.66s
LSTM (Bin)	99.9868%	0.76	1.00	365.82s	12.77s

Table 2.8: Categorical Target Vector Modification Confusion Matrices

True\Predict	**Normal (0)**	**Attack (1)**
Normal (0)	15	92
Attack (1)	19	733579

(a) RNN Binary Classification Confusion Matrix

True\Predict	**Normal (0)**	**Attack (1)**
Normal (0)	52	55
Attack (1)	42	733556

(b) LSTM Binary Classification Confusion Matrix

Table 2.7 illustrates this change did not yield significant benefit, only showing a small improvement in the F1-score of the LSTM network and a small decrease in training time for the RNN. Table 2.8(a) shows no changes outside margin of error, while Table 2.8(b) shows an improvement in the TP rate of normal traffic and a worse FP rate in attack traffic compared to the base models. This is still, however, a net improvement compared to the replicated models.

2.6.2.2 Modification: Activation Functions (Epochs = 4, Batch = 100)

The next modification examines the activation functions used at the various layers. More specifically, the activation functions of the input and hidden layers were changed from tanh to ReLU, and the output layer activation function was changed from sigmoid to SoftMax. The change of activation function was also applied to the models which had already been modified in the previous section. The results are recorded in Tables 2.9 and 2.10(a-d):

Table 2.9: Activation Function Modification (Epochs = 4, Batch = 100) Results

Algorithms	**Accuracy**	**F1-score**	**Weighted F1-score**	**Training Time**	**Classification Time**
RNN Bin	99.9854%	0.50	1.00	341.92s	11.56s
LSTM Bin	99.9854%	0.50	1.00	368.43s	12.83s
RNN Bin (Cat)	99.9898%	0.78	1.00	338.32s	12.19s
LSTM Bin (Cat)	99.9893%	0.73	1.00	364.66s	13.42s
RNN 5-Class	97.1103%	0.84	0.97	309.21s	12.07s
LSTM 5-Class	95.7923%	0.66	0.96	347.14s	13.41s

Table 2.10: Activation Function Modification Confusion Matrices

True\Predict	**Normal (0)**	**Attack (1)**
Normal (0)	0	107
Attack (1)	0	733598

(a) RNN Binary Classification Confusion Matrix

True\Predict	**Normal (0)**	**Attack (1)**
Normal (0)	0	107
Attack (1)	0	733598

(b) LSTM Binary Classification Confusion Matrix

True\Predict	**Normal (0)**	**Attack (1)**
Normal (0)	47	60
Attack (1)	15	733583

(c) RNN Binary Classification Confusion Matrix (Categorical)

True\Predict	**Normal (0)**	**Attack (1)**
Normal (0)	32	75
Attack (1)	3	733595

(d) RNN Binary Classification Confusion Matrix (Categorical)

These modifications returned some notable results. First, looking at the binary classification in Table 2.9, it can be seen that when the target vector has not been modified, the models fail to classify the non-attack traffic. However, when the target vector is changed into a binary class matrix, an improvement in the F1-score of the RNN is observed. A slight dip in the F1-score of the LSTM is also noted.

For the multi-class classifiers, a slight increase in the F1-score of the RNN and a decrease in the F1-score of the LSTM network are noted. However, this is also accompanied by a decrease in training time.

Tables 2.10(a) and 2.10(b) show that the use of the new activation function made the models predict all traffic as attack traffic. Conversely, Tables 2.10(c) and 2.10(d) show that the ReLU activation, combined with a categorical target vector, provides a net improvement in TP and FP rates relative to the replicated models.

2.6.3 Additional Algorithms and Attack-only Classification

Based on the results of the previous section, it was decided to continue with the modified version of the models, both regarding the target vector and activation functions. The next step was to start tuning the hyper-parameters of these models. The tuning focused mainly on the batch size and number of epochs.

From the results of [8], it was decided to also implement a RF and KNN on this data. In addition, a Convolutional Neural Network and a Deep Neural Network were also introduced. The decision to implement these models are supported by the results of [14] and [6] for CNN, and [13] and [3] for DNN.

From this, the 4-Class model was implemented (the second stage classifier). This system ignores benign traffic and only looks at the four attack classes.

The final hyperparameter options were 100 epochs at a batch size of 5000. As a matter of comparison, the model architecture was maintained and not made any deeper or shallower. The results can be found in Tables 2.11 and 2.12(a-g):

Table 2.11: Final Implementation Results

Algorithms	**Accuracy**	**F1-score**	**Weighted F1-score**	**Training Time**	**Classification Time**
DNN Bin	99.9985%	0.97	1.00	281.06s	9.44s
CNN Bin	99.9974%	0.96	1.00	294.75s	14.13s
RNN Bin	99.9981%	0.97	1.00	288.87s	12.13s
LSTM Bin	99.9961%	0.94	1.00	307.42s	13.29s
SVM Bin	99.9862%	0.55	1.00	2.93s	0.05s
RF Bin	99.9898%	0.73	1.00	250.44s	2.49s
KNN Bin	99.9989%	0.98	1.00	1695.29s	55.04s
DNN 5-Class	97.5711%	0.93	0.98	257.11s	10.52s
CNN 5-Class	97.7132%	0.94	0.98	293.07s	14.94s
RNN 5-Class	97.5243%	0.94	0.98	333.48s	12.41s
LSTM 5-Class	97.4660%	0.95	0.97	307.47s	13.07s
SVM 5-Class	83.6295%	0.42	0.83	138.31s	0.04s
RF 5-Class	92.2151%	0.56	0.92	261.08s	3.86s

KNN 5-Class	99.0624%	0.97	0.99	1683.04s	55.37s
DNN 4-Class	97.7151%	0.97	0.98	252.47s	10.43s
CNN 4-Class	97.8938%	0.97	0.98	289.71s	14.73s
RNN 4-Class	97.6148%	0.97	0.98	286.00s	12.57s
LSTM 4-Class	97.4794%	0.96	0.97	302.17s	13.25s
SVM 4-Class	84.8457%	0.54	0.85	139.09s	0.05s
RF 4-Class	90.2681%	0.69	0.90	259.91s	3.59s
KNN 4-Class	99.0785%	0.98	0.99	1448.17s	53.89s

Table 2.12: Final Implementation Confusion Matrices

True\Predict	**Normal (0)**	**Attack (1)**
Normal (0)	101	6
Attack (1)	5	733593

(a) DNN Binary Classification Confusion Matrix

True\Predict	**Normal (0)**	**Attack (1)**
Normal (0)	98	9
Attack (1)	10	733588

(b) CNN Binary Classification Confusion Matrix

True\Predict	**Normal (0)**	**Attack (1)**
Normal (0)	102	5
Attack (1)	9	733589

(c) RNN Binary Classification Confusion Matrix

True\Predict	**Normal (0)**	**Attack (1)**
Normal (0)	105	2
Attack (1)	26	733572

(d) LSTM Binary Classification Confusion Matrix

True\Predict	**Normal (0)**	**Attack (1)**
Normal (0)	6	101
Attack (1)	0	733598

(e) SVM Binary Classification Confusion Matrix

True\Predict	**Normal (0)**	**Attack (1)**
Normal (0)	32	75
Attack (1)	0	733598

(f) RF Binary Classification Confusion Matrix

True\Predict	**Normal (0)**	**Attack (1)**
Normal (0)	104	3
Attack (1)	5	733593

(g) KNN Binary Classification Confusion Matrix

2.7 Results Discussion

Table 2.13: Metric Improvements: Modified vs Replicated

Algorithm	F1-score Improvement	Training Time Improvement	Classification Time Improvement
RNN Bin	0.36	42.39s	-0.91s
LSTM Bin	0.22	55.89s	-0.47s
SVM Bin	0.00	0.00s	0.00s
RNN 5-Class	0.13	-6.64s	-0.97s
LSTM 5-Class	0.20	51.09s	-0.13s
SVM 5-Class	0.00	0.00s	0.00s

The final set of results in Tables 2.8 and 2.9(a-g), and summarised in Table 2.13, clearly shows that modifications performed (using a categorical target vector, ReLu and SoftMax activation functions, and increased batch sizes and epochs) throughout the course of the experiment yielded significantly better results. By using a SoftMax output function paired with a categorical target vector (for which SoftMax is usually applied), improvement is observed as is to be expected in a categorical situation. Furthermore, the use of ReLU (which is less computationally intensive than tanh) allows for more epochs without significant penalty to training and classification times.

Disregarding the SVM model, which saw no change, the other two models showed clear improvements in F1-score. Similarly, training time decreased substantially in all cases, barring a small increase in training time for the one-stage 5-Class RNN. Considering the substantially larger number of epochs, this is very good. The main negative aspect of the obtained results lies in the longer classification time, albeit almost negligible amounts. Again, considering the gains in F1-score and training time, this appears to be a worthwhile trade-off.

As noted previously, the imbalanced nature of the data makes the F1-score the most important metric in determining the efficacy of a model. Using this, paired with training time, some interesting observations become apparent when considering the additional algorithms introduced.

Firstly, the k-Nearest Neighbours algorithm with k=3 performed the best of all the algorithms. Unfortunately, this comes at a significant performance penalty. KNN takes five-and-a-half times longer than the next longest-running algorithm for binary classification. A similar ratio is true for the 5- and 4-Class systems. Naturally, a balance should be sought between computation time and accuracy. Fortunately, the DNN implementation in all cases performs similarly but is able to complete much faster. The accuracy result is echoed by the confusion matrix of the DNN when comparing it to that of the KNN implementation.

It should also be noted that the architecture of the models as proposed by Koroniotis et al. [16], and modified in this paper, are unorthodox in terms of the composition of the hidden layers. It is possible that more complex models may be able to perform even better than these modified versions.

Table 2.14: Proposed Two-Stage System Benefits

Algorithm	**Classification Time Benefit**	**F1-score Benefit**
DNN	1.08s	0.04
CNN	0.6s	0.02
RNN	0.28s	0.03
LSTM	-0.04s	-0.01
SVM	-0.01s	0.13
RF	1.37s	0.17
KNN	0.33s	0.01

(a) Stage one: Classification Time Benefits: Binary vs 5-Class

Algorithm	**Classification Time Benefit**	**F1-score Benefit**
DNN	0.09s	0.04
CNN	0.21s	0.03
RNN	-0.16s	0.03
LSTM	-0.18s	0.01
SVM	-0.01s	0.12
RF	0.27s	0.13
KNN	1.48s	0.01

(b) Stage one: Classification Time Benefits: 4-Class (attack-only) vs 5-Class

Table 2.14 (a-b) shows the proposed performance benefits of the two-stage system when compared to a single-stage system. At stage one, binary classification is performed to minimise the load on the IoT device that the stage is deployed on. Thus, the classification time benefit is of highest import here. Table 2.14(a) illustrates that there is indeed a noticeable decrease in time required to classify for most algorithms. This indicates reduced computational load, meaning this stage could be used in lower compute-power devices and conditions. Further to this, there is also a F1-score benefit observed. This would potentially reduce the number of false positives generated by the two-stage system compared to a system that classifies attack and benign traffic (5-Class).

Table 2.14(b), similarly, looks at the potential benefits of stage two. Here, the emphasis is on accurate classification of the attack type and classification speed is less important. An improvement in the F1-score of the 4-Class attack-only classifier is observed when compared to the attack and benign 5-Class system. This indicates a potential improvement in the proposed system as a whole to accurately classify flows when compared to a single-stage 5-Class model.

The promising results here make a strong case for the proposed system to be considered for deployment as an efficient IDS for practical IoT implementations.

2.8 Conclusion

In this paper, a two-stage system is proposed and considered for an IoT network. This two-stage system offers time and classification metric improvements when compared to a single-stage detection and classification system. Furthermore, we examine the algorithms applicable to such a system. By applying these algorithms in multiple scenarios and with various modifications, a "best" algorithm for each scenario can be identified. Notably, the use of an unmodified (unbalanced) BoT-IoT dataset to allow for the traffic to be as close as possible to that which was recorded is useful for simulating a real-world environment. First, the work of Koroniotis et al. [16] was replicated as a base and point of comparison. The results of that implementation prompted the modification of the implementation. These modifications included experimenting with activation functions and target vector composition. Finally, once optimal modifications had been found, tuning was conducted on the epochs and batch size to optimise for F1-score and training/classification time.

The final results showed the KNN implementation with k=3 performed best in terms of F1-score but took significantly longer to train than the other models. The next best-performing models (considering F1-score and time metrics) were Neural Network implementations (Binary - DNN, 5-Class - LSTM, 4-Class - CNN), all with similar training and classification times. Pair this with evidence from the literature, and Neural Networks are shown to be the best approach for threat detection.

The results obtained in the investigation of these algorithms - when viewed through the lens of the proposed two-stage system - are promising. The proposed two-stage system would result in a decreased load on the devices on which the first stage is deployed, as it is shown that the time (and thus, computation) requirements of the classification is reduced by 0.51s on average. Furthermore, increased accuracy is expected when classifying attacks in the second stage as F1-score gains of 0.05 on average are observed when compared to a 5-Class attack and benign system.

The work performed in this paper is not without limitations. Firstly, it was performed on a test system which is not necessarily representative of a system on which the IDS as proposed may be implemented. It is also possible that different and/or better results may be achieved if the models used are implemented and tuned differently (different hyperparameters, different architectures etc.). Finally, the potential benefits of implementing the two-stage system as described are theoretical as the system is not implemented at this stage.

This work only focusses on a small aspect of a much larger goal which is creating safer, more robust IoT networks. Future work should study the performance of an integrated two-stage system using the machine- and deep-learning models proposed in this paper. Additional work can also be done to further enhance the benefits to be had at the individual stages of such a system by further reducing computation requirements of the first stage and further improving the classification capability of the second stage.

References

[1] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, ‘Machine learning for Internet of Things data analysis: A survey’, *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, Aug. 2018, doi: 10.1016/j.dcan.2017.10.002.

[2] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, ‘DDoS in the IoT: Mirai and Other Botnets’, *Computer*, vol. 50, no. 7, pp. 80–84, 2017, doi: 10.1109/MC.2017.201.

[3] C. Liang, B. Shanmugam, S. Azam, M. Jonkman, F. D. Boer, and G. Narayansamy, ‘Intrusion Detection System for Internet of Things based on a Machine Learning approach’, in *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, Mar. 2019, pp. 1–6. doi: 10.1109/ViTECoN.2019.8899448.

[4] Y. N. Soe, P. I. Santosa, and R. Hartanto, ‘DDoS Attack Detection Based on Simple ANN with SMOTE for IoT Environment’, in *2019 Fourth International Conference on Informatics and Computing (ICIC)*, Oct. 2019, pp. 1–5. doi: 10.1109/ICIC47613.2019.8985853.

[5] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, ‘Deep Learning-Based Intrusion Detection for IoT Networks’, in *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)*, Kyoto, Japan, Dec. 2019, pp. 256–25609. doi: 10.1109/PRDC47002.2019.00056.

[6] B. Susilo and R. F. Sari, ‘Intrusion Detection in IoT Networks Using Deep Learning Algorithm’, *Information*, vol. 11, no. 5, p. 279, May 2020, doi: 10.3390/info11050279.

[7] M. Shafiq, Z. Tian, Y. Sun, X. Du, and M. Guizani, ‘Selection of effective machine learning algorithm and Bot-IoT attacks traffic identification for internet of things in smart city’, *Future Generation Computer Systems*, vol. 107, pp. 433–442, Jun. 2020, doi: 10.1016/j.future.2020.02.017.

[8] J. Alsamiri and K. Alsubhi, ‘Internet of Things Cyber Attacks Detection using Machine Learning’, *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 10, no. 12, Art. no. 12, Jun. 2019, doi: 10.14569/IJACSA.2019.0101280.

[9] R. Calderon, ‘The Benefits of Artificial Intelligence in Cybersecurity’, *Economic Crime Forensics Capstones*, Jan. 2019, [Online]. Available: https://digitalcommons.lasalle.edu/ecf_capstones/36

[10] G. Kumar and K. Kumar, ‘The Use of Artificial-Intelligence-Based Ensembles for Intrusion Detection: A Review’, *Applied Computational Intelligence and Soft Computing*, vol. 2012, pp. 1–20, 2012, doi: 10.1155/2012/850160.

[11] L. Lazic, ‘Benefit From AI In Cybersecurity’, Oct. 2019.

[12] A. Tabassum, A. Erbad, and M. Guizani, ‘A Survey on Recent Approaches in Intrusion Detection System in IoTs’, in *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, Jun. 2019, pp. 1190–1197. doi: 10.1109/IWCMC.2019.8766455.

[13] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, ‘Deep Learning Approach for Intelligent Intrusion Detection System’, *IEEE Access*, vol. 7, pp. 41525–41550, 2019, doi: 10.1109/ACCESS.2019.2895334.

[14] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, ‘Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study’, *Journal of Information Security and Applications*, vol. 50, p. 102419, Feb. 2020, doi: 10.1016/j.jisa.2019.102419.

[15] N. Guizani and A. Ghafoor, ‘A Network Function Virtualization System for Detecting Malware in Large IoT Based Networks’, *IEEE J. Select. Areas Commun.*, vol. 38, no. 6, pp. 1218–1228, Jun. 2020, doi: 10.1109/JSAC.2020.2986618.

[16] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, ‘Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset’, *arXiv:1811.00701 [cs]*, Nov. 2018, Accessed: Sep. 30, 2020. [Online]. Available: http://arxiv.org/abs/1811.00701

[17] Khraisat, Gondal, Vamplew, Kamruzzaman, and Alazab, ‘A novel Ensemble of Hybrid Intrusion Detection System for Detecting Internet of Things Attacks’, *Electronics*, vol. 8, no. 11, p. 1210, Oct. 2019, doi: 10.3390/electronics8111210.

[18] I. Ullah and Q. H. Mahmoud, ‘A Two-Level Hybrid Model for Anomalous Activity Detection in IoT Networks’, in *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan. 2019, pp. 1–6. doi: 10.1109/CCNC.2019.8651782.

[19] I. Ullah and Q. H. Mahmoud, ‘A Two-Level Flow-Based Anomalous Activity Detection System for IoT Networks’, *Electronics*, vol. 9, no. 3, p. 530, Mar. 2020, doi: 10.3390/electronics9030530.

[20] T. Singh and N. Kumar, ‘Machine learning models for intrusion detection in IoT environment: A comprehensive review’, *Computer Communications*, Feb. 2020, doi: 10.1016/j.comcom.2020.02.001.

[21] Y. Xin *et al.*, ‘Machine Learning and Deep Learning Methods for Cybersecurity’, *IEEE Access*, vol. 6, pp. 35365–35381, 2018, doi: 10.1109/ACCESS.2018.2836950.

Chapter 3

Journal Article 2

Investigating Machine- and Deep-Learning Model Combinations for a Two-Stage IDS for IoT Networks

André van der Walt, Tahmid Quazi, Brett van Niekerk

Under Review at *Sensors*

This article has been modified from the version currently under review, which is in US English, to UK English for consistency with the rest of this dissertation. Other minor typesetting changes were also done.

3.1 Abstract

Rapidly evolving, novel attacks and large false positive rates make securing IoT devices, that are already resource constrained in many situations, a significant challenge for network administrators. AI can provide tailored security to the unique conditions and threats found in IoT networks. Previous work investigated the applicability of seven different ML and DL algorithms for use in a two-layer IDS system. This work investigates the performance of various algorithm combinations of the two-stage system. Stage one determines whether network traffic is malicious, and stage two classifies malicious traffic into DDoS, DoS, Scan or Theft categories. The performance is evaluated for the stage one binary classification, stage two multi-class classification, and then all algorithm combinations for the multi-layer system. The results show that a CNN binary classifier at stage one and a KNN 4-Class model at stage two performs best. Malicious traffic classification time is improved by 0.25 seconds and the benign class F1-score is improved by 0.23, indicating significantly improved false positive rates. The system achieves an overall F1-score of 0.94 which, along with the other metrics, shows that the system would perform well as an IDS and improves in the crucial areas of attack detection and false positive rates over a single-stage AI system. Additional investigations arising from findings are presented, namely GPU data-transfer overhead, data scaling impact and the effect of benign samples on stage two, giving insight into how the dataset interacts with AI models and how these models may be improved in future work.

3.2 Introduction

The world has seen large-scale adoption of Internet-of-Things (IoT) devices over the last couple of years, resulting in record amounts of traffic moving across global networks with an estimated 79.4 Zettabytes of data expected to be generated by upwards of 75 billion devices by 2025 [1]–[3]. This trend is expected to proceed into the future and for IoT to become an increasingly pervasive part of daily life, business, health etc. Their rapid adoption has made them a massive target for attackers and bad actors. Palo Alto Unit 42 [4] in its 2020 IoT Threat Report identifies that 98% of IoT device traffic is unencrypted with 57% of devices vulnerable to severe attacks. It further identifies a lack of security systems designed for IoT as existing systems are designed for conventional computer systems but neglect custom-Operating System (OS) devices.

IoT devices, because of their extremely wide array of use-cases and varying implementations make standardization of protocols for these devices very difficult, especially in terms of security. This is compounded by their, in many cases, low compute ability. These circumstances allow catastrophic and, often, far-too-easy breaches in security. Perhaps the most infamous attack in this regard are attacks based on the Mirai Botnet, resulting in Distributed Denial-of-Service (DDoS) events capable of generating 1.1 Tbps of traffic and sustained attacks for days on end [5]. This is just one of the many threats faced by IoT devices. Palo Alto [4] identifies exploit attacks (scans, zero-days, SQL injection etc.) as the main threat to IoT devices. It notes a move towards worms that can propagate through the network as the new preferred method of attack.

Along with Botnet DDoS attacks, the large number of false positives generated by traditional Intrusion Detection Systems (IDS) and rapid attack evolution are a challenge for network administrators [6], [7]. Previous work has shown the use of artificial intelligence (AI) to be effective in detecting these attacks [8]–[12]. The need to use AI in IoT for various applications has been recognized for some time now. It is evident that this must be applied to the security of these devices as well [13]. Any system designed for this purpose must acceptably account for the wide variety of possible implementations and network conditions as well as the capability of the devices that constitute that network and the unique threats faced.

3.3 Related Works

Previous work in literature has examined how to best defend against and prevent these threats, leveraging the capabilities of learning approaches.

Koroniotis et al. [14], the authors of the BoT-IoT dataset, identify the shortcomings in most other datasets as the lack of IoT traces. The authors describe the data capture procedure, the features extracted by the Argus client program and generated features that constitute the BoT-IoT dataset.

After identifying the ten most "valuable" features, they propose a 5% subset of the data to be used for training and testing purposes. Using this subset, Support Vector Machine (SVM), Recurring Neural Network (RNN) and Long-Short Term Memory (LSTM) models are evaluated. The authors achieve very good results with all three models: 88% accuracy with the SVM and 99% accuracy with the RNN and LSTM models.

Alsamiri and Alsubhi [15] interrogate the use of machine learning (ML) algorithms for detecting threats in an IoT environment. They use the BoT-IoT dataset's raw traffic files and extract features using CICFlowMeter. They apply seven ML algorithms to classify ten different attack categories. It is shown that K-Nearest Neighbors (KNN) performs best in terms of F1-score, achieving a 0.99 score, but takes significantly longer in time metrics (almost twice as slow as the next slowest algorithm).

Hussain et al. [16] seek to address shortcomings in exiting solutions' attack detection rate. The authors here propose a two-stage system, consisting of a SVM anomaly detection first stage with an Artificial Neural Network (ANN) misuse detection second stage, evaluated with the NSL-KDD [17] dataset. The authors find that the two-stage system outperforms single stage systems, with a 99.95% detection accuracy and 0.2% false positive rate.

Kaja et al. [18] attempt to address the problem of excessive false positives common in many IDS'. The authors propose a two-stage detection system and evaluate using the KDDCUP99 [19] dataset. The first stage detects the presence of an attack using unsupervised learning, with the second stage classifying the attack using supervised learning. A K-Means algorithm is employed at the first stage with multiple ML algorithms tested at the second stage. The authors identify the J48 algorithm as the best performer with no false positives and a 0.999 F1-score.

Khraisat et al. [19] propose a two-stage model, using a C5 Signature-IDS (SIDS) first stage to separate known and unknown attacks. The second stage is a One-Class SVM anomaly detection model. Using the BoT-IoT dataset and Information Gain feature selection, a 0.957 F-measure and 94% accuracy is achieved at stage one, classifying the 5-attack classes. A 0.927 F-measure and 92.5% accuracy is achieved in stage two, classifying benign and malicious traffic. The combined result gives an overall F-measure of 0.957 and 99.97% accuracy. The authors used data balancing techniques to address the imbalance in the dataset, using only a few thousand entries as opposed to the few million contained in the dataset.

Ullah and Mahmoud [20] propose a two-stage system, detecting anomalies at stage one and then classifying the attack type at stage two. The authors use the CICIDS2017 [21] and UNSW-NB15 [22] datasets, Recurrent Feature Elimination and Synthetic Minority Oversampling Technique to address imbalance. The authors achieve a perfect F1-score for binary classification in nearly all cases using a decision tree, barring a 0.99 F1-score for UNSW-NB15 anomaly traffic. Using an Edited Nearest Neighbors algorithm for multi-class classification, 1.00 and 0.97 F1-scores were achieved for CICIDS2017 and UNSW-NB15 respectively.

Bovenzi et al. [23] propose a two-stage hierarchical Network-IDS (NIDS) which is then evaluated with the BoT-IoT dataset. The first stage detects anomalous traffic with the second classifying any identified attacks. The anomaly detecting first stage leverages a modified Deep Autoencoder, dubbed M2-DAE. In the second stage, Random Forest (RF), Naive Bayes (NB) and Multi-Layer Perceptron (MLP) are investigated. It was found that Random Forest performed best at the second stage, yielding a system F1-score of 0.9761.

Ullah and Mahmoud [24] again propose a two-layer system to detect and classify attack traffic. The BoT-IoT dataset is used here for evaluation. The first layer performs binary classification of anomalous or normal flows, with the second layer classifying the category or subcategory of flows identified as anomalous. The second layer is, however, trained to also classify benign traffic where this work was already performed at the first stage. Stage one uses a decision tree for binary classification, achieving a 99.99 F1-score. The second stage uses a RF classifier and achieves an average of 99.80 and 98.80 for 5-Class (Category) and 11-Class (subcategory) F1-scores.

Van der Walt et al. [25] propose a two-stage system, detecting attacks at stage one and classifying them into categories at stage two. The authors explore seven ML and DL algorithms, looking at their ability to classify in binary (attack/benign), 4-Class (attack classes only) and 5-Class (attack classes and benign) scenarios to determine their viability for use in the proposed system. The BoT-IoT dataset is used. The authors found that, aside from KNN, which was removed for having excessive time penalties when classifying, the best performing algorithms for binary, 4-Class and 5-Class were Deep Neural Network (DNN), LSTM and Convolutional Neural Network (CNN) respectively. The authors here stop short of testing the integrated model and draw conclusions from the individual results only. Potential F1-score improvements of 0.05 on average and time metric improvements of 0.51s on average are shown to be possible, using the suggested algorithms in the proposed model.

Table 3.1: Comparison of Related Works

Work	**IoT IDS**	**BoT-IoT Dataset**	**ML**	**DL**	**Multi-layer**	**Notes**
Koroniotis et al. [14]	X	X	X	X		
Alsamiri and Alsubhi [15]	X	X	X			
Hussain et al. [16]			X	X	X	
Kaja et al. [18]			X		X	

Khraisat et al. [19]	X	X	X		X	
Ullah and Mahmoud [20]	X		X		X	
Bovenzi et al. [23]	X	X	X		X	
Ullah and Mahmoud [24]	X	X	X		X	
Van der Walt et al. [25]	X	X	X	X	X*	*This work does not investigate an integrated system and only considers the theoretical benefits of a two-stage system.
This Work	X	X	X	X	X	

Related works that investigated the use of AI-based IDS' for IoT networks and systems are often only single-stage systems and do not consider the potential benefits of layering multiple algorithms [14,15]. The work of this paper will investigate how these algorithms can be utilised collaboratively to produce a superior system compared to current state-of-the-art single-stage systems.

Much of the work investigating these two-stage systems does not consider their applicability in the IoT environment, despite the potential advantages in such computationally constrained devices. When these systems are investigated for this purpose, work is often conducted with inappropriate datasets for the types of threats commonly faced by IoT networks [6,14] or only one or two algorithms are investigated, rarely including Deep Learning (DL) algorithms among those. The work presented in this paper addresses this by investigating seven different algorithms - three machine learning algorithms and four deep learning algorithms.

The research of Ullah and Mahmoud [24], which inspired this work, is limited in the algorithms considered. The authors consider only one algorithm at each stage and no deep learning algorithms are evaluated while there is clear evidence of their benefits over traditional and ML methods [13]. Furthermore, Ferrag et al. [26] show that different algorithms are able to classify various attack types with differing accuracy, presenting the case for a wider array of algorithms to be considered when such a system is designed. This paper considers seven algorithms, investigating all possible combinations in the two-stage system, to identify a best-case system.

The system proposed by Ullah and Mahmoud repeats work at the second layer that was already done in the first. Their second stage classifier is also allowed to classify a flow as benign traffic. Given a sufficiently accurate first stage classifier, this work can be avoided and allows the second stage system to be less restricted in terms of time metrics. This allows the system more time to classify attack types as all attack traffic at this stage would have been identified and need not pass to the rest of the network. The system examined in this work foregoes this redundant operation at the second stage for the sake of lessening the computational requirements of the system where possible.

The previous work of this paper's authors [25] describes a proposed system that addresses some of the identified shortcomings in other works. Using the BoT-IoT dataset, the proposed system performs binary (attack/benign) classification at the first stage and only classifies attack types at the second stage. Additionally, seven algorithms (4 DL, 3 ML) are suggested and investigated for applicability. The aim of this paper is to investigate the practicalities of implementing the system and testing algorithm combinations where they were not considered in [25].

This work contributes to literature by testing the effectiveness of a proposed two-stage IDS in an IoT network. Necessary steps between the system's stages are first identified, followed by homogenizing the test environment and re-running some of the experiments done in [25] to give an accurate comparison of their performance and ability. A discussion of the processing overhead introduced by data transfer to a Graphics Processing Unit (GPU) (as done in [25]) follows, demonstrating the time metric benefits observed. A total of 49 different algorithm combinations are then tested (both ML and DL algorithms) to identify an optimal system composition. An investigation into observed misclassifications is then conducted, identifying the second-stage models' sensitivity to benign samples missed by the first-stage classifier. It is shown that the proposed system offers benefits in threat detection and a lowered false positive rate relative to previous systems described in literature.

The contributions of this paper are therefore as follows:

- Evaluating and testing a two-stage system for detecting and classifying attacks in an IoT network using both ML and DL algorithms.
- Investigating data-scaling and the data-transfer overhead associated with using the BoT-IoT dataset for GPU-enhanced DL training and testing.
- Investigating the effect of benign samples on models designed to classify attack classes.

The paper, following this point, is organised as follows: Section 3 presents a description of the implemented system. Section 4 details the experimental setup, dataset and tools used, as well as a discussion of some choices made before commencing with the work. Section 5 documents the experimental work performed, and the results obtained at the various stages of the experiment. Section 6 is used to discuss the results obtained in the previous section. Section 7 is the conclusion, followed by recommendations for future work.

3.4 System Description

The system in Figure 3.1 illustrates the logical flow of data through the system. The first stage is responsible for sorting benign traffic from attack traffic. The second stage classifies malicious traffic identified in the first stage further into sub-categories to identify the attack type. These classes are DDoS, DoS, Scan or Theft attacks. The flow is already assumed to only contain malicious traffic and thus the second stage is not trained to classify benign flows again as seen in the system proposed by [24]. After the classification, the packet can be saved for later review and training, and then dropped from the network.

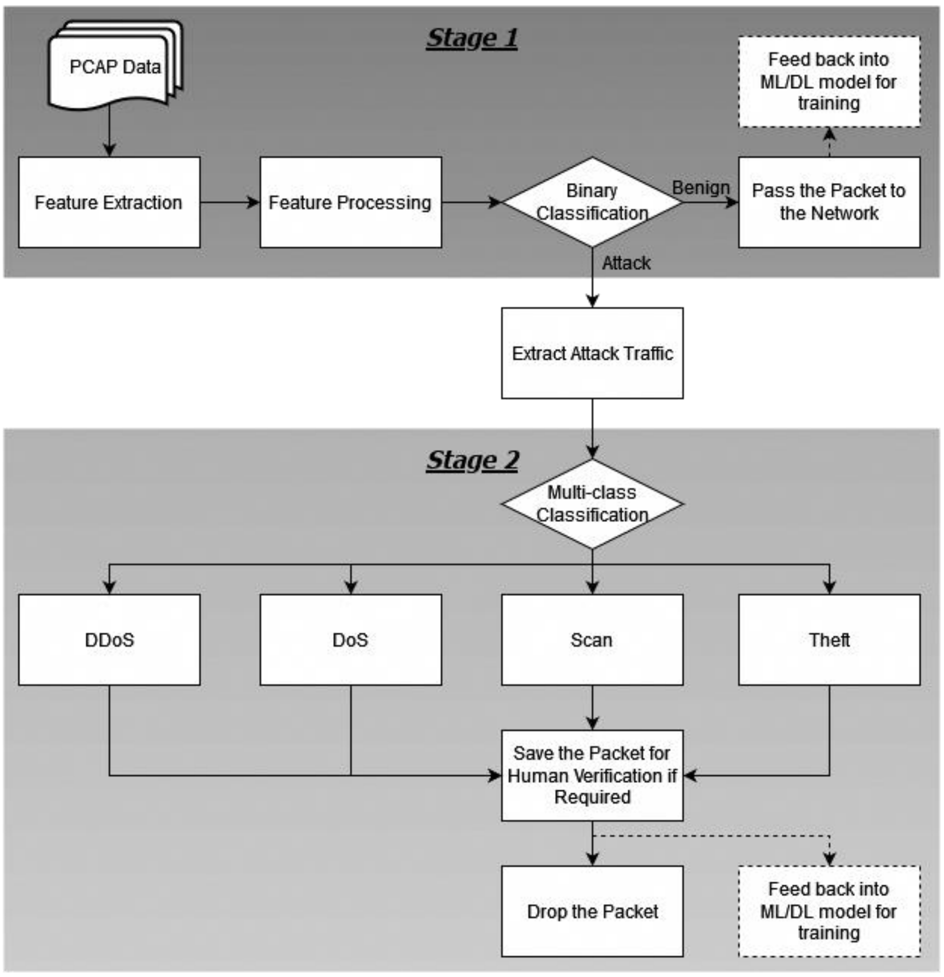

Stage 1
PCAP Data
Feed back into ML/DL model for training
Feature Extraction
Feature Processing
Binary Classification
Benign
Pass the Packet to the Network
Attack
Extract Attack Traffic
Stage 2
Multi-class Classification
DDoS
DoS
Scan
Theft
Save the Packet for Human Verification if Required
Drop the Packet
Feed back into ML/DL model for training

Figure 3.1: Proposed Two-Stage System UML Diagram adapted from [25]

Figure 3.2 shows the possible locations of the system stages in the IoT stack. Stage one is placed at the network layer either on a host IoT device or a gateway device. The second stage is located on a dedicated IDS server or a more powerful gateway node. It is evident that there are many instances in which such a system may be deployed in an environment where compute performance and network bandwidth are limited.

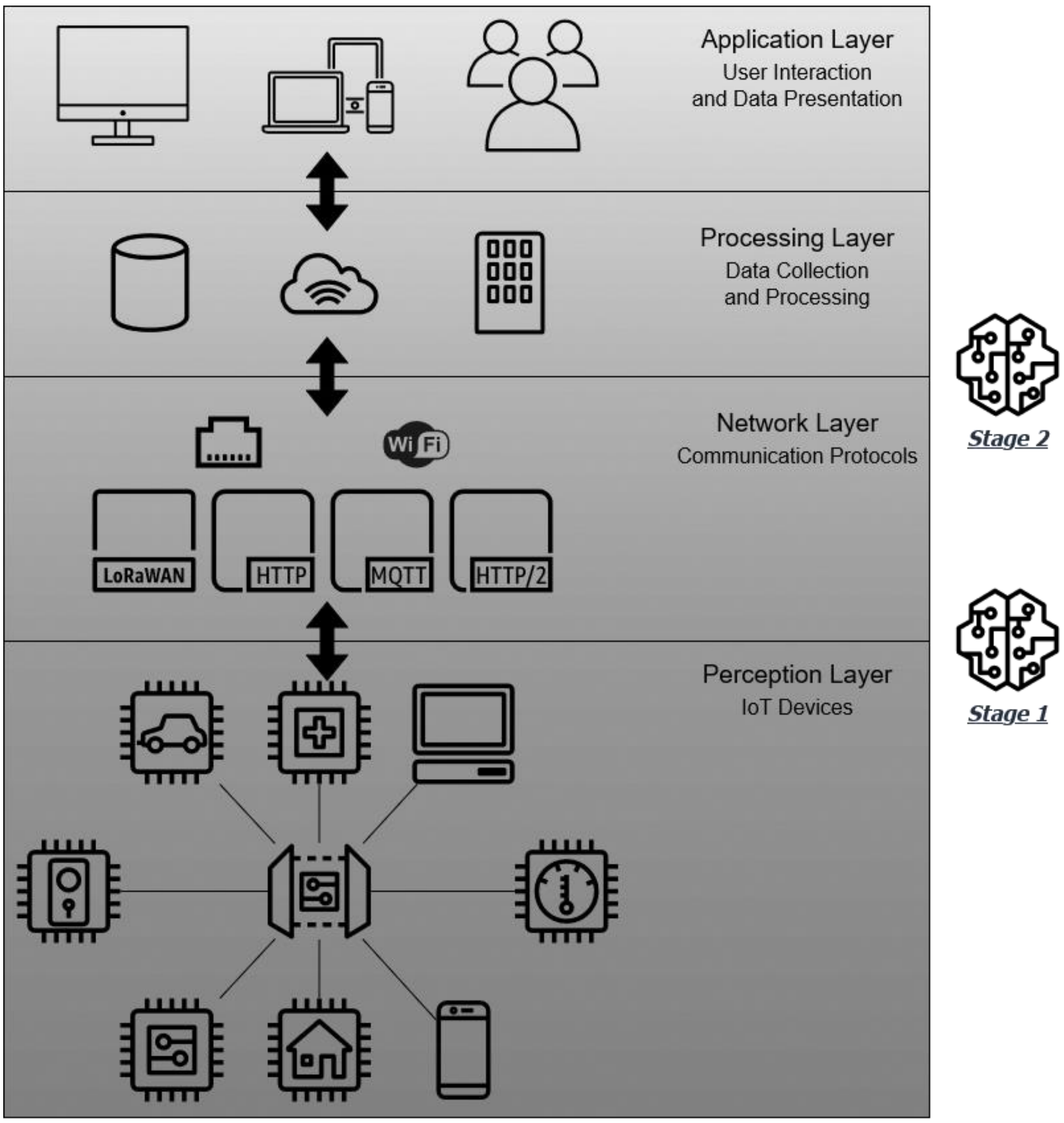


Figure 3.2: Proposed Two-Stage System in IoT Network adapted from [25].

3.5 Experimental Setup

This section discusses the dataset, experimental environment, and some additional considerations while performing the work described in this paper.

3.5.1 BoT-IoT Dataset, Imbalance and Pre-processing

The use of a recent dataset is important to accurately represent the current state of the field [27]. The BoT-IoT dataset is used to maintain congruence with the work in [25]. This dataset is very recent and contains IoT traces of a weather station, smart fridge, smart lights, a garage door, and smart thermostat. Created by the Cyber Range Lab of The Center of UNSW Canberra Cyber, it contains 69.3 GB of captured normal and botnet traffic. Each flow is labelled, first as normal (benign) or attack (malicious) traffic, then according to its category and then according to its sub-category. Attack traffic is divided into four categories: DDoS, DoS, Scan and Theft. These categories are then divided into subcategories: DDoS and DoS both contain HTTP, TCP and UDP sub-categories, Scan is divided into OS and Service scans, and Theft is split into Data Exfiltration and Fingerprinting [14].

Kororniotis et al. propose a 5% (3 million records, 1.07 GB) training and testing subset of the data [14]. The authors also identify the ten most important features. This 5% subset, using the proposed 10-best features, is used in this paper, as was also the case in [25]. It should be noted that the BoT-IoT dataset is severely imbalanced [9]. Attack traffic constitutes 99.99% of the dataset and DDoS and DoS attacks constitute a high percentage of this attack traffic. However, the objective of this work, as in [25], is to examine the capability of the various algorithms on the dataset as it was presented. Thus, no balancing techniques such as SMOTE, were used.

3.5.2 Tools

All experiments were conducted on a Windows 10 PC. The PC has the following specifications:

- CPU: Intel Core i7-10700k (8C/16T) @ 4.9GHz
- Memory: 64GB DDR4-3200 CL16
- GPU: Nvidia GeForce RTX 2070 Super

To create the experimental environment and models, the Anaconda Python 3 Distribution was used. For all deep learning Models, Keras (with a TensorFlow 2 backend) was used. For machine learning, Scikit-Learn was used. It should be noted that the Python Environment used is the exact same one used in [25]. Furthermore, while the test system does have GPU, it was not utilised, and these experiments were all run on the CPU as an equal-ground comparison is sought between the ML and DL models.

3.5.3 Performance Evaluation

Due to the imbalance of the dataset, F1-score will be used as the main evaluation metric derived from the Confusion Matrices.

Table 3.2: Confusion Matrix

	Predicted Positive	**Predicted Negative**

Labelled Positive	True Positive (TP)	False Negative (FN)
Labelled Negative	False Positive (FP)	True Negative (TN)

The metrics in Table 3.2 are as follows [28]:

- True Positive (TP): Items correctly classified as positive
- False Negative (FN): Items incorrectly classified as negative (i.e., they were positive)
- False Positive (FP): Items incorrectly classified as positive (i.e., they were negative)
- True Negative (TN): Items correctly classified as negative

Additional metrics can be derived from those in the table [29]. These are:

- Accuracy: Given by the formula $(TP + TN)/(TP + TN + FP + FN)$, this ratio shows the overall accuracy of the model and its ability to, in general, correctly predict a given input
- Precision: Given by $TP/(TP + FP)$, this shows how frequently the model is correct when predicting positive
- True Positive Rate (or Recall or Sensitivity): Given by $TP/(TP + FN)$, this is the ratio of correctly classified positive samples to number of positive samples
- False Positive Rate: Given by $FP/(FP + TN)$, this is the ratio of incorrectly classified negative samples to number of negative samples
- F1-score: Given by $2 * TP/(2 * TP + FN + FP)$, this is the harmonic mean of precision and recall. Generally, this metric is most important in cybersecurity applications [29]. This is emphasized again when considering an imbalanced dataset such as BoT-IoT.

The Weighted F1-score will not be used to evaluate performance due to the dataset's large imbalance. As stated in the previous section, all the experiments were run on the same CPU to attempt to draw comparisons between the training and classification times of the models to gain insight into their computational requirements. Thus, these times will also be a factor when judging an algorithm or algorithm combination's performance.

3.6 Results

The aim of the discussion in this section seeks to identify the best algorithms at each stage of the system as well as the best stage one and two combinations. This work follows immediately from the work performed in [25]. The same algorithms are implemented and evaluated for each stage and their combinations are tested.

Ullah and Mahmoud [24] perform similar work, albeit utilizing different algorithms and allowing their system to classify benign traffic again at the second stage. The work in this paper explicitly considers an array of ML and DL models and attempts to minimise redundant work where possible.

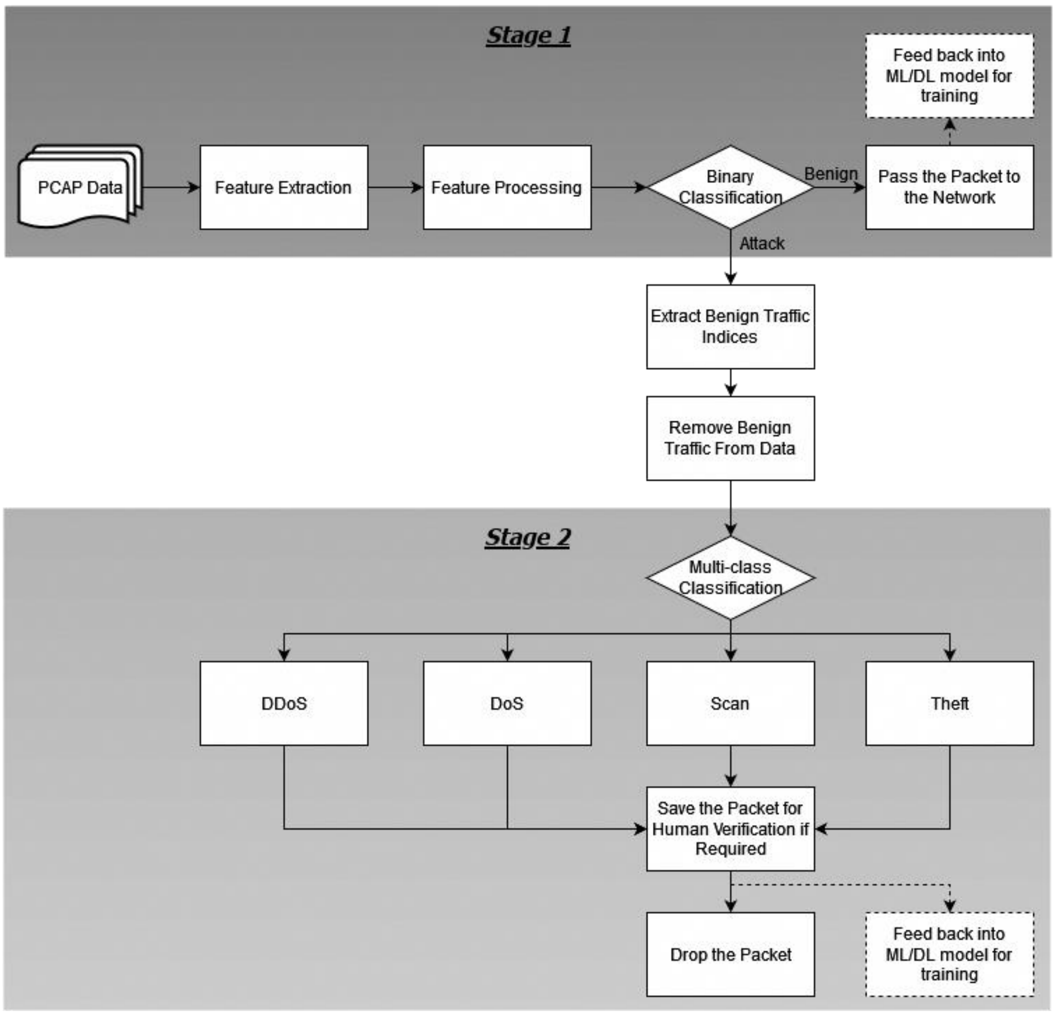

Stage 1
Feed back into ML/DL model for training
PCAP Data
Feature Extraction
Feature Processing
Binary Classification
Benign
Pass the Packet to the Network
Attack
Extract Benign Traffic Indices
Remove Benign Traffic From Data
Stage 2
Multi-class Classification
DDoS
DoS
Scan
Theft
Save the Packet for Human Verification if Required
Drop the Packet
Feed back into ML/DL model for training

Figure 3.3: Two-Stage System UML Diagram with Inter-Stage Processing.

The system in this work performs binary classification at the first stage to decide between benign and malicious traffic. Benign traffic identifiers (sequence numbers, indices etc.) are used to eliminate these flows from the data which is then passed to the second stage. This stage classifies the remaining traffic into four possible attack classes, namely: DDoS, DoS, Scan and Theft attacks. Figure 3.3 illustrates the system along with the inter-stage processing. The performance of each model at each stage is evaluated and compared and the final performance of the system is compared to a single 5-Class model that classifies attack and benign traffic.

3.6.1 Stage 1 Results

The performance of the various algorithms is first explored in the context of the first stage binary classification. The objective of this classifier is to distinguish, in binary terms, malicious traffic from benign traffic. This work is almost exactly repeated from [25] but is now performed on the test PC CPU, allowing for better comparison of ML and DL algorithms, as opposed to the GPU used for some algorithms in [25].

Table 3.3: Stage one Results

Algorithm	Accuracy	Training Time	Classification Time	F1-score
CNN	99.99%	370.94s	3.81s	0.96
DNN	99.99%	360.58s	3.54s	0.96
RNN	99.99%	385.14s	4.65s	0.96
LSTM	99.99%	446.38s	5.82s	0.96
SVM	99.99%	2.70s	0.02s	0.55
RF	99.99%	244.39s	2.46s	0.73
KNN	99.99%	1501.28s	55.31s	0.98

Table 3.4: Binary Classification Confusion Matrices.

True \ Predict	Normal (0)	Attack (1)
Normal (0)	100	7
Attack (1)	10	733588

(a) CNN Binary Classification Confusion Matrix.

True \ Predict	Normal (0)	Attack (1)
Normal (0)	101	6
Attack (1)	12	733586

(b) DNN Binary Classification Confusion Matrix.

True \ Predict	Normal (0)	Attack (1)
Normal (0)	98	9
Attack (1)	7	733591

(c) RNN Binary Classification Confusion Matrix.

True \ Predict	Normal (0)	Attack (1)
Normal (0)	105	2
Attack (1)	14	733584

(d) LSTM Binary Classification Confusion Matrix.

True \ Predict	Normal (0)	Attack (1)
Normal (0)	6	101
Attack (1)	0	733598

(e) SVM Binary Classification Confusion Matrix.

True \ Predict	Normal (0)	Attack (1)
Normal (0)	32	75
Attack (1)	0	733598

(f) RF Binary Classification Confusion Matrix.

True \ Predict	Normal (0)	Attack (1)
Normal (0)	104	3
Attack (1)	5	733593

(g) KNN Binary Classification Confusion Matrix.

Table 3.3 shows the accuracy, time and F1-score metrics for each algorithm and tables 3.4 (a) to (g) show the classification performance by way of confusion matrices. These tables provide initial insight into the expected performance of the algorithms. The accuracy metric is shown to be unreliable as a method of comparison in Table 3.3. All algorithms achieve a 99.99% accuracy. This is caused by the severe imbalance of the dataset, where 99.99% of all samples are attack traces.

The SVM model is fastest by a considerable margin for both training and classification times but has underwhelming F1-score performance. This is echoed by the confusion matrix, showing that the SVM model only correctly classified six benign samples. The RF model (another ML algorithm) performs next fastest but again does so at the expense of F1-score. The final ML algorithm, KNN, performs best of all the algorithms in F1-score but takes significantly longer to train and classify than any of the other algorithms (as also noted in [15] and [25]). The DL algorithms perform slightly worse than the KNN implementation but have much more reasonable training and classification times.

3.6.2 GPU overhead

During the implementation of the stage one algorithms on the CPU, a reduction in classification time was observed compared to the GPU implementation in [25]. An investigation was done to determine the magnitude of these changes and to identify any other metrics that have changed.

Table 3.5: GPU vs CPU Times

Algorithm	GPU Train	GPU Test	GPU F1-score	CPU Train	CPU Test	CPU F1-score
CNN Bin	294.75s	14.13s	0.96	370.94s	3.81s	0.96
DNN Bin	281.06s	9.44s	0.97	360.58s	3.54s	0.96
RNN Bin	288.87s	12.13s	0.97	385.14s	4.65s	0.96
LSTM Bin	307.42s	13.29s	0.94	446.38s	5.82s	0.96
SVM Bin	2.93s	0.05s	0.55	2.70s	0.02s	0.55
RF Bin	250.44s	2.49s	0.73	244.39s	2.46s	0.73
KNN Bin	1695.29s	55.04s	0.98	1501.28s	55.31s	0.98
CNN 5-Class	293.07s	14.94s	0.94	367.26s	4.06s	0.92
DNN 5-Class	257.11s	10.52s	0.93	354.27s	3.57s	0.95
RNN 5-Class	333.48s	12.41s	0.94	380.52s	4.57s	0.95
LSTM 5-Class	307.47s	13.07s	0.95	430.57s	5.62s	0.93
SVM 5-Class	138.31s	0.04s	0.42	138.31s	0.04s	0.42
RF 5-Class	261.08s	3.86s	0.56	261.08s	3.86s	0.56
KNN 5-Class	1683.04s	55.37s	0.97	1683.04s	55.37s	0.97
CNN 4-Class	289.71s	14.73s	0.97	357.78s	3.89s	0.97
DNN 4-Class	252.47s	10.43s	0.97	339.37s	3.57s	0.97
RNN 4-Class	286.00s	12.57s	0.97	367.74s	4.64s	0.96
LSTM 4-Class	302.17s	13.25s	0.96	433.24s	5.78s	0.84
SVM 4-Class	139.09s	0.05s	0.54	139.10s	0.05s	0.54
RF 4-Class	259.91s	3.59s	0.69	259.91s	3.59s	0.69

KNN 4-Class	1448.17s	53.89s	0.98	1448.17s	53.89s	0.98

Table 3.5 shows the results found when comparing the CPU and GPU implementations. As initially observed, there was a large improvement in classification times of the DL models when run on the CPU. Notably, the training time of these algorithms increased. Most importantly, changes in the F1-score are observed as well. These changes initially appear to be within the margin of per-run error. However, the 4-Class LSTM model saw a 0.12 decrease in F1-score when run on the CPU. The ML algorithms saw negligible changes in all metrics.

The authors theorise that the observed changes could be caused by the data transfer overhead when data is loaded onto the GPU over the PCIE 3.0 data link. If the data batch size transferred onto the GPU is too small, the accumulated overhead will result in a bottleneck. As the testing dataset is 1/4th the size of the training dataset, it is likely that the data size does not sufficiently utilise the available bandwidth when classifying. As for the changes in F1-score, the authors suspect this may be caused by the differing math libraries used on the CPU and GPU and how the model algorithms are implemented using mathematical operations. This does require further investigation to be confirmed.

3.6.3 Stage 1 and 2 Combinations

This section looks at the performance of all possible combinations of the examined algorithms. To ensure that there is minimal run-to-run variance, the indices of the traffic flows identified as benign were exported and used in the second stage to ensure all the first stage outputs (second stage inputs) for a given algorithm are the same. Furthermore, a random, single run of all the second stage algorithms was saved to be reused for all the combinations such that the run-to-run variance is eliminated.

Table 3.6: Stage One and Two Combinations

Stage 2	Stage 1	Time	Stage 2 Accuracy	Stage 2 F1-score	System Accuracy	System F1-score
CNN	CNN	7.56s	71.939%	0.57	71.942%	0.76
CNN	DNN	7.59s	71.940%	0.57	71.942%	0.75
CNN	RNN	7.65s	71.939%	0.57	71.942%	0.76
CNN	LSTM	7.75s	71.940%	0.57	71.942%	0.76
CNN	SVM	7.42s	71.930%	0.56	71.930%	0.59
CNN	RF	7.48s	71.933%	0.57	71.934%	0.66
CNN	KNN	7.50s	71.940%	0.57	71.943%	0.76
DNN	CNN	7.33s	80.512%	0.46	80.513%	0.64
DNN	DNN	7.22s	80.512%	0.46	80.514%	0.64
DNN	RNN	7.70s	80.512%	0.46	80.514%	0.64
DNN	LSTM	7.27s	80.513%	0.46	80.514%	0.64
DNN	SVM	7.61s	80.502%	0.46	80.502%	0.48
DNN	RF	7.28s	80.505%	0.46	80.505%	0.55

DNN	KNN	7.25s	80.513%	0.46	80.515%	0.65
RNN	CNN	9.13s	71.156%	0.61	71.159%	0.80
RNN	DNN	9.23s	71.156%	0.61	71.159%	0.79
RNN	RNN	9.07s	71.156%	0.61	71.159%	0.80
RNN	LSTM	9.99s	71.156%	0.61	71.159%	0.80
RNN	SVM	9.03s	71.147%	0.59	71.147%	0.61
RNN	RF	8.93s	71.149%	0.61	71.151%	0.70
RNN	KNN	9.30s	71.156%	0.61	71.160%	0.81
LSTM	CNN	9.35s	82.200%	0.60	82.201%	0.78
LSTM	DNN	9.37s	82.200%	0.60	82.201%	0.78
LSTM	RNN	9.31s	82.200%	0.60	82.201%	0.79
LSTM	LSTM	9.44s	82.200%	0.60	82.201%	0.79
LSTM	SVM	9.32s	82.189%	0.60	82.189%	0.62
LSTM	RF	9.41s	82.192%	0.60	82.193%	0.69
LSTM	KNN	9.54s	82.200%	0.60	82.202%	0.79
SVM	CNN	0.03s	81.570%	0.42	81.548%	0.60
SVM	DNN	0.03s	81.548%	0.42	81.549%	0.60
SVM	RNN	0.03s	81.546%	0.42	81.548%	0.60
SVM	LSTM	0.03s	81.548%	0.42	81.549%	0.60
SVM	SVM	0.03s	81.536%	0.42	81.536%	0.44
SVM	RF	0.03s	81.539%	0.42	81.539%	0.51
SVM	KNN	0.03s	81.547%	0.42	81.549%	0.61
RF	CNN	3.61s	90.841%	0.56	90.841%	0.74
RF	DNN	3.60s	90.841%	0.56	90.841%	0.74
RF	RNN	3.60s	90.841%	0.56	90.841%	0.74
RF	LSTM	3.68s	90.842%	0.56	90.841%	0.74
RF	SVM	3.63s	90.829%	0.56	90.829%	0.58
RF	RF	3.63s	90.832%	0.56	90.833%	0.65
RF	KNN	3.61s	90.841%	0.56	90.842%	0.75
KNN	CNN	75.70s	95.600%	0.76	95.599%	0.94
KNN	DNN	75.23s	95.600%	0.76	95.599%	0.94
KNN	RNN	75.71s	95.600%	0.76	95.599%	0.94
KNN	LSTM	76.08s	95.601%	0.76	95.599%	0.94
KNN	SVM	75.65s	95.588%	0.76	95.588%	0.78
KNN	RF	76.11s	95.591%	0.76	95.591%	0.85
KNN	KNN	75.65s	95.601%	0.76	95.601%	0.95

Table 3.6 shows all the combinations implemented. Some clear trends are immediately visible. All systems using SVM and RF first stages perform far worse than the remaining systems. KNN first stages result in the best system F1-score in all cases, but this comes at an unacceptable time penalty in the first stage (as seen in table 3.3) which allows these systems to be all but eliminated from consideration. The effect of different DL algorithms at the first stage is minimal, when comparing F1-scores, with a slight advantage found with RNN and LSTM first stages.

The disconnect between accuracy and F1-score is also clearly demonstrated again with the RF second stage having good accuracy but poor F1-score in all cases. As expected from the work in [25] and the first stage metrics, the KNN second stage systems perform best in terms of F1-score at the detriment of its time metrics. The SVM and RF second stage systems also perform worse than any of the DL second stage systems. Of the DL second stage systems, RNN performs best, closely followed by CNN.

3.6.4 Minmax Scaling

Analysing the results of the previous section showed that the systems were misclassifying many more samples than initially expected as per the 4-Class system results (see table 3.5). This prompted an exploration of the processing steps performed between stages.

The system currently records the indices of benign traffic at the end of stage one. These indices are then used to strip the benign traffic from the already processed data. This data is then passed to the second stage for classification as shown in Figure 3.3.

One of the crucial pre-processing steps is to scale the data according to the minimum and maximum values in column (MinMax Scaling). As this step is done before the benign samples are removed, it was suspected that these benign samples may 'contaminate' the data passed to the second stage (which was trained as a 4-Class classifier on data cleaned of all benign flows). To test this, a modification was made to the systems with CNN first stages.

In this modified system (seen in figure 3.4), the scaled data in the first stage is not used but instead the benign flows are removed from a copy of the processed data before scaling is applied. After these flows are removed, the data is scaled on the remaining flows. Where, in the original system in figure 3.3, the MinMax scaling occurred before removing the benign flows, here it occurs after.

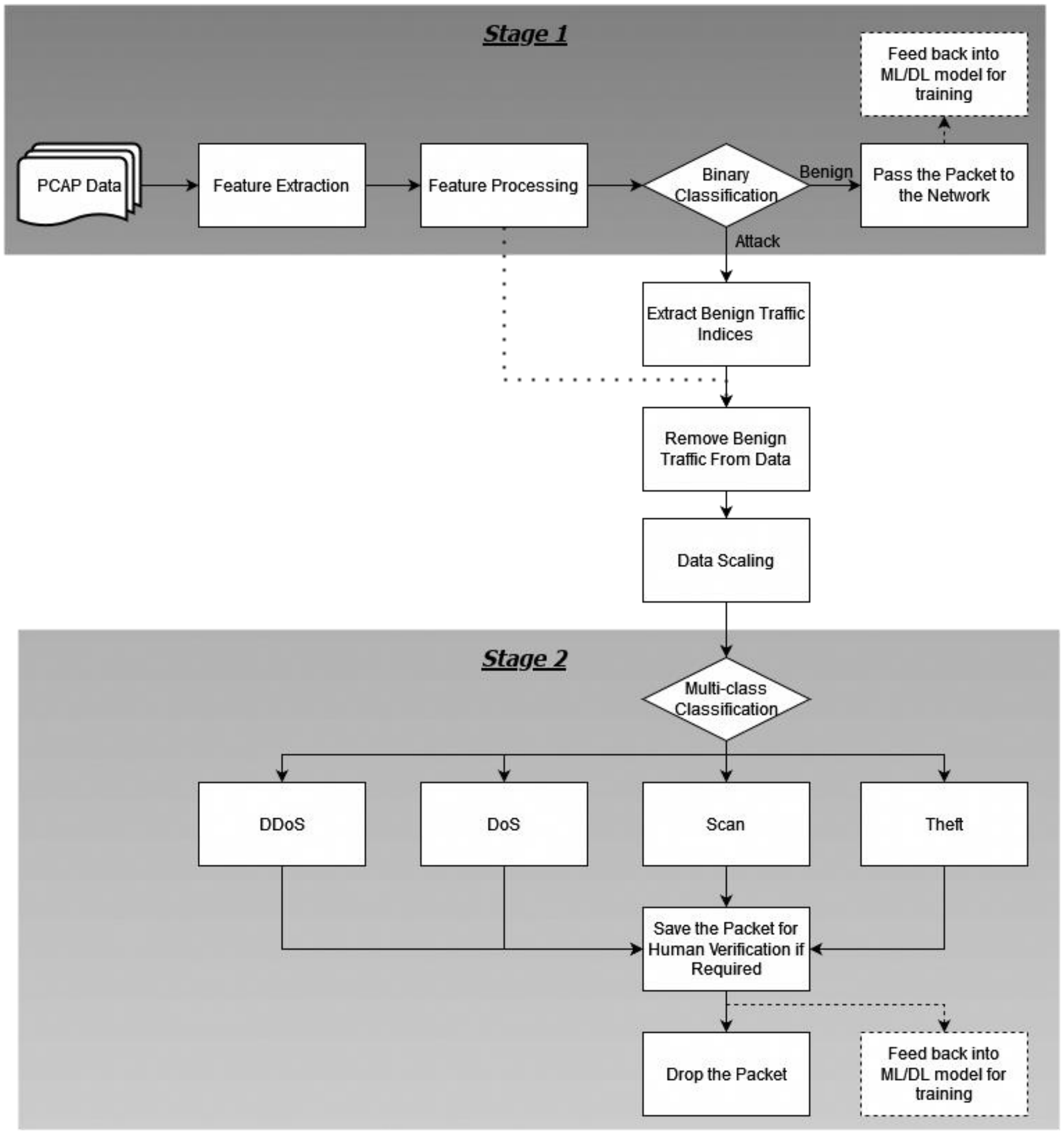

Stage 1
Feed back into ML/DL model for training
PCAP Data
Feature Extraction
Feature Processing
Binary Classification
Benign
Pass the Packet to the Network
Attack
Extract Benign Traffic Indices
Remove Benign Traffic From Data
Data Scaling
Stage 2
Multi-class Classification
DDoS
DoS
Scan
Theft
Save the Packet for Human Verification if Required
Drop the Packet
Feed back into ML/DL model for training

Figure 3.4: MinMax Scaling After Removing Benign Flows.

Table 3.7: Minmax Scaling Location Effect (% Accuracy/Macro Avg F1-score/Weighted Avg F1-score).

Algorithm	**Minmax before**	**Minmax after**
CNN	71.939/0.57/0.73	71.946/0.57/0.73
DNN	80.512/0.46/0.81	80.533/0.46/0.81
RNN	71.156/0.61/0.71	71.177/0.61/0.71
LSTM	82.200/0.60/0.82	82.499/0.60/0.83
SVM	81.547/0.42/0.81	81.585/0.42/0.81
RF	90.841/0.56/0.91	89.514/0.55/0.89
KNN	95.600/0.76/0.96	95.592/0.76/0.96

Table 3.7 shows the results of the modified system versus the original. The results are minimally different with one case of improved F1-score occurring once per system. As these results do not show significant difference, and to rule out the scaling as a potential cause of the misclassifications, the 'original' (MinMax scaling done before removing the benign flows) was compared to an identical system which used the Minimum and Maximum values from training of the 4-Class model to scale the second stage input.

Figure 3.5 shows this system. As the data in the stage one system is already scaled, it cannot be used to scale again with the 'imported' training values. Thus, a copy of the data is again taken, before the scaling in stage one is performed, to use for this purpose.

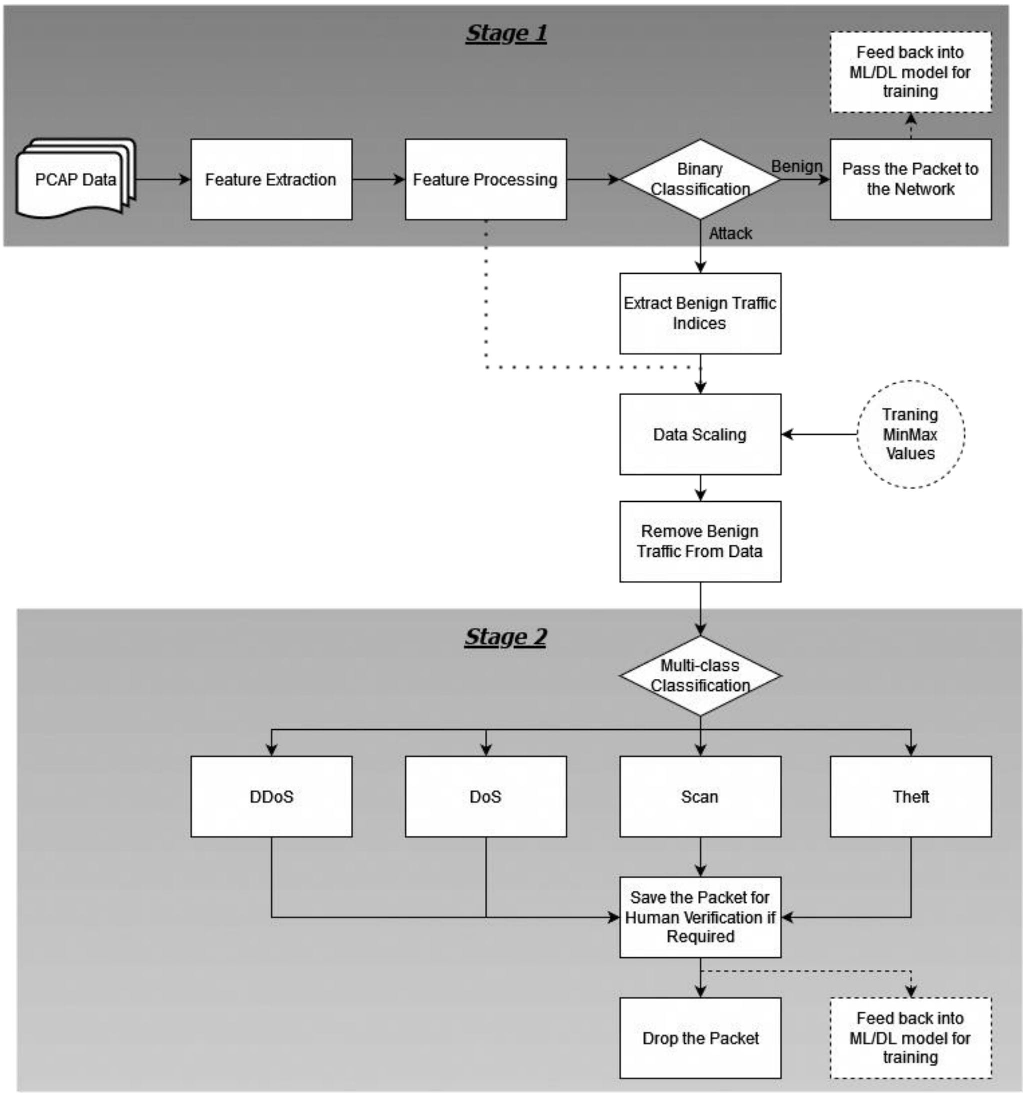

Stage 1
Feed back into ML/DL model for training
PCAP Data
Feature Extraction
Feature Processing
Binary Classification
Benign
Pass the Packet to the Network
Attack
Extract Benign Traffic Indices
Data Scaling
Traning MinMax Values
Remove Benign Traffic From Data
Stage 2
Multi-class Classification
DDoS
DoS
Scan
Theft
Save the Packet for Human Verification if Required
Drop the Packet
Feed back into ML/DL model for training

Figure 3.5: MinMax Scaling with Training Values Before Removing Benign Flows.

Table 3.8: Minmax Scaling Extremes Effect (% Accuracy/Macro Avg F1-score/Weighted Avg F1-score).

Algorithm	**Original System**	**Minmax from Training**
CNN	71.939/0.57/0.73	71.940/0.57/0.73
DNN	80.512/0.46/0.81	80.512/0.46/0.81
RNN	71.156/0.61/0.71	71.156/0.61/0.71
LSTM	82.200/0.60/0.82	82.207/0.60/0.82
SVM	81.547/0.42/0.81	81.554/0.42/0.81
RF	90.841/0.56/0.91	90.859/0.56/0.91
KNN	95.600/0.76/0.96	95.992/0.76/0.96

Now, there is even less difference observed between the two systems. This shows that there is not a significant enough difference between the values of benign and malicious flows that would cause the MinMax Scaling to have a significant effect on the performance of the system. It does not, however, explain the large classification errors observed.

3.6.5 Effect of Benign Samples on Stage 2

To explain the misclassification observed, benign samples were introduced into the CNN 4-Class system's classification step. This is to simulate a worst-case scenario where a stage one model does not identify any of the benign samples.

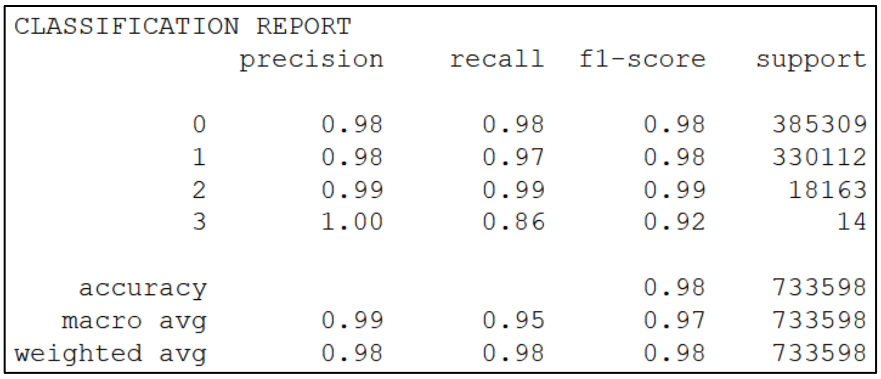
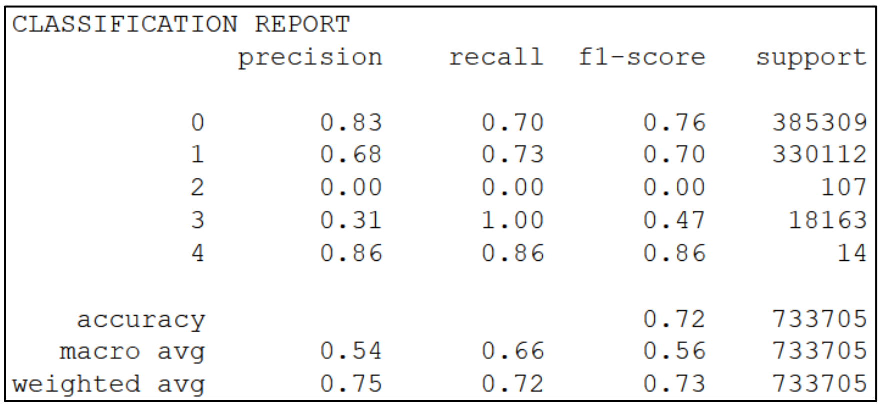

```
CLASSIFICATION REPORT
              precision    recall  f1-score   support

           0       0.98      0.98      0.98    385309
           1       0.98      0.97      0.98    330112
           2       0.99      0.99      0.99     18163
           3       1.00      0.86      0.92        14

    accuracy                           0.98    733598
   macro avg       0.99      0.95      0.97    733598
weighted avg       0.98      0.98      0.98    733598
```

Figure 3.6: CNN 4 Class Classification Report.

Table 3.9: CNN 4-Class Confusion Matrix.

True / Predict	**0**	**1**	**2**	**3**
0	377892	7358	59	0
1	8300	321777	35	0
2	77	93	17993	0
3	1	0	1	12

Figure 3.6 and Table 3.9 show the expected performance of the CNN 4-Class system when no benign samples are included.



```
CLASSIFICATION REPORT
              precision    recall  f1-score   support

           0       0.83      0.70      0.76    385309
           1       0.68      0.73      0.70    330112
           2       0.00      0.00      0.00       107
           3       0.31      1.00      0.47     18163
           4       0.86      0.86      0.86        14

    accuracy                           0.72    733705
   macro avg       0.54      0.66      0.56    733705
weighted avg       0.75      0.72      0.73    733705
```

Figure 3.7: CNN 5 Class Classification Report

Table 3.10: CNN 5-Class Confusion Matrix with Benign Samples.

True / Predict	0	1	2	3	4
0	269790	111249	0	4270	0
1	53378	239826	0	36908	0
2	0	36	0	69	2
3	28	10	0	18125	0
4	1	0	0	1	12

Figure 3.7 and Table 3.10 show the performance of the system when the benign samples are included.

Table 3.11: Effect of Benign Samples.

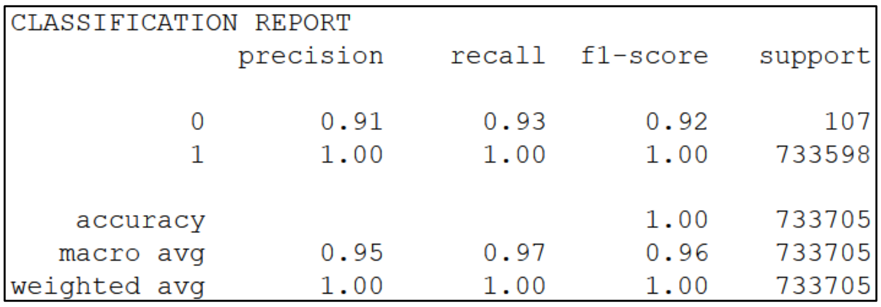
Algorithm	Classification Time	Accuracy	F1-score
CNN 4-Class	7.71s	97.829%	0.97
CNN 4-Class with Benign	7.74s	71.930%	0.56

Table 3.11 summarizes the differences in notable metrics between the two systems. The observed behaviour shows that the inclusion of benign samples has a drastic effect on the ability of the system to classify all classes. The system with benign samples exhibits behaviour similar to the two-stage system with a CNN second stage and SVM first stage. This stands to reason as the SVM first stage misses almost all benign samples. Thus, the performance of the system is heavily dependent on the ability of the first stage to accurately classify benign samples.

It could be that the benign samples appear to the model to be false starts or ends to attacks or may be disrupting concurrent flows. This behaviour is undesirable and should be investigated further by future work to 'tune it out' of the system.

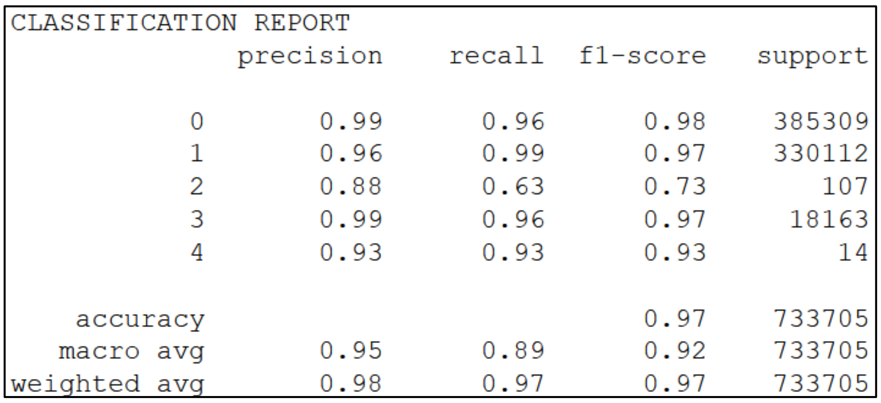
3.7 Results Discussion

The work presented in the previous section shows promising results when the first stage results and perceived classification time improvements are considered. However, the concept of a two-stage system then seems less viable from the results presented in section 5.3. It is shown that the cause of the poor performance is a lack of resilience in the models to benign data when classifying attack types. The inclusion of any benign samples into the second stage results in the system misclassifying many flows. Any of the two-stage systems under-perform against a single-stage 5-Class system. Even in the best-case scenario (without considering time metrics), the best two-stage system (KNN at first and second stage) matches the 5-Class DNN and RNN systems and is beaten by the KNN 5-Class system. However, to simply compare a single F1-score value does not give the complete picture when considering these systems. The objective of the system is to act as an IDS which requires, first and foremost, the ability to separate benign from malicious traffic.



```
CLASSIFICATION REPORT
              precision    recall  f1-score   support

           0       0.91      0.93      0.92       107
           1       1.00      1.00      1.00    733598

    accuracy                           1.00    733705
   macro avg       0.95      0.97      0.96    733705
weighted avg       1.00      1.00      1.00    733705
```

Figure 3.8: CNN Binary Classification Report.



```
CLASSIFICATION REPORT
              precision    recall  f1-score   support

           0       0.99      0.96      0.98    385309
           1       0.96      0.99      0.97    330112
           2       0.88      0.63      0.73       107
           3       0.99      0.96      0.97     18163
           4       0.93      0.93      0.93        14

    accuracy                           0.97    733705
   macro avg       0.95      0.89      0.92    733705
weighted avg       0.98      0.97      0.97    733705
```

Figure 3.9: CNN 5 Class Classification Report.

Figures 3.8 and 3.9 show the classification reports of the CNN stage one binary classifier and the single stage 5-Class CNN model respectively. Note here the F1-score for class '0' in Figure 3.8

and for class '2' in Figure 3.9. These are the F1-scores for the benign class in each system. Clearly, the binary system greatly outperforms the single-stage 5-Class system in this regard.

Table 3.12: Benign Sample Classification Ability.

Algorithm	**Binary F1-score**	**5-Class F1-score**
CNN	0.96	0.73
DNN	0.92	0.88
RNN	0.92	0.84
LSTM	0.93	0.80
SVM	0.11	0.02
RF	0.46	0.80
KNN	0.96	0.95

Table 3.12 summarizes this same result for all 11 algorithms. In all cases, the binary models are far more capable of classifying benign samples correctly (i.e., will produce fewer false positives). The implication of this for the two-stage system is that it would be more valuable as an IDS than any of the 5-Class models would [6], [7]. Furthermore, it shows that the 5-Class F1-score is skewed by the severe imbalance of the dataset.

Perhaps the most notable result here is the equivalent performance of the CNN and KNN Binary models. Considering the time metrics, the CNN model clearly outperforms the KNN model, achieving a similar F1-score while being 14.52 times faster. This allows for an informed decision on a proposed 'best' combination for a two-stage system.

In the two-stage system, the first stage has the requirements of being as fast and accurate as possible. From table 3.6, this is achieved by the CNN first stage model. Stage two should be as capable as possible to classify various attack types but is not severely constrained by the need for fast classification times. Thus, from the results presented in section 5, KNN would be best here. This results in a ML-DL combination 'best' system which was also identified as promising by [11].

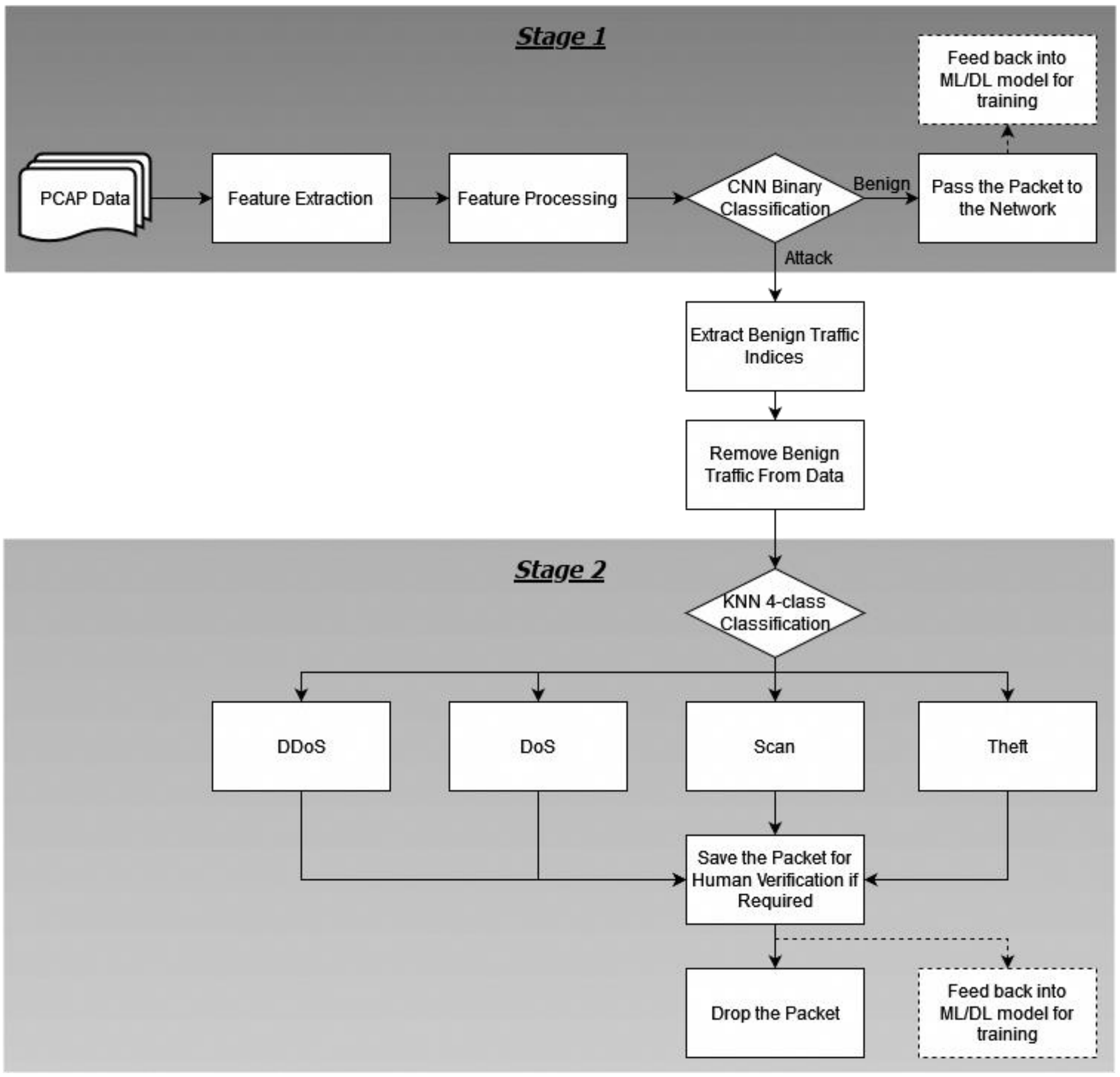

Stage 1
PCAP Data
Feature Extraction
Feature Processing
CNN Binary Classification
Benign
Pass the Packet to the Network
Feed back into ML/DL model for training
Attack
Extract Benign Traffic Indices
Remove Benign Traffic From Data
Stage 2
KNN 4-class Classification
DDoS
DoS
Scan
Theft
Save the Packet for Human Verification if Required
Drop the Packet
Feed back into ML/DL model for training

Figure 3.10: Final Proposed Two-Stage System

Table 3.13: Two-Stage System Comparison.

Metric	Two-Stage System	Compared System
Stage 1: Classification Time	3.81s	4.06s (CNN 5-Class)
Stage 1: Benign F1-score	0.96	0.73 (CNN 5-Class)
Stage 2: System F1-score	0.94	0.98 (KNN 5-Class)

Figure 3.10 shows the proposed final system and Table 3.13 shows how the system compares in important metrics at each stage to the relevant 5-Class systems. While the system does not perform as well in attack type classification as the KNN 5-Class, the results in Table 3.13 strongly suggest that the proposed two-stage system would perform far better as an IDS than any of the examined 5-Class systems.

3.8 Conclusion

In this paper, the performance of various algorithm combinations for the two-stage system proposed in [25] were investigated. This system aims to provide time and classification metric improvements when compared to single-stage systems by layering different ML and DL algorithms. The first stage classifies benign and malicious traffic, with the second stage identifying the type of attack.

First, the stage one, binary classification models were evaluated on the test system CPU, as opposed to the GPU used for the DL models in [25]. This allows for 'equal ground' comparison of the various algorithms. It was found that the results deviated from [25], noting changes in F1-scores and decreased classification times in some cases. This was explored further, and all models (Binary, 4-Class, and 5-Class) were run again on the CPU. This revealed similar behaviour across all models. The decrease in classification times of the DL models specifically, leads the authors to believe that these differences are the result of the data transfer overhead when moving data to the GPU for classification. The observed behaviour did not alter the status or viability of the models and further testing proceeded.

Next, all 49 algorithm combinations were investigated, and pertinent metrics recorded. This showed clear trends in model performances. It was shown that the ML models, when used at the first stage, produced unacceptable results. In the case of SVM and RF first stages, they performed poorly when distinguishing between benign and malicious samples, resulting in lower-than-average F1-scores for the system. SVM produced 0.12 lower F1-scores on average and RF, 0.05 lower on average. The classification time performance of the KNN model rules it out for use in the first stage, taking 10-times longer than the next slowest model. When used at the second stage however, the KNN model performed consistently better than other algorithms and is the clear 'winner'.

The generally poor performance of the systems prompted an investigation into the cause. Initial suspicions that the order of pre-processing steps is the cause was ruled out, finding instead that the 4-Class models used in the second stage are extremely sensitive to benign sample contamination and that the inclusion of any benign samples would severely hamper the ability of the model to accurately predict attack types and result in lower-than-expected F1-scores.

However, when considering the appropriate metrics at each stage and remaining cognisant of the fact that the system is designed to be an IDS, it is shown that the proposed system would still perform better than any of the single-stage 5-Class systems. This is due to the 0.25s classification time speed-up over the 5-Class CNN system and the significant 0.23 F1-score improvement for benign samples, reducing the number of false positives.

It should also be noted that there were a few challenges encountered while performing the work described. A major challenge arose before any experimentation even began; it quickly became apparent that there is very little related literature in this highly specific area. While this provided some additional leeway in performing the work, it also complicated the procedure of verifying and comparing results to similar works. During experimentation, the described observations of GPU overhead, MinMax scaling and remnant benign samples at stage 2 provided additional areas of investigation that had to be ruled out as factors that could invalidate any of the results obtained. Finally, a challenge that persisted throughout the work performed is that of ensuring homogenous test system conditions. The test system is a traditional Windows OS PC, meaning significant multi-tasking is occurring while experiments are being run. To limit this as far as possible, attempts were made to ensure that no unnecessary programs or services were running during any of the experiments, and in some cases it was necessary to run experiments multiple times to validate results obtained.

The proposed system achieves an overall F1-score of 0.94. While this is not as high as a KNN 5-Class system, the classification time benefit and benign sample F1-score improvements offered by the proposed system make this a better solution to the challenges faced by IoT network security managers. Specifically, the proposed system offers improved threat detection and lower false positive rates. This would reduce the need for the security administrator’s intervention in the system operation, which has potential operational and financial benefits to the organisation [30].

Future work should further explore and develop the two-stage system proposed in this paper. While the system is shown to perform better as an IDS than a single-stage ML or DL solution, there are several areas in which the system could be even further improved. Additional tuning of the models used at each stage may yield further time and classification metric enhancements. The second-stage shortcomings, namely when benign samples are not correctly identified at the first stage, should be further investigated, and addressed. The 'lightweight' requirement of this system should also be further explored with the intention of reducing the load placed on IoT devices in resource-constrained situations. The observed drop in F-score of the LSTM model when running of the CPU of the test PC should also be investigated to better understand why it occurs and if it may be prevented. Similarly, the seemingly insignificant impact of changing when minmax scaling is performed during pre-processing should be interrogated. Ideally, future work would investigate the system as proposed, implemented on appropriate hardware and/or networks and evaluated with traffic representative of real-world scenarios. Finally, the financial and resource benefits such a system may offer to an organisation or business must also be explored.

References

[1] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, ‘Machine learning for Internet of Things data analysis: A survey’, *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, Aug. 2018, doi: 10.1016/j.dcan.2017.10.002.

[2] M. R. Shahid, G. Blanc, Z. Zhang, and H. Debar, ‘IoT Devices Recognition Through Network Traffic Analysis’, in *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA, Dec. 2018, pp. 5187–5192. doi: 10.1109/BigData.2018.8622243.

[3] H. Nguyen-An, T. Silverston, T. Yamazaki, and T. Miyoshi, ‘IoT Traffic: Modeling and Measurement Experiments’, *IoT*, vol. 2, no. 1, pp. 140–162, Feb. 2021, doi: 10.3390/iot2010008.

[4] Palo Alto, ‘2020 Unit 42 IoT Threat Report’. Palo Alto, 2020. Accessed: Sep. 17, 2021. [Online]. Available: https://unit42.paloaltonetworks.com/iot-threat-report-2020/

[5] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, ‘DDoS in the IoT: Mirai and Other Botnets’, *Computer*, vol. 50, no. 7, pp. 80–84, 2017, doi: 10.1109/MC.2017.201.

[6] R. Calderon, ‘The Benefits of Artificial Intelligence in Cybersecurity’, *Economic Crime Forensics Capstones*, Jan. 2019, [Online]. Available: https://digitalcommons.lasalle.edu/ecf_capstones/36

[7] G. Kumar and K. Kumar, ‘The Use of Artificial-Intelligence-Based Ensembles for Intrusion Detection: A Review’, *Applied Computational Intelligence and Soft Computing*, vol. 2012, pp. 1–20, 2012, doi: 10.1155/2012/850160.

[8] C. Liang, B. Shanmugam, S. Azam, M. Jonkman, F. D. Boer, and G. Narayansamy, ‘Intrusion Detection System for Internet of Things based on a Machine Learning approach’, in *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, Mar. 2019, pp. 1–6. doi: 10.1109/ViTECoN.2019.8899448.

[9] Y. N. Soe, P. I. Santosa, and R. Hartanto, ‘DDoS Attack Detection Based on Simple ANN with SMOTE for IoT Environment’, in *2019 Fourth International Conference on Informatics and Computing (ICIC)*, Oct. 2019, pp. 1–5. doi: 10.1109/ICIC47613.2019.8985853.

[10] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, ‘Deep Learning-Based Intrusion Detection for IoT Networks’, in *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)*, Kyoto, Japan, Dec. 2019, pp. 256–25609. doi: 10.1109/PRDC47002.2019.00056.

[11] B. Susilo and R. F. Sari, ‘Intrusion Detection in IoT Networks Using Deep Learning Algorithm’, *Information*, vol. 11, no. 5, p. 279, May 2020, doi: 10.3390/info11050279.

[12] M. Shafiq, Z. Tian, Y. Sun, X. Du, and M. Guizani, ‘Selection of effective machine learning algorithm and Bot-IoT attacks traffic identification for internet of things in smart city’,

Future Generation Computer Systems, vol. 107, pp. 433–442, Jun. 2020, doi: 10.1016/j.future.2020.02.017.

[13] A. Tabassum, A. Erbad, and M. Guizani, 'A Survey on Recent Approaches in Intrusion Detection System in IoTs', in *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, Jun. 2019, pp. 1190–1197. doi: 10.1109/IWCMC.2019.8766455.

[14] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, 'Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset', *arXiv:1811.00701 [cs]*, Nov. 2018, Accessed: Sep. 30, 2020. [Online]. Available: http://arxiv.org/abs/1811.00701

[15] J. Alsamiri and K. Alsubhi, 'Internet of Things Cyber Attacks Detection using Machine Learning', *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 10, no. 12, Art. no. 12, Jun. 2019, doi: 10.14569/IJACSA.2019.0101280.

[16] J. Hussain, S. Lalmuanawma, and L. Chhakchhuak, 'A two-stage hybrid classification technique for network intrusion detection system', *IJCIS*, vol. 9, no. 5, p. 863, 2016, doi: 10.1080/18756891.2016.1237186.

[17] 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*, Ottawa, ON, Canada, Jul. 2009, pp. 1–6. doi: 10.1109/CISDA.2009.5356528.

[18] N. Kaja, A. Shaout, and D. Ma, 'A Two Stage Intrusion Detection Intelligent System', presented at the The International Arab Conference on Information Technology, 2018.

[19] Khraisat, Gondal, Vamplew, Kamruzzaman, and Alazab, 'A novel Ensemble of Hybrid Intrusion Detection System for Detecting Internet of Things Attacks', *Electronics*, vol. 8, no. 11, p. 1210, Oct. 2019, doi: 10.3390/electronics8111210.

[20] I. Ullah and Q. H. Mahmoud, 'A Two-Level Hybrid Model for Anomalous Activity Detection in IoT Networks', in *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan. 2019, pp. 1–6. doi: 10.1109/CCNC.2019.8651782.

[21] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, 'Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization':, in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, Funchal, Madeira, Portugal, 2018, pp. 108–116. doi: 10.5220/0006639801080116.

[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)*, Canberra, Australia, Nov. 2015, pp. 1–6. doi: 10.1109/MilCIS.2015.7348942.

[23] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescape, ‘A Hierarchical Hybrid Intrusion Detection Approach in IoT Scenarios’, in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, Taipei, Taiwan, Dec. 2020, pp. 1–7. doi: 10.1109/GLOBECOM42002.2020.9348167.

[24] I. Ullah and Q. H. Mahmoud, ‘A Two-Level Flow-Based Anomalous Activity Detection System for IoT Networks’, *Electronics*, vol. 9, no. 3, p. 530, Mar. 2020, doi: 10.3390/electronics9030530.

[25] A. van der Walt, T. Quazi, and B. van Niekerk, ‘Two-Stage IDS for IoT Using Layered Machine- and Deep-Learning Models’, *Unpublished*, 2021.

[26] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, ‘Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study’, *Journal of Information Security and Applications*, vol. 50, p. 102419, Feb. 2020, doi: 10.1016/j.jisa.2019.102419.

[27] T. Singh and N. Kumar, ‘Machine learning models for intrusion detection in IoT environment: A comprehensive review’, *Computer Communications*, Feb. 2020, doi: 10.1016/j.comcom.2020.02.001.

[28] S. Kumar, A. Viinikainen, and T. Hamalainen, ‘Evaluation of ensemble machine learning methods in mobile threat detection’, in *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, Dec. 2017, pp. 261–268. doi: 10.23919/ICITST.2017.8356396.

[29] Y. Xin *et al.*, ‘Machine Learning and Deep Learning Methods for Cybersecurity’, *IEEE Access*, vol. 6, pp. 35365–35381, 2018, doi: 10.1109/ACCESS.2018.2836950.

[30] L. Lazic, ‘Benefit From AI In Cybersecurity’, Oct. 2019.

Chapter 4

Concluding Remarks

In this dissertation, a two-stage intrusion detection system using machine- and deep-learning methods for IoT networks has been proposed and investigated. This two-stage system improved upon time and classification metrics when compared to single-stage detection methods and, thus, the system is shown to be a viable IDS solution.

In Chapter 2, the two-stage system was proposed. Applicable algorithms were identified and investigated in specific 'classification scenarios' to determine the potential performance benefits of a two-stage system. The BoT-IoT dataset is utilised in an unmodified form to achieve as close a representation to real-world traffic as possible. First, the implementation and classification scenarios of Koroniotis et al. [32] were replicated, serving as a base for the research conducted. These implementations were then modified to find more effective hyperparameters, model architectures and target vectors. Tuning was conducted to optimise F1-score and training time metrics.

Additional ML and DL models were also explored, resulting in a total of seven models - DNN, CNN, RNN, LSTM, SVM, RF, KNN - evaluated in three classification scenarios - binary (attack/benign), 4-Class (attack only) and 5-Class (attack and benign). The results obtained indicated that the KNN implementation, with k=3, outperforms all other algorithms in terms of F1-score but takes far longer to train and classify. All the next-best performing models were neural network implementations. DNN performed best in the binary classification scenario, LSTM performed best in the 5-Class scenario and CNN performed best in the 4-Class scenario. This reinforces the opinion found in literature, that DL methods present the most promising outlook when it comes to securing systems.

Using these results, the expected performance benefits of a two-stage system were presented. Time metrics improvements of 0.51s on average over a single-stage system were observed at the first stage with a F1-score improvement of 0.05 on average seen at the second stage. This indicates a system that can perform better than a single-stage method at a lesser performance cost.

In Chapter 3, the expected performance of the system was put to the test and the system was evaluated in its final arrangement. The performance of all algorithm combinations was investigated. The first stage, binary classification models were re-evaluated on the test system CPU, as opposed to the GPU used for the DL models previously, to provide a more direct comparison of the time metrics for each model. Variations were observed in the F1-scores and, interestingly, decreases in classification time metrics where increases were expected. To understand this anomaly, all models (Binary, 4-Class and 5-Class) were re-evaluated on the CPU.

Similar behaviour was seen in all models. As these changes were seen only in the models that had previously utilised the GPU, it is theorised that the behaviour is caused by the transfer time overhead when moving data between the CPU and GPU. The observed behaviour did not, however, significantly alter the objective of the work or its ability to proceed.

All 49 possible algorithm combinations were then tested, evaluating pertinent metrics. All cases where a ML model - SVM, RF or KNN - was employed at the first stage resulted in a system that produced unacceptable performance. In the case of SVM and RF at the first stage, poor F1-score performance was observed. SVM produced a 0.12 lower F1-score on average and RF produced a 0.05 lower F1-score on average. Using KNN at the first stage provided an excellent F1-score result but extremely poor first stage classification time performance, taking 10-times longer than the next slowest model. CNN and DNN performed best at the first stage but produced extremely similar results, varying within margin of error in classification time. At the second stage, the KNN model can be used. As the second stage is not necessarily a time-sensitive stage, time metrics can be sacrificed in favour of classification metrics, in which KNN performs best.

The system's performance deviated from what was expected, underperforming in certain areas. This was investigated to determine the cause. While it was initially thought that the order in which data-scaling was performed during data pre-processing was the cause, this was ruled out, showing minimal differences when this step's order was modified. It was then discovered that the second-stage 4-Class models are negatively affected by any benign samples missed by the first stage. If any benign samples were put through to the second stage, the model would misclassify several samples, resulting in low F1-scores.

Despite this apparent poor performance, it is seen that the two-stage system still outperforms a single-stage 5-Class system in two very important metrics. A two-stage system consisting of a CNN binary model at the first stage and a KNN 4-Class model at the second, performs 0.25s faster than a 5-Class CNN in binary classification, reducing any potential network traffic bottleneck. The two-stage system also exhibited a 0.23 F1-score improvement for benign samples over a CNN 5-Class system. This would significantly reduce false positive alarms. The conclusion can then be drawn that despite an overall system F1-score decrease of 0.04, the two-stage system would outperform any of the evaluated 5-Class systems as an IDS.

The two-stage system proposed in this dissertation should be further explored in future work. There is room for further development and enhancements in several areas of the system. In terms of the per-stage performance, additional tuning or variation of the models may yield improved performance. Most importantly, the sensitivity of the second stage to benign sample ‘contamination’ from the first stage must be explored to a greater degree and addressed. Finally, the system should also be optimised further to make it more ‘lightweight’ and thus, suitable to a wider range of IoT applications where resources may be constrained.