BOTNET DETECTION USING DEEP LEARNING APPROACHES

Prepared By

Kiran Kumar Mohanraj - 20054954
MSc Cyber Security
University of the West of England, Bristol

This study was completed as the Dissertation for the MSc in Cyber Security at the University of the West of England, Bristol. This is my work, where the works of others were used or drawn upon, it has been attributed.

BOTNET DETECTION USING DEEP LEARNING APPROACHES

Master of Science

Cyber Security

UWE, Bristol, January 2022

Kiran Kumar Mohanraj

ABSTRACT

With the proliferation of the Internet of Things (IoT), computer networks have rapidly expanded in size. While Internet of Things Devices (IoTDs) benefit many aspects of life, these devices also introduce security risks in the form of vulnerabilities which give hackers billions of promising new targets. For example, botnets have exploited the security flaws common with IoTDs to gain unauthorized control of hundreds of thousands of hosts, which they then utilize to carry out massively disruptive distributed denial of service (DDoS) attacks. Traditional DDoS defense mechanisms rely on detecting attacks at their target and deploying mitigation strategies toward the attacker but differentiating between botnet attack traffic from normal traffic is extremely difficult, rendering mitigation strategies ineffective. An expanding body of work seeks to sidestep this difficulty by using sophisticated machine learning algorithms to detect botnet-based attacks at their source; however, many of these algorithms are computationally demanding and require specialized hardware, which is expensive, rendering them impractical. In this study, Deep Residual Convolutional Neural Network (CNN) model, was proposed to detect botnet attacks, namely, BASHLITE and Mirai, on nine commercial IoT devices. Extensive empirical research was performed by employing a real N-BaIoT dataset extracted from a real system, including benign and malicious patterns. This study also implements Artificial Neural Network (ANN) Model and Convolutional Neural Network and Long Short-Term memory (CNN-LSTM) model to evaluate the proposed model. The experimental results exposed the superiority of the Deep Residual CNN model with accuracies of 87.27% in detecting botnet attacks from Provision PT-737E Security Camera, whereas the ANN and CNN-LSTM model showed 83.66% and 75.02% respectively. Overall, the Deep Residual CNN model was successful in detecting botnet attacks from various IoT devices with optimal accuracy.

KEYWORDS: convolutional neural networks, deep learning, distributed denial of service attacks, IoT security, long short-term memory, concurrent neural networks.

ACKNOWLEDGEMENTS

The completion of this thesis would not have been possible without the guidance of my supervisor, Mr. Ian Johnson. Without the thoughtful questions which provided direction for my curiosity, the challenge to push myself toward my goals, and the limitless supply of patience he afforded me I never could have made it this far.

I would also like to thank Dr. Phil Legg, and the other faculty and staff of the FET - Computer Science and Creative Technologies department who provided guidance in the completion of this degree.

Lastly, I would like to thank my parents, relatives, and friends for always supporting and believing in me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1 Background

With the rapid growth of smart devices and high-speed networks, the Internet of Things (IoT) has gained wide acceptance and popularity as the main standard for low-power lossy networks (LLNs) having constrained resources. It represents a network where ‘‘things’’ or embedded devices having sensors are interconnected through a private or a public network. The devices in IoT can be controlled remotely to perform the desired functionality. The information sharing among the devices then takes place through the network which employs the standard protocols of communication (Atzori, Iera and Morabito, 2010). At the start of this decade, there were an estimated 12.5 billion IoT devices, almost twice as much as the world's population of 6.8 billion people (Evans, 2011). The projections estimate that 125 billion IoT devices will be connected to the Internet by 2030 (Heslop, 2019), the IoT has established itself as a cornerstone of the modern technological landscape. However, ensuring the security of these systems has persisted as an open issue long after their adoption.

The security challenges of IoT can be broadly divided into two classes; Technological challenges and Security challenges. The technological challenges arise due to the heterogeneous and ubiquitous nature of IoT devices, while the security challenges are related to the principles and functionalities that should be enforced to achieve a secure network. Technological challenges are typically related to wireless technologies, scalability, energy, and distributed nature, while security challenges require the ability to ensure security by authentication, confidentiality, end-to-end security, integrity etc. Security should be enforced in IoT throughout the development and operational lifecycle of all IoT devices and hubs (Leo *et al.*, 2014).

In 2017, 86% of the total attacks that have been launched are observed to be of multiple type's means composed of different types of variations possible in firing a distributed denial-of-service (DDoS) attack, hence making it difficult to identify and mitigate (Kaspersky, 2018). In Oct. 2016, Dyn (major US DNS service provider) was under one of the largest and most powerful DDoS attacks in recent history by Mirai malware family. The malware infected over 1.2 million IoT devices and targeted many popular online services such as Google, Amazon, etc (Herzberg, 2016). Attacks implemented by the Mirai botnet laid the foundation of appearance of great number of botnets consisting of IoT devices, for example, the botnet Leet (Bekerman and Zawoznik, 2016) and the botnet Amnesia (LaptrinhX, 2017). The majority of botnets gain unauthorized access to IoT devices by performing brute-force attacks on TELNET and/or SSH services. According to an examination of cybersecurity of the Internet of Things (Prokofiev, Smirnova and Silnov, 2017), nearly 400,000 IoT devices accept connections via these 2 services, moreover, some devices do not require authentication. Nowadays a significant increase of a rate of DDoS-attacks powered by IoT devices is in evidence. The large number of insecure IoT devices with high computation power make them an easy and attractive target for attackers seeking to compromise these devices and use them to create large-scale botnets. These botnets has a command-and-control infrastructure and is used for DDoS attacks (Feily, Shahrestani and Ramadass, 2009).

Botnet attack detection using machine learning has been shown to be highly accurate (Diro and Chilamkurti, 2018), but carries a high computational cost. Many approaches favor the use of deep learning algorithms, which have achieved state-of-the-art results across multiple application domains, but, while highly accurate, these algorithms often necessitate the use of

GPU acceleration to ensure low runtime latency (Goodfellow *et al.*, 2016). Unfortunately, the price of artificial intelligence grade GPUs places them out of reach for the average user. This casts doubt on the abilityof deep learning-based botnet detection mechanisms to gain widespread adoption. To overcome this challenge, this thesis presents a network-level botnet attack detection system using deep learning.

1.2 Research Motivations

Deep learning approaches to botnet attack detection are highly effective (McDermott, Majdani and Petrovski, 2018), but the computational complexity of such approaches make them impractical for access network deployment. Current approaches contain inefficiencies, such as unnecessary preprocessing, traffic analysis on a per-device basis (Meidan *et al.*, 2018), and analysis on a per-packet basis (McDermott, Majdani and Petrovski, 2018). The motivation of this work is to develop a botnet attack detection system which provides accurate detection while scaling with the massive network sizes of the IoT.

The design of the proposed botnet attack detection system and the methods used to evaluate it are designed in accordance with the following strategies:

1) Ensure computational efficiency by:
   a) Utilizing deep learning algorithms,
   b) Eliminating unnecessary data preprocessing.
2) Ensure scalability by:
   a) The system design not increasing in complexity with the size of the network,
   b) The system design not processing individual network packets.
3) Ensure effectiveness by testing the system with data specific to IoT botnet attack scenarios.

1.3 Research Contribution

The proposed botnet attack detection system utilizes a novel detection mechanism which operates as follows:

- The IoT botnets are detected and classified according to the type of attack.
- For detection, three deep learning approaches are implemented.
- The proposed Deep residual CNN model is evaluated with the existing ANN and CNN-LSTM model to prove its effectiveness.
- The evaluation metrics are used to calculate the accuracy and loss of the deep learning model.

1.4 Overview

The remainder of this thesis is organized as follows: the structure of the IoT in terms of architecture, application domains, and security vulnerabilities is described in Chapter 2, followed by description of botnet's configuration and architecture in Chapter 3. Chapter 4 provides an in-depth analysis of botnet attacks and its types. Chapter 5 contains a literature review covering the current state-of-the-art in botnet attack detection mechanisms. The novel botnet attack detection mechanism is presented in Chapter 6. Chapter 7 presents the results and discussion of the experimentation. Concluding remarks and directions for future work are discussed in Chapter 8.

2 THE INTERNET OF THINGS

2.1 IoT Emergence

The term IoT-Internet of Things was coined by “Kevin Ashton” in 1999 with regard to Supply Chain Management (Asthon, 2010). But seeing the current scenario, the term IoT has been expanded to wide range of applications like Utilities, Transport, Wearable Technologies, Smart Home Automation and even Military Applications like Robotics and Drones. IoT technology is transforming the way we live life every day. From consumer point of view, new IoT products like Home Automation, wearable devices, Household devices etc. are evolving and IoT technology is also giving lots of benefits even to disabled people with development of various health monitoring devices, wearable medical gadgets, Remote Health Monitoring equipment’s etc. IoT Technologies like Vehicle Networking, Traffic Management Intelligent Systems are integrated with sensors fitted on roads transforming traditional cities to “SMART CITIES” and overcoming various issues like Traffic Jams, Road Accidents etc. (Chen *et al.*, 2014).

When comparing the IoT with traditional network technology, the key feature is the connected “things”, or IoTDs (IoT Devices). There is a high degree of variation among IoTDs in terms of size, shape, communication functionality, and power. To provide clarity among this diversity, (Miorandi *et al.*, 2012) presented a definition for IoTDs that encompasses theircommon characteristics. This definition outlines six defining features of an IoTD:

1. It must possess some computational capacity.
2. It must be embodied by physical hardware.
3. It must possess some communication capability.
4. It must be associated with a unique identifier.
5. It must be associated with a hostname and IP Address.
6. It must possess either the ability to gather information from its physical environment (a sensor), the ability to change its physical environment (an actuator), or both.

These characteristics illuminate the greater functionality of the IoT, namely its ability to either sense or change a physical environment. When IoTDs are connected to the Internet, they form a global and ubiquitous network which enables web services and applications to interact with the physical world (Ratasich *et al.*, 2019).

2.2 The Architecture of the Internet of Things

The building blocks of IoT are sensory devices, remote service invocation, communication networks, and context-aware processing of events; these have been around for many years. However, what IoT tries to picture is a unified network of smart objects and human beings responsible for operating them, who are capable of universally communicating with each other.

When talking about a distributed environment, interconnectivity among entities is a critical requirement, and IoT is a good example. A holistic system architecture for IoT needs to guarantee flawless operation of its components (reliability is considered as the most import design factor in IoT) and link the physical and virtual realms together. To achieve this, careful consideration is needed in designing failure recovery and scalability (Wu *et al.*, 2010). Additionally, since mobility and dynamic change of location has become an integral part of IoT systems with the widespread use of smartphones, state-of-the-art architectures need to have a

certain level of adaptability to properly handle dynamic interactions within the whole ecosystem (Luzuriaga *et al.*, 2015).

(Xu, He and Li, 2014) provide a big-picture overview of the IoT through a service-oriented architecture (SOA). The SOA for the IoT, illustrated in Figure 1, contains four layers: the sensing layer, the networking layer, the service layer, and the interface layer (Xu, He and Li, 2014). Each of these layers build on the others to accomplish higher order functionality.

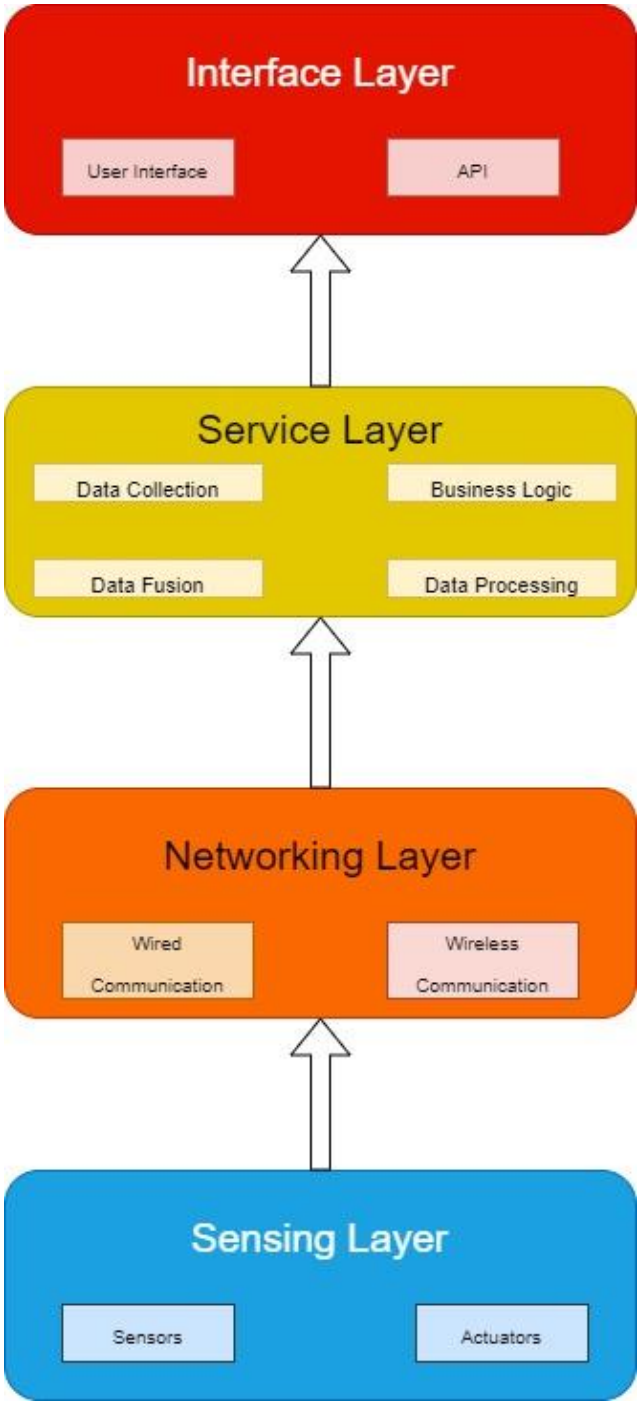

Figure 1: The service-oriented architecture of the IoT

Each layer has clearly defined responsibilities:

- **The Sensing Layer** is comprised of sensors and actuators, which either provide information about their environment to entities in proceeding layers or allow them to change the physical world in some way.
- **The Networking Layer** consists of the protocols which enable IoTDs in the sensing layer to communicate with each other and entities in the service and interface layers.
- **The Service Layer** implements the business logic of IoT applications. It is responsible for informing services which IoTDs can provide them with data they require, assembling services together to achieve higher order functionality, providing reputation mechanisms among services to establish trust, and providing APIs and middleware which ensure services can interoperate.
- **The Interface Layer** provides APIs through which applications can communicate with each other and user interfaces which allow users to interact with the service and sensing layers.

This architecture illuminates the higher-level design of any number of IoT applications, from smart homes to industrial grids.

2.3 IoT Application Domains

IoT applications adhere to common principles in terms of design and functionality, but there is a great deal of variation among different IoT application domains. When considering the varying landscapes of different IoT domains, such as smart homes, the smart grid, and healthcare, the numerous security concerns and vulnerabilities of these systems become apparent.

2.2.1 Smart Homes. Smart homes are IoT environments where sensors and actuators are embedded in appliances, home goods, furniture, etc. By integrating these devices with the service layer and interface layer, the home environment can be automated, providing various utilities to users. (Domingo, 2012) states that a smart home can provide an occupant with recipes tailored around their dietary requirements, taking health conditions like high

cholesterol and diabetes into account, and based on the groceries they currently have in the house. Smart homesalso provide a user with the ability to control their home appliances remotely, an option which has gained commercial success recently with products such as Google Home and Amazon Alexa.

Smart homes can facilitate significant quality of life improvements through facilitating greater independence for the elderly and those with disabilities. (Ghayvat *et al.*, 2015) presented an IoT system in which they integrated sensors in a home which monitored the activity of the inhabitants. Sensor readings from the smart home were used to extrapolate the resident's wellbeing and notify caretakers if there was cause for concern. Similarly, (Domingo, 2012) describedthe potential for an "object search engine" to aid blind people locate objects in their home and traverse physical obstacles.

Smart homes provide economic and environmental benefits to their users. (Orsi and Nesmachnow, 2017) designed a system that significantly reduced the energy usage of home appliances. This was accomplished by monitoring the energy utilization of a device and optimizing its operation in achieving a desired user state. In an experiment, their system reduced the energy consumption of a water heater by nearly 40% by scheduling cycles in which the device would shut off.

2.2.2 Smart Grid. The smart grid is an emerging application of the IoT which integrates urban utility systems, such as electric grids and water systems. This allows two-way communication between the consumer and supplier of a utility, enabling increased efficiency for both parties (Al-Turjman and Abujubbeh, 2019). For users, the smart grid provides increased control over the amount of money they spend each month on utilities by enabling real time updates on energy consumption (Al-Turjman and Abujubbeh, 2019). On the supplier side, the smart grid enables the identification points of inefficiency and waste within

their infrastructure. The city of Tokyo integrated their water supply into the IoT and identified areas where water was being leaked. This resulted in an estimated $170 million saved annually (Shah and Yaqoob, 2016). Similarly, the smart grid enables rapid identification of the source and cause of power outages, allowing for faster repairs (Joseph and Jasmin, 2015).

2.2.3 Healthcare. The IoT has been integrated into healthcare to ensure favorable medical outcomes for patients. Wearable technology can detect negative health events, such as falls (Wu *et al.*, 2017), alerting caregivers when assistance is needed. Commercial wearable technology, such as smart watches, have recently gained popularity in allowing users to track various metrics regarding their health, like steps, heart rate, blood pressure, and the duration and quality of their sleep (Dinh-Le *et al.*, 2019). (Dinh-Le *et al.*, 2019) state that there is opportunity for healthcare providers to use this datato better serve their patients. By making these metrics visible to healthcare providers, it would bepossible for them to notice concerning trends in data gathered over a longer period than practicalfor regular physicals.

2.4 IoT Security

It is clear that the IoT often generates, stores, processes, and communicates sensitive data. (Dinh-Le *et al.*, 2019) note that medical data is subject to legally enforceable guidelines, such as HIPAA in the United States, concerning an individual's right to privacy. Failure to properly protect the confidentiality of data in the IoT could result in legal repercussions for companies producing IoT solutions within healthcare.

The IoT also performs critical tasks. Disruptions to the operation of a smart grid could negatively impact an area's economy by preventing businesses from operating withoutaccess to necessary resources. More alarming, disruptions to the functioning of some IoT applications could endanger human users, such as systems which are critical to the care of those

with disabilities or the elderly (Ghayvat *et al.*, 2015). The SOA for the IoT outlines the need for trust mechanismsto be established at the service layer (Xu, He and Li, 2014), but no such considerations are made concerning communications with the sensing layer.

Well established, mature security protocols exist for traditional computer networks, these protocols are often inapplicable to the IoT (Sinche *et al.*, 2020). Traditional network hosts, such as desktop computers and laptops, are often seen as having virtually unlimited memory and computational power from the point of view of a security engineer, and security mechanisms in these environments can be both complex and multi-layered (Sinche *et al.*, 2020). In contrast, the definition of an IoTD presented by (Miorandi *et al.*, 2012) makes it clear that IoTDs can have extremely limited computational resources in comparison to traditional network hosts. These limitations in terms of power, memory, and CPU speed often makes traditional security measures entirely unusable on IoTDs (Sinche *et al.*, 2020).

(De Cannière, Dunkelman and Knežević, 2009) identified three design considerations that should be taken into account when creating security mechanisms for IoTDs. First, the mechanism must have a small footprint. They note that in some situations, the addition of a single logic gate could be the deciding factor between a security mechanism being included in an IoTD and the mechanism being discarded entirely. According to (Beaulieu *et al.*, 2015), different IoTDs could have different factors deciding whether a mechanism's footprint is too big. Some IoTDs may present an environment in which a mechanism designed for a hardware implementation is preferred while others may require a mechanism optimized for software performance. Second, the mechanism must have low power consumption. Many IoTDs rely on batteries or other external power sources, meaning security mechanisms which significantly increase the device's power draw could be intolerable. Third, the mechanism must provide good overall runtime efficiency. IoT systems are often time sensitive and require real-time or low

latency data streams. If the security mechanism in question significantly slows the flow of data from the device, it could be rendered unusable. These factors force design engineers to leave out security mechanisms in the IoT domain or choose mechanisms that offer a compromise between security and performance (Beaulieu *et al.*, 2015).

The situation is further complicated by the fact that even the best security mechanisms are rendered ineffective by poor user compliance (Mwagwabi, McGill and Dixon, 2018). (Iqbal *et al.*, 2020) reported that the iBaby M3S wireless baby monitor was subject to unauthorized access simply due to the default username and password from the manufacturer both being set as "admin". While one could hope that users would comply to best practices concerning setting and protecting their usernames and passwords, current data on the topic does not leave much room for an optimistic outlook (Mwagwabi, McGill and Dixon, 2018).

Ironically, it is the computational resources represented by the device, no matter how limited, that attackers are after. The concept of a botnet has recently emerged and refers to an attacker gaining remote control of mass quantities of poorly secured IoTDs, then using their combined resources to carry out tasks that would otherwise be far too complex for them individually (Kambourakis, Kolias and Stavrou, 2017). Botnets have been able to amass armies of up to half a million infecteddevices, using them to launch some of the most devastating DDoS attacks in history.

3 BOTNETS

As previously discussed in the introduction, Botnet is a network of infected computers responsible for carrying out distributed attacks (Amini, Araghizadeh and Azmi, 2015). In this section, we delve into exploring the architectures and configuration of the botnet and its lifecycle.

3.1 Configuration

Various configurations or topologies that can be summarized into below four categories are described in (Amini, Araghizadeh and Azmi, 2015; Mahmoud, Nir and Matrawy, 2015). In this star or Centralized C&C Topology, there is a single server (botmaster) that communicates with all the botnet members. Failure of the server can cause disruption of the botnet. Legacy botnets are based on this type of architecture but, recently shift has been made towards P2P architectures. To make botnet fault-tolerant, instead of a standalone server, multiple servers work in tandem to coordinate attacks, malware distribution, and weak system identification by sending commands to bots in Multi-Server C&C Topology. Hierarchical C&C Topology as the name implies has a central botmaster that controls bot and then the bot assumes the role of botmaster and in turn controls its botnet members and so on (as shown in Figure 2). In Peer-to-Peer topology there is no central server, instead, every bot member is capable enough to do tasks of a botnet. In other words, every bot is a bot as well as a botmaster.

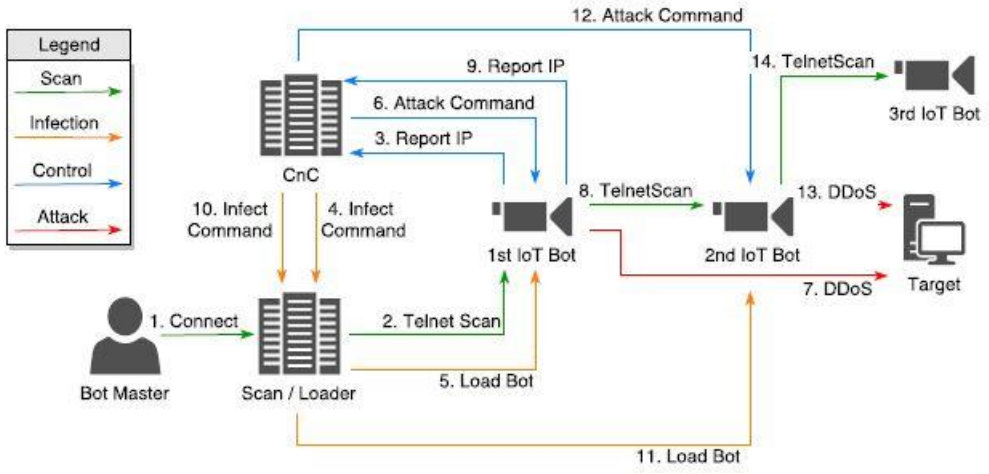

Figure 2: Hierarchical C&C Topology (McDermott, Majdani and Petrovski, 2018)

3.2 Architecture

Botnets are broadly classified based on the protocol used by command-and-control server into IRC-based, HTTP-based, DNS-based or Peer to Peer (P2P) botnets. There are also some lesser-known botnet categories like POP-based botnets for email attacks, edge devices botnets like SMS and MMS-based botnet and social network botnets as specified in (Mahmoud, Nir and Matrawy, 2015).

An IRC-based botnet used Internet Relay Chat Protocol (IRC) (Wang *et al.*, 2009). An IRC network is made up of multiple servers that work in co-ordination to relay messages across servers. Each server has multiple channels that have published topics. A user connects to one of the IRC servers with a UNIQUE Id and then uses one of the channels to communicate. (Wang *et al.*, 2009) explains that a channel in an IRC network is a bot (malware) and all the systems infected by the same malware belong to the same channel. On

the other hand, an HTTP-based botnet (also known as Web-based botnet) relies on HyperText Transfer Protocol (HTTP) for its communication with the bot members. (Hsu *et al.*, 2017) specifies that most of the internet traffic is HTTP traffic and HTTP-based botnets clearly take advantage of this and disguise themselves as normal traffic. Compared to IRC-based botnet (Wang *et al.*, 2009), (Hsu *et al.*, 2017) explains HTTP-based botnet are relatively difficult to detect primarily for two reasons: first they hide behind normal traffic and second, most of the firewall/proxies do not have the capability for deep packet inspection (examine network packet at each layer).

A DNS-based botnet utilizes the Domain Name System protocol (DNS) useful for contacting the C&C server by issuing DNS queries. DNS systems are a directory of IP addresses responsible for converting Domain Name into IP addresses (also called as domain name resolution) and vice- versa. (Singh, Singh and Kaur, 2019) delineates that the bots and botmaster communicate by using DNS queries and in order to avoid detection, the C&C server keeps changing the domain name by using Domain Generation Algorithm (DGA) or fast-flux which makes them robust to detection. Newly generated IP and domain names are constantly being updated in the DNS system. In comparison to IRC-based, HTTP-based, and DNS-based, which uses client-server architecture, P2P- based botnet uses peer to peer network in a distributed fashion which makes them robust for detection. Since, there is no single master, every bot in the network is a master and capable of infecting vulnerable systems and carry out the attack. (Su *et al.*, 2018) examines various P2P traffic types and concludes that P2P-based detection is difficult owing to its inherent distributed nature.

4 BOTNET ATTACKS AND TYPES

4.1 Lifecycle

Botnet lifecycle comprises of five stages as specified in (Feily, Shahrestani and Ramadass, 2009; Amini, Araghizadeh and Azmi, 2015; Mahmoud, Nir and Matrawy, 2015). In the initial Infection stage, the C&C server scans the network and looks for vulnerabilities in the network, servers, and system. Obvious flaws like buffer overflow, backdoors, incomplete mediation, password guessing on SQL servers are done. Once the weak systems are identified, they are targeted with a shell script and run on it in the secondary infection stage. The shell scripts enable the systems to download malware or bot binary codes from the C&C server. In the connection stage, once the malware is run on the host system, a connection is established to the C&C server and the botmaster can now send the commands to the system and is now a part of the botnet. In Malicious Command and Control phase, the C&C server sends attack commands to the botnet members to disrupt online services. The update and maintenance phase is an ongoing process that is required as a C&C server in order to avoid detection it keeps migrating the server.

4.2 Attack Methods

Botnet attacks are motivated by various reasons like economic, political and ideological considerations. Sometimes, extortion or ransom, personal feuds, naïve enthusiasts or script kiddies and cyber warfare could be the possible reasons for such attacks.

4.2.1 Denial of Service (DoS) or Distributed Denial of Service (DDoS):

Denial of service is a kind of attack in which the target resource is inundated with many requests which overwhelm the target making it unavailable to the user. Generally, DoS is executed by a single machine and it is not capable enough to bring down the target system. Practically, multiple machines are required to launch such attacks, hence DDoS attacks are ubiquitous and difficult to break down (Feily, Shahrestani and Ramadass, 2009; Amini, Araghizadeh and Azmi, 2015). (Douligeris and Mitrokotsa, 2003) broadly classifies DDoS attacks into three categories: volume-based attacks, protocol attacks, and application-layer attacks. Volume-based attacks include UDP flood and ICMP flood. A UDP flood is any DDoS attack that uses User Datagram Protocol (UDP) packets to bring down the network by rapidly sending spoofed UDP packets to the host. The host in response sends error messages since spoofed packets do not have any legit connections. ICMP (Internet Control Message Protocol) flood is also known as a ping attack in which diagnostic tools like traceroute or ping are used to check the device health and connectivity. Many ping requests will require the same number of responses and make the device crash.

4.2.2 Miscellaneous Attacks:

The Spamming and Traffic Monitoring attack include bots that are used as a sniffer to steal sensitive data like usernames and passwords from the infected machines. Internet users are targeted with spam emails that are sent out in bulk using botnets such as Grum responsible for 25% of the total spam emails (Nagamalai, Dhinakaran and Lee, 2007). With the help of botnet, bots can spread out keyloggers which are software that captures key sequence presses on the keyboard and designed in such a way that gets activated when popular keywords like PayPal and Yahoo are entered (Sagiroglu and Canbek, 2009). Spam emails are disguised as legit emails that direct users to legitimate websites to

enter bank details, tax details, personal details, and card details. Such information is traded on the dark web for fees. Pay-per-click is one of Google's AdSense programs in which various websites display Google advertisements and earn money when the user clicks on those ads (Cole, Mellor and Noyes, 2009). Botnet, apart from carrying out attacks, also has the potential to propagate botnet over various geographical regions by targeting less secure systems. Adware is harmless ads but in disguise collects browser data by using spyware software. It is used to lure users into clicking false advertisements or apps with the pretext of monetary or personal profits (Cole, Mellor and Noyes, 2009).

5 RELATED WORKS IN BOTNET DETECTION METHODS

In this section, we will dive deep into the traditional botnet detection method, state of the art detection method followed by an amalgamation of machine learning, and deep learning techniques for detecting botnet.

5.1 Classical Methods

A honeypot is a computer system that is placed in the DMZ (Demilitarized Zone) network of the company that is used to lure the attackers into attacking it (Douligeris and Mitrokotsa, 2003). A honeypot is a vulnerable system and any communication between honeypot and outside the system is considered suspicious. (Feily, Shahrestani and Ramadass, 2009; Amini, Araghizadeh and Azmi, 2015) broadly studied the concept of honeypot and identified that the system used as honeypot does not have any production value. A typical honeypot captures information such as the signature of the bot (malware), C&C server mechanism, botnet details, techniques used by the attacker, attacker motivation and most importantly, the loophole of the system that bot exploited. Apart from discussing the simplicity of the model, (Feily, Shahrestani and Ramadass, 2009; Amini, Araghizadeh and Azmi, 2015; Mahmoud, Nir and Matrawy, 2015) also enlightened on the limitations associated with the honeypots. Honeypots (or Honeynet) have limitations in detecting several exploitations, bots that use propagation, cannot scale to other malicious attacks, and it can only generate a report of the attack on the honeypot system. It cannot detect the attack in real-time as well as bot attacks on some other system.

5.2 Signature and Anomaly Based

Intrusion Detection System (IDS) is the emerging field in botnet detection as an alternative to Honeypots. IDS systems are classified as signature-based and anomaly-based detection. Recent works in botnet detection, utilizes the different mechanisms of anomaly-based detection as compared to signature-based detection. Signature-based IDS maintains a database of attack signature and uses it to scan and compare the incoming traffic against the available signatures. Immediate detection and zero false-positive rates as noted in (Zeidanloo *et al.*, 2010) are the benefits of signature-based IDS. They are useful in the sense the exact cause of the attack is also known in IDS Response and the network administrators can take appropriate steps by quarantining the segment of the network or the infected system. (Douligeris and Mitrokotsa, 2003) notes some of the associated disadvantages like detection time decreases as the signature database increases in size, the signature needs to be updated on daily basis, cannot detect variant of the botnet attack and cannot identify zero-day attacks or new botnet. (Hoang and Nguyen, 2018) renders signature-based IDS as ineffective owing to modern botnets being equipped with advanced code patching and dodging techniques. Due to its inherent nature of being static, anomaly-based detection techniques are widely implemented.

Research has shown that anomaly-based techniques are much better at detection as it does not require a database of signature and is capable enough to detect new or unknown botnets. Anomaly- based techniques try to detect botnet by using various network characteristics like network protocols, packet size, stateful and stateless features, traffic size, unusual system, and abnormal behaviors. They are implemented as either host-based IDS or network-based IDS as explained in (Zeidanloo *et al.*, 2010). A host-based IDS is implemented

on the system and is unaware of the network traffic whereas a network-based IDS is unaware of host network traffic. Anomaly-based detections detect behavior that goes out of normal behavior. Due to this feature, Anomaly-based IDS is effective in identifying new or variant of existing botnets in real-time. Surveys on Anomaly-based IDS in (Feily, Shahrestani and Ramadass, 2009; Zeidanloo *et al.*, 2010; Amini, Araghizadeh and Azmi, 2015) has confirmed that anomaly-based IDS cannot be used as a foolproof measure of detecting botnet because as it is difficult to identify the normal threshold and also the network activity changes over time and hence, the IDS has to evolve. Also, such systems cannot effectively reveal the specific type of attack which makes it difficult to block the botnet. Hence, hybrid systems of Anomaly and Signature-based implementation are widely used in insecure networks.

5.3 Machine Learning-Based

The field of machine learning and deep learning is now the most widely implemented and experimented area. (Hoang and Nguyen, 2018; McDermott, Majdani and Petrovski, 2018; Meidan *et al.*, 2018) have described various techniques of botnet detection using machine learning or deep learning in combination with botnet characteristics.

Botnet detection using machine learning techniques like k-NN (k-Nearest Neighbor), Decision Tree (DT), Random Forest (RF), and Naïve Bayes model based on DNS Query data is mentioned in (Hoang and Nguyen, 2018). Bots of the botnet receive code and commands from the C&C server by performing lookup queries generated using DGA or fast-flux. (Hoang and Nguyen, 2018) identifies that the IP address of the C&C server is not a legit name and keeps on randomly changing to avoid detection. Also, the generated malicious domain names have characteristics like DNS, network and lexical features entirely different from benign

domain names. (Hoang and Nguyen, 2018) approaches to solve the problem by collecting 16 vocabulary features from 2-g and 3-g clusters like mean, variance, standard deviation, entropy, consonants, vowels, number, and character characteristics and 2 characteristics from vowel distribution. IP addresses are random with datasets generated from Conficker, and DGA botnet (malicious) and top domain names from Alexa Internet (benign) (collection of domain names) in conjunction with machine learning models, (Hoang and Nguyen, 2018) demonstrated the effectiveness of Random Forest machine learning model by delivering an accuracy of 90.80% in botnet detection. Also, the proposed random forest-based detection system suffered from a relatively high false-positive rate. Some related research papers to (Hoang and Nguyen, 2018), (Ramachandran, Feamster and Dagon, 2006) implemented techniques for detecting botnets by administering the DNSBL (Domain Name System Blacklist) list which contains IP addresses of spam bot members and matching it with DNS queries domain names. (Villamarin-Salomon and Brustoloni, 2008) extends its work on the idea that most of the DNS queries are made to terminated domains or NXDOMAIN (non-existent domains) as the C&C server keeps on changing the IP addresses and it was successful in detecting traffic with a large number of queries and terminated domains.

Botnet detection models in the works of literature are heavily built for network and network devices as discussed in (Ramachandran, Feamster and Dagon, 2006; Villamarin-Salomon and Brustoloni, 2008; Hoang and Nguyen, 2018). Work expressed in (Doshi, Apthorpe and Feamster, 2018) proposes a detection model for the Internet of Thing (IoT) devices by highlighting the fact that IoT specific behaviors are unique like they have few endpoints for connections and the time interval between the packet is well regulated. It postulates that botnets such as Mirai are exploiting insecure IoT devices to carry out DDoS

attacks in large numbers. A survey in (Amini, Araghizadeh and Azmi, 2015) predicts that by 2020, the number of IoT devices will be around 20 billion and 10% of these devices have flaws ranging from un-encrypted data transmission, outdated BIOS firmware, and exposed telnet ports. (Doshi, Apthorpe and Feamster, 2018) considers Random Forest, k-nearest neighbors, Decision Trees, SVM and neural network and train these models on network flow parameters like length of the packet, size of the interval and protocol used. The detection pipeline is flow-based (considers network behavior), uses either stateless or stateful features and is protocol-agnostic (robust to different protocols). This paper expresses the issues surrounding IoT devices like lightweight characteristics, limited memory, and computation power. Steps involved in building an IoT-based detection model is capturing the traffic, grouping the packets based on device (stateful) and by time (study temporal patterns), extracting stateful and stateless features followed by a binary classification. It experimented with the dataset and found out that normal traffic packet size varies between 100 to 1200 bytes whereas attack traffic is under 100 bytes owing to repeated attacks. Also, the attack traffic has a lesser inter-packet interval in comparison to normal traffic. Most of the time protocols used during attack were TCP as opposed to UDP during normal situations. The classifiers were able to obtain training accuracy from 0.91 to 0.99. However, though the accuracy was 0.99, it was built on simulated data generated using DDoS attack. (Doshi, Apthorpe and Feamster, 2018) predicts the model might be overfitting the data but is unsure of its performance on real-time attacks which opens the door for future research in the detection of botnets.

Work was done in (Doshi, Apthorpe and Feamster, 2018; Meidan *et al.*, 2018) that focused on detecting botnet using IP address details and traffic flow characteristic, respectively. On the other hand, (Ramachandran, Feamster and Dagon, 2006) focusses on

leveraging the detection of botnet using an efficient flow-based technique by reducing the packet size and time of traffic flow under consideration. The model was developed to detect two P2P botnets namely Storm and Waledac botnets during the honeynet project. It has been noted in many papers that modern botnets are resilient to detection by employing techniques like protocols obfuscation, encrypted communication, fast-flux and random domain name generation using DGA. P2P botnets have a disastrous effect on industrial systems and their infrastructures. In order to train the models, (Ramachandran, Feamster and Dagon, 2006) captured network traffic of five tuples like source IP address, source port, destination IP address, destination port and protocol used. As well for every traffic, 39 other statistical features were extracted. It employed batch analysis and limited analysis of the captured traffic by using eight different machine learning algorithms like Naïve Bayesian Classifier (NB), Logistic Regression (LR), Bayesian Network Classifier (BNet), Linear Support Vector Machine (LSVM), Neural Networks, Random Forest Classifier, Random Tree Classifier, and Decision Tree Classifier. In the modeling process of botnet detection, all MLA (machine learning algorithms) delivered impressive performance except Naïve Bayesian Classifier for both malicious as well as non- malicious traffic. The tree classifiers delivered promising classification performance, but Random Forest Classifier delivered the highest accuracy. Remaining MLAs, delivered the worst performance for non-malicious traffic as compared to normal traffic since the dataset was skewed in the former case. Additionally, this paper was successful in stating that high performance in detection was obtained by monitoring the traffic flow for only 60 seconds with an accuracy of 95%. Also, the initial 10 packets per flow are evident enough to detect botnet as opposed to monitoring the entire flow.

To summarize, (Doshi, Apthorpe and Feamster, 2018) experimented with detecting botnet on consumer internet of thing device-based attack. (Meidan *et al.*, 2018) considered a deep learning model, whereas (Doshi, Apthorpe and Feamster, 2018) demonstrated the effectiveness of k-NN, Linear Support Vector Machine (LSVM), Decision Tree, Random Forest, and Neural Network by achieving an average accuracy of 99%. It also considered stateful features like bandwidth, and IP destination address cardinality and novelty and stateless features like packet size, inter-packet interval and protocols separately and together during the training phase. (Stevanovic and Pedersen, 2014) employed flow-based machine learning technique for botnet detection and experimented with models like Naïve Bayes, Bayesian Net, Artificial Neural Network, Support Vector Machine, Random Tree, Random Forest, and Decision Tree. The flow-based model was able to achieve accurate detection of traffic only for 10 packets and 60 seconds of the flow. Next, we will consider the advancements done using Deep Learning for botnet detection.

5.4 Deep Learning-Based

Deep Learning is a research field and is constantly evolving. Recent researches in the detection of the botnet have shifted its gear towards building the model using Deep Learning techniques. Autoencoders, Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN) have played a leading role in blocking attacks of a botnet. Next, we describe deep learning models developed for the detection of the botnet.

For botnet detection, (Hoang and Nguyen, 2018) used a supervised learning model whereas (Meidan *et al.*, 2018) used an unsupervised learning model. Most of the detection models are built for botnet on the network, (Meidan *et al.*, 2018) focused on IoT devices that

are more vulnerable to botnet attacks since they are relatively less secure owing to their size and computation requirements. This paper studies the effect of Mirai and BASHLITE botnet attack on IoT devices by capturing network snapshots of traffic behavior emanating from malware attacked IoT devices. It promotes building a model that expedites the alerting process which can effectively help in quarantining the device as soon as possible and specifically for large scale enterprise networks where the number of connected IoT devices through Wi-Fi (Wireless Fidelity) and Bluetooth is extremely large. The author develops a novel network-based model using deep learning-based autoencoder for each device which delivers a lower false-positive rate. The autoencoder is trained on benign network behavior (snapshots) and tries to compress it, called an encoding phase. During the decoding phase, it tries to re-construct the snapshot, and failure to do so means an anomalous behavior is observed. (Stevanovic and Pedersen, 2014; Doshi, Apthorpe and Feamster, 2018; Hoang and Nguyen, 2018) focus on initial stages of infection, (Meidan *et al.*, 2018) operated on the attack stage which provides benefits like heterogeneity tolerance (separate encoder for each device), efficiency (trains on a batch of observation and then discards it) and Open World (detect abnormal behavior). To build the model, 23 features were obtained from 115 traffic statistics over the various time frame of varying minutes. Throughout the training process, hyperparameter learning was also performed followed by learning of threshold for abnormal behavior and effective window size. TPR for autoencoder was 100%, though it raised few false alarms and required the smallest detection time for most IoT devices. The deep autoencoder failed in the scenarios because it tends to fit common patterns.

A novel approach in the detection of the botnet by using graph-based features is discussed in (Stevanovic and Pedersen, 2014). It employs the detection of malicious hosts by

evaluating the temporal activity of the infected device or network activity across a fixed interval and overlapped windows. It notes that 70% of all the spam is controlled by botnets and the newer botnets use peer-to-peer architecture in comparison to centralized architecture. Moreover, they make detection of botnet more difficult by changing their peer-to-peer protocols and encrypt their packet data. (Ramachandran, Feamster and Dagon, 2006) adopts a flow-based approach and are often specific to a botnet and not generalizable. On the contrary, (Stevanovic and Pedersen, 2014) uses graph-based model that avoids the sequential characteristic of data and focusses on the communication structure of the node. The model considered; Long Short-Term Memory (LSTM) is well suited for time-series evaluation of parameters. In this case, LSTM gives True Positive Rate (TPR) of 0.946 which is better than the state-of-the-art TPR but delivers a slightly higher False Positive Rate (FPR) of 0.037. It takes advantage of botnets' activity and dormancy tendency and extracts ten graph features for an interval of 300 seconds long and with an overlap of 150 seconds with the previous interval. It widely experimented on the CTU-13 dataset and took care of the imbalance problem during the pre-processing phase by performing down-sampling by a ratio of 10:1: non-malicious: malicious. By hyper-parameterizing step-size and window-size for the scenarios 6, 7, 10, 11 and 12, leverage LSTM’s property to remember previous values. However, this paper suffers from limitations like a major chunk of training goes in pre-processing of data using graph construction.

(Jung *et al.*, 2020) offered a CNN-based deep learning model consisting of a data handling component and an 8-layer CNN. Until implementing the CNN model, they segmented and standardized the energy utilization data obtained, to help the CNN model to attain greater precision. The model categorizes handled data into four classes, including the

botnet class, which is the prime objective. To show results, they conducted a self-evaluation; a cross-device assessment; and leave-one-device-out and leave-one-botnet-out examinations on three conventional kinds of IoT devices—a security camera, a router and a voice assistant. The self-assessment reached a classification accuracy of 96.5%, and cross-tests attained approximately 90% accuracy. In the same manner, leave-one-out tests attained more than 90% accuracy for botnet identification.

To summarize the techniques, (Doshi, Apthorpe and Feamster, 2018; Hoang and Nguyen, 2018; Meidan *et al.*, 2018) considered only the current behavior of the network and (Sinha, Viswanathan and Bunn, 2019) employed temporal evolution of the network activity of the behavior for botnet detection by using graph-based techniques. Tracking temporal activity is best done by using LSTM (Long Short-Term Memory) based neural network architecture. The botnet detection model developed in (Stevanovic and Pedersen, 2014) is robust to botnet architecture, insensitive to botnet characteristics and generalizable to other botnet attacks. Despite detection models being made more robust, attackers still try to develop code evasion techniques and are evading machine learning model based detection by studying their prediction pattern. (Wu *et al.*, 2019) proposes a reinforcement based deep learning model that generates fake traffic to learn and deceive the detection model. Building a detection model helps an attacker to build a model that can avoid detection. On the other hand, this motivates researchers in developing more robust models to overcome such hacks of evading detection.

6 RESEARCH METHOD

6.1 Research Approach

The method for IoT botnet attack detection proposed in this work is motivated by the advances in botnet attack detection using machine learning (Ramachandran, Feamster and Dagon, 2006; Doshi, Apthorpe and Feamster, 2018; Hoang and Nguyen, 2018) described in the previous chapter. While the deep learning-based approaches (Meidan *et al.*, 2018; Jung *et al.*, 2020) show particularly high accuracy, the computational complexity of these algorithms calls into question their viability on hardware commonly available on access networks. This research focuses on the developing a botnet attack detection system using Deep residual CNN algorithms which achieves high accuracy compared to the existing deep learning algorithms.

The system uses three deep learning model namely ANN, CNN-LSTM and Deep Residual CNN to detect intrusion from IoT devices. The system was tested by employing real traffic data gathered from nine commercial IoT devices authentically infected by two common botnet attacks, namely, Mirai and BASHLITE. The system was set to recognize zero-day attacks from IoT devices to identify well-known attacks. Figure 3 shows the system architecture of the developing system. The main components of the proposed system are described in the next section.

Figure 3: The proposed system

6.2 Dataset

The N-BaIoT dataset was collected from a machine-learning repository. The dataset utilized in this study includes the statistics of network traffic captured in a lab environment in which the typical normal behaviour and attack cases are simulated (Meidan *et al.*, 2018). The network contains nine IoT devices belonging to different application categories such as security camera, webcam, baby monitor, thermostat, and door-bell (as shown in Table 2). The malicious traffic includes the attacks launched by IoT devices compromised by Bashlite and Mirai malware. The N-BaIoT dataset contains the features extracted from the raw IoT network traffic data. More specific, whenever a packet is received, a behavioral snapshot of the protocols and hosts that transmitted each packet is computed. Each snapshot corresponds to the packet's contextual information as reflected in a set of statistical features, i.e., the arrival of each packet invokes the extraction of 23 statistical features from five-time windows (100ms, 500ms, 1.5sec, 10sec and 1min), and then five 23-dimensional vectors from each window are concatenated into a single 115-dimensional vector (in the rest of the text we will refer to the 115-dimensional vector as instance).

The traffic aggregation is performed in five major feature categories, host-IP, host-MAC&IP, channel, network-jitter, and socket. These categories and features (with the statistical methods used for producing the relevant feature) are shown in Table I. Statistics of the outgoing network traffic originated from the same IP are given in Host-IP category. Host-MAC&IP covers the traffic having the same MAC and IP addresses. Channel category includes the network statistics determined by source and destination hosts whereas socket also covers source and destination ports beside their IP addresses. Network-jitter category covers the time intervals between packet arrivals of channel type communication. Packet counts,

mean and variance of packet sizes, are included in all categories. Additionally, more detailed statistics such as magnitude, radius, covariance and correlation coefficient of packet sizes are given for channel and socket categories. The classifiers are based on three labels, normal, Bashlite (gafgyt) or Mirai. The dataset has 502,605 normal, 2,835,317 bashlite and 2,935,131 mirai records, meaning that the label distributions are approximately 8%, 45%, and 47%. Further details and the datasets themselves are publicly available at https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT.

Table 1: Feature Categories

Feature Categories	**Features**
Host-IP	Packet count, mean and variance (outbound)
Host-MAC&IP	Packet count, mean and variance (outbound)
Channel	Packet count, mean and variance (outbound) Magnitude, Radius, Covariance, Correlation Coef. (Inbound and outbound)
Network Jitter	Count, mean and variance of packet jitter in channel
Socket	Packet count, mean and variance (outbound) Magnitude, Radius, Covariance Correlation Coefficient (inbound and outbound)

Table 2: IoT devices used for developing datasets

Device type	**Devices used in the model**
Doorbell	Danmini Doorbell Ennio
Thermostat	Ecobee
Baby monitor	Phillips B120N/10
Security camera	Provision PT-737E Provision PT-838 Simple Home XCS7-1002-WHT Simple Home XCS7-1003-WHT
Web camera	Samsung SNH1011N

In this research, three deep neural networks were used to identify cyberattacks on a Provision PT-737E Security Camera. Table 3 summarizes attack types in the dataset, including two common botnet attacks, namely, BASHLITE and Mirai. BASHLITE attacks, one type of botnet attack representing DDoS attacks, were developed using C programming for infecting Linux systems. This attack is the most common botnet attack that infects IoT devices, such as cameras. In contrast, Mirai botnet attacks, discovered in 2016 by Paras, use malware run on ARC processors to infect large-scale IoT networks.

Table 3: Botnet attacks

Major attacks	Sub attacks	Description
BASHLITE	Junk	By sending spam data
	TCP flood	Sends flood of request
	UDP flood	Sends flood of request
	Scan	Scans the network for victim devices
	COMBO	Opens connection IP address and network port by sending spam data
Mirai	ACK	Sends flood of acknowledgment
	SYN	Sends synchronize-packet-flood
	Plain UDP	Sends flood of request
	UDP flood	UDP flood by optimizing seeding packet per second
	Scan	Scans the network for victim devices

6.3 Deep learning algorithms

Deep learning is one of the artificial intelligence algorithms used to handle analysis, complex processes, and big data. The deep learning model is applied to detecting botnet attacks from an IoT environment. In this proposed research, three deep learning models were applied to identify and classify botnet attacks from different IoT devices.

6.3.1 Artificial Neural Networks (ANN). An Artificial Neural Network (ANN) is a mathematical model that tries to simulate the structure and functionalities of biological neural networks. Basic building block of every artificial neural network is artificial neuron, that is, a simple mathematical model (function). Such a model has three simple sets of rules: multiplication, summation and activation. At the entrance of artificial neuron, the inputs are

weighted what means that every input value is multiplied with individual weight. In the middle section of artificial neuron is sum function that sums all weighted inputs and bias. At the exit of artificial neuron, the sum of previously weighted inputs and bias is passing through activation function that is also called transfer function (as shown in Figure 4).

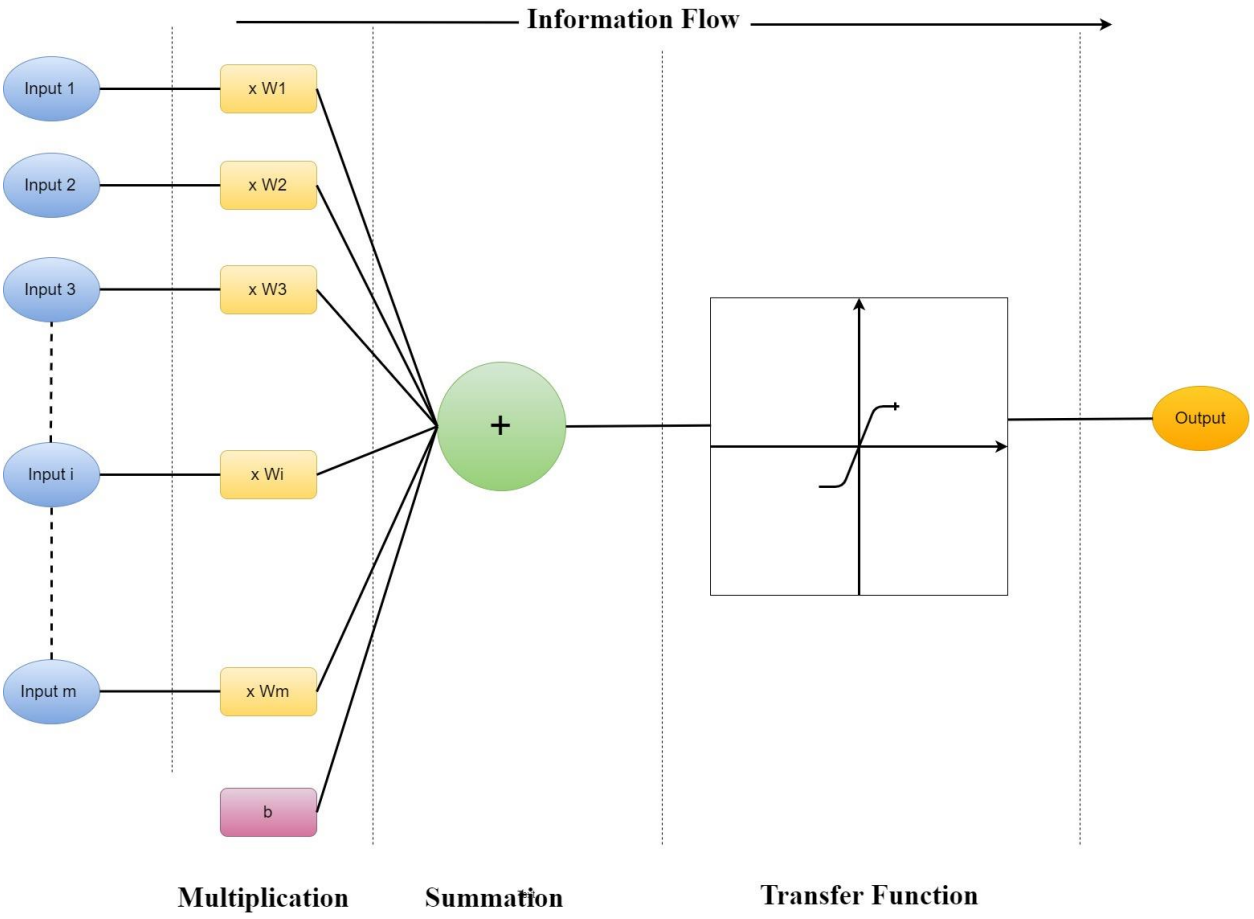

Figure 4: Working principle of an artificial neuron

The ANN algorithm used for this research has four hidden layers, and the “Sequential” module from the Keras library is used to create a sequence of ANN layers stacked one after the other. Each layer is defined using the “Dense” module of Keras where we specify how many neurons would be there, which technique would be used to initialize the weights in the

network and what will be the activation function for each neuron in that layer etc. The input shape of the data is (115,1). The output shape of each hidden layer and its parameters are shown in Figure 5. The output layer has shape of (11,1), thus providing 11-class classification of the botnet attacks.

A rectified linear (ReLU) is a nonlinear activation function used to apply the element-wise activation function for the first two dense layers. The ReLU function returns 0 for negative values, and for positive values, it returns any value x. The ReLU function has a range from 0 to infinity (Zoph and Le, 2018):

$$ReLU(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \quad (1)$$

The Softmax function is used as the activation for the last layer of a classification network because the result could be interpreted as a probability distribution. It converts a vector of values to a probability distribution. The elements of the output vector are in range (0, 1) and sum to 1. Each vector is handled independently (Kouretas and Paliouras, 2019).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 10)                1160
_________________________________________________________________
dense_1 (Dense)              (None, 40)                440
_________________________________________________________________
dense_2 (Dense)              (None, 10)                410
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 11
_________________________________________________________________
dense_4 (Dense)              (None, 11)                22
=================================================================
Total params: 2,043
Trainable params: 2,043
Non-trainable params: 0
_________________________________________________________________
```

Figure 5: Model summary of ANN model

6.3.2 Convolutional Neural Networks (CNN) - LSTM. CNN is a deep learning algorithm that is used to build an efficient system for image classification. However, the CNN model can also help design efficient systems for security purposes. The CNN algorithm is similar to the ordinary neural network: the CNN algorithm consists of four main layers, namely, the input layer, convolutional layer, pooling layer, and fully connected layer (Yao *et al.*, 2021).

- **Convolutional Layer**. The convolutional layer is used to explore, size, and filter the training sample, including numerous filters known as convolution kernels. The convolutional layer develops the weight matrix for the input sample and recodes the weighted summation kernel layer. The filter is integer values that are used to subset the input pixel values. Three significant hyperparameters, such as filter size, stride, and zero padding, play

roles in increasing the performance of the convolutional kernels, choosing appropriate values that can help reduce the complexity of the neural network and increase the performance of the system. Figure 6 shows the details of the CNN algorithm layers. The input shape is (115, 1). We have used two values for filters, 64 and 32, with some kernel size, 5. The values of parameters are strides, 1, and padding, some. The convolutional layer is processed using

$$x_i = f(w_i \otimes x_{i-1} + b_i) \quad (2)$$

where x is the sample of training input data, w_i is the weighted matrix, x_{i-1}, is the sample of training input data, $\otimes$ is the convolution operation, f is the activation function, and b_i is the basis of neural network.

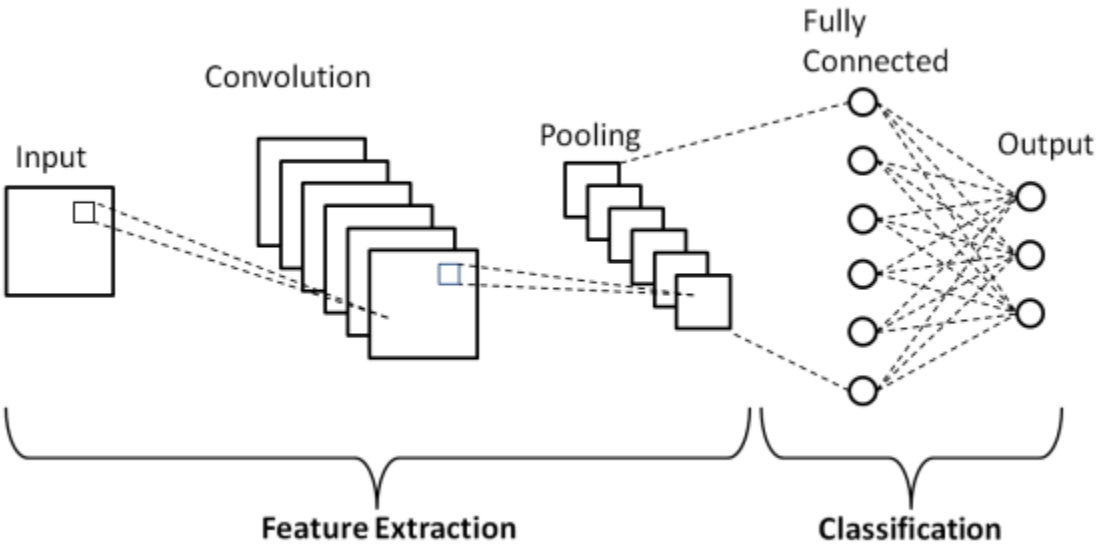

Figure 6: A generic architecture of a convolutional neural network (CNN) (Balaji, 2020)

- **Pooling Layer.** A pooling layer is used to reduce the number of parameters in the features map by selecting the maximum values in each region for designing a fit matrix average pooling. This matrix is processed into the next layer. For this research, the maximum pooling size of 5 is considered.

$$Q_j = Max(P_j^0, P_j^1, P_j^2, \dots, P_j^t) \quad (3)$$

where Q_j is the output results from the dataset, j is the pooling region, Max is the operation, and P_j^t is the element of the pooling.

- **Fully Connected Layer.** The last layer of the convolutional neural network is represented by the fully connected layer. Each node in the fully connected layer is connected directly to each node in layers (L − 1) and (L + 1). There is not any connection between nodes in the same layer, in contrast with the traditional ANN. Therefore, this layer takes a long training and testing time. At the same network, more than one fully connected layer can be used.

- **Long Short-Term Memory (LSTM).** The recurrent neural network (RNN) algorithm is one of the deep learning models used in many real-life applications. The long short-term memory model is one type of RNN. The LSTM is used to process sequence data that have feedback connect dissimilar to standard feedforward neural networks. The LSTM has three main gates: input gate, forget gate, and output gate. The input gate is used to store the training data in long-term memory. While the long-term memory initializes from the current input data, the short-term memory initializes from the previous time step. The input gate has filters used to extract training data and discard not useful information, whereas the useful information passes into sigma function. The sigma function has two indicator values: 0 and 1. The 1 value indicates the values that are very important, while the 0 value indicates values that are unimportant. The output from the input layer is saved in long-term memory. The forget gate is one of most significant gates in the LSTM model. It is used to decide which information to save or discard, by multiplying the forget vector values by current input gate.

The output from the forget gate will be passed to the next cell to obtain a new version from long-term memory. Figure 7 shows the structure of the LSTM model. Unlike standard feedforward neural networks, LSTM has feedback connections. It can process not only single data points but also entire sequences of data.

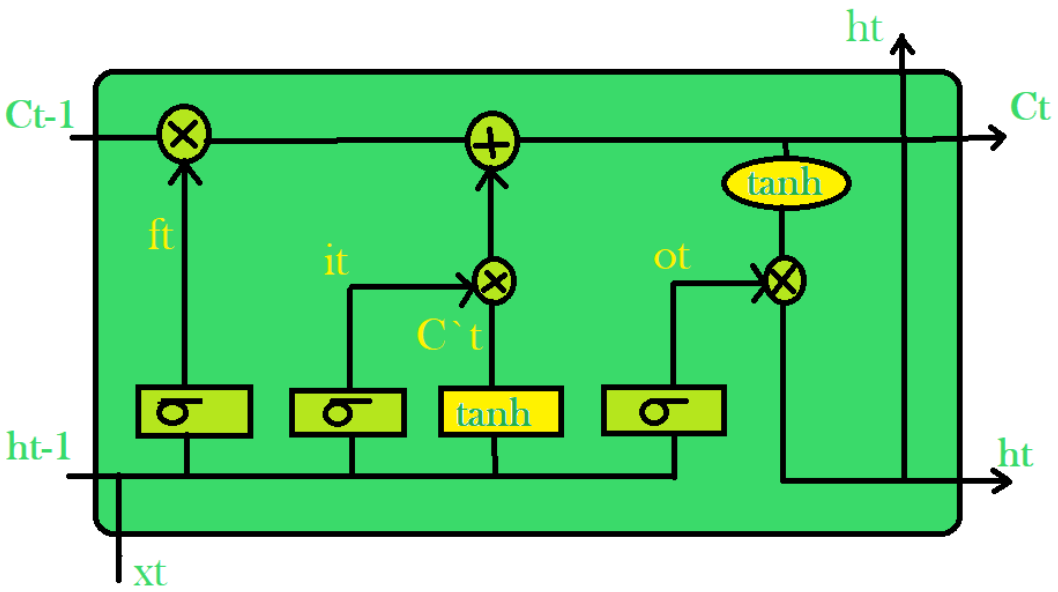

Figure 7: Structure of the LSTM model. (GeeksforGeeks, 2021)

In this research, the hybridized CNN and LSTM models are used to detect botnet attacks from various types of IoT devices. The 1D convolutional layers were used with the filter values 64 for the first layer and 32 for the another. The model has two LSTM layers with filter values of 32 and 16. The LSTM hidden layers are flattened and provided to the dense layers which provides the classified attacks. The ReLU function was used as the activation function for all layers except the last dense layer where Softmax function was used. Figure 8 displays a generic structure of the hybrid CNN-LSTM model that was used in this study. A snapshot of the CNN-LSTM model is presented in Figure 9.

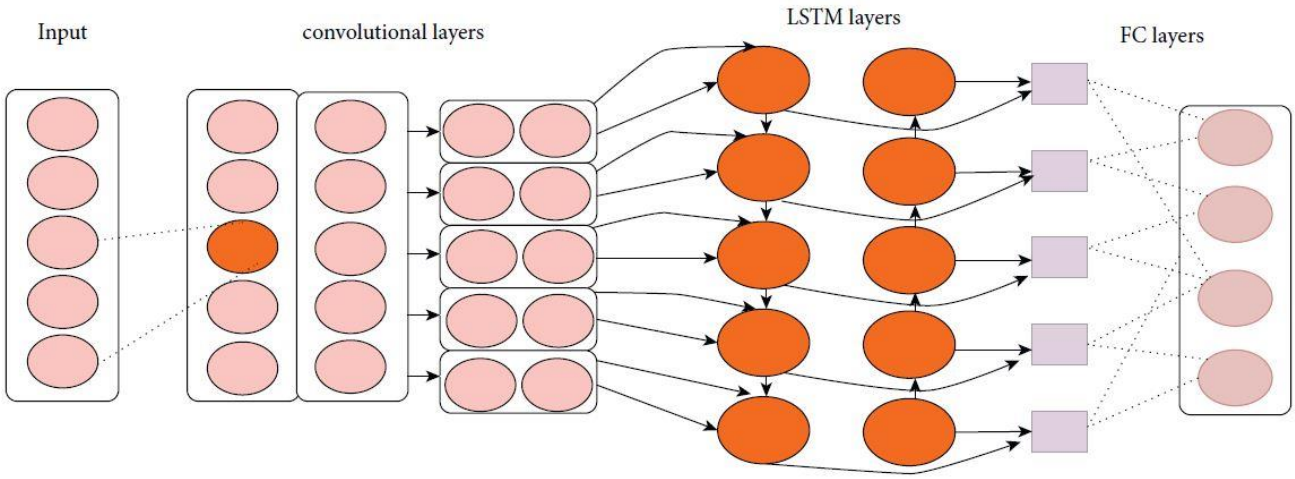

Figure 8: Structure of the hybrid CNN-LSTM model

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 115, 64)           384
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 115, 32)           10272
_________________________________________________________________
lstm (LSTM)                  (None, 115, 32)           8320
_________________________________________________________________
lstm_1 (LSTM)                (None, 115, 16)           3136
_________________________________________________________________
flatten (Flatten)            (None, 1840)              0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               235648
_________________________________________________________________
dense_6 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_7 (Dense)              (None, 11)                715
=================================================================
Total params: 266,731
Trainable params: 266,731
Non-trainable params: 0
_________________________________________________________________
```

Figure 9: Snapshot of the CNN-LSTM model

6.3.3 Deep Residual CNN. The residual network was first proposed by (He *et al.*, 2015) and has achieved great success in image processing field over the years. In general, the residual network has three significant features. First, the identity skip-connections are introduced, which allow data flowing from other layers directly to the subsequent layers. Second, the depth of the network structure can be largely extended using the skip connections. Finally, removing single layers from residual networks basically has no remarkable influence on the testing performance (He *et al.*, 2015). A residual learning block is shown in Figure 10, that can be defined as,

$$z = F(x, \{w_i\}) + x \tag{4}$$

Where x and z denote the input and output vectors of the layer, respectively. F represents the residual function. Take the structure in Fig. 1 for instance, $F = w_2\varphi(w_1^T.x)$ where φ is the nonlinear activation function, and the biases are not shown for simplicity. The operation $F(x, \{w_i\}) + x$ is performed by a short cut connection and element-wise addition. It should be noted that the dimensions of F and x are supposed to be the same. When they are not matched, linear projection of x can be used for the addition.

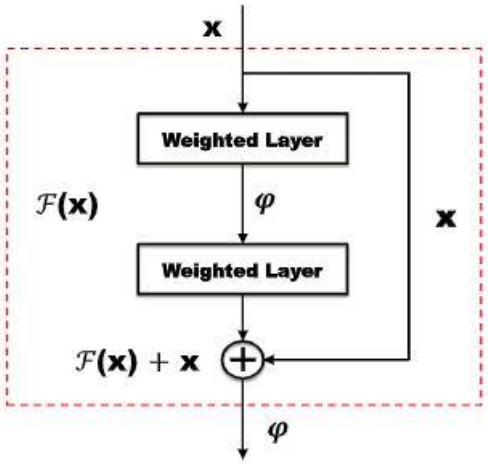

Figure 10: One building block in residual learning (He *et al.*, 2015)

A deep residual CNN model proposed in this study is shown in Figure 11. The deep convolutional neural network (DCNN) consists of 5 stacked 1D CNN structures, a max-pooling layer, a fully-connected layer and a softmax classifier. A 1-dimensional convolutional layer is first designed with 32 local filter kernels of 5 length window size. Multiple residual building blocks are stacked to learn high-level representations. By default, five building blocks are employed. In each block, two convolutional layers are adopted for the residual learning. For each convolutional layer in the proposed network, zeros-padding operation is implemented to keep the feature map dimension from changing. Pooling layers can be used in the network to reduce the number of parameters and accelerate the training process while keeping the significant feature information. In this study, one max-pooling layer is adopted between the residual building blocks. Finally, the high-level representations are put into a fully-connected layer and a softmax regression.

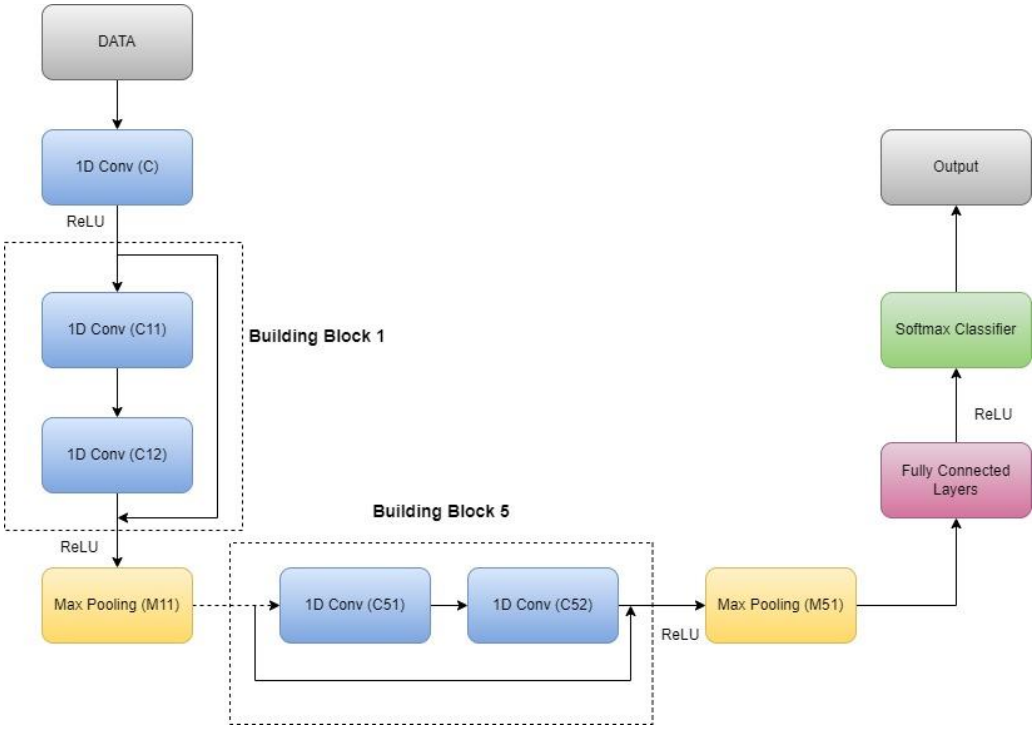

Figure 11: Proposed Deep residual CNN architecture

6.4 Evaluation Metrics

After successfully hyper tuning and training the model, the performance and accuracy of the model need to be evaluated. In this project, a classification report and roc curve were used as a parameter to assess the model performance.

6.4.1 Classification Report

The classification report is a method from slearn.metrics that gives accuracy, precision, recall, and f1-score. At the same time, it also displays the confusion matrix.

a. Accuracy: It is defined as the number of correct predictions from the total predictions. The mathematical expression for it is

$$Accuracy = Correct\ prediction\ /\ Total\ number\ of\ datasets.$$

However, it only works well on a balanced dataset, and the accuracy represented for an imbalanced dataset is biased.

b. Precision: It represents the number of correct predictions for that class given the prediction results. It is given by the following expression as

$$Precision = True\ Postive\ /\ (True\ Positive + False\ Positive)$$

c. Recall: It represents given a class, how much the classifier detects it and is given by the following formula as

$$Recall = True\ Postive\ /\ (True\ Positive + False\ Negative)$$

d. F-1 Score: It is the harmonic mean of recall and precision. It punishes the extreme value more and F1 Score is represented mathematically as

$$F1\ Score = 2 * (Precision * RecallPrecison + Recall)$$

e. Confusion Matrix: Confusion matrix provides a one-stop solution to monitor model performance on the imbalanced dataset. It provides a guide to calculate true positive (TP), true

negative (TN), false positive (FP) and false-negative (FN) for each class. Using the above count, it can be used to calculate accuracy, recall, precision, recall, and F1-score. A sample confusion matrix is in Figure 12 with markings for TP, TN, FP, and FN for a class.

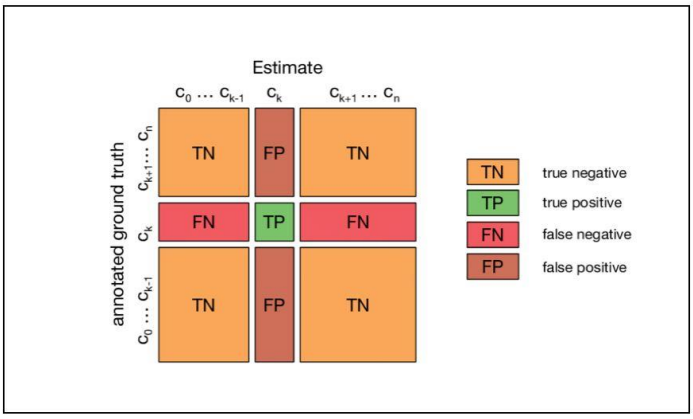

Figure 12: Illustration of Confusion Matrix

6.4.2 ROC Curve

ROC stands for receiver operating characteristic and is highly regarded for visualizing how much tradeoff one can make while training the model. It is a plot of True Positive Rate vs False Positive Rate at various thresholds. TPR is also called sensitivity and FPR is $1-specificity\ or\ true\ negative\ rate$. ROC AUC is the area under the curve often used to measure ROC Curve and a sample is shown in Figure 13.

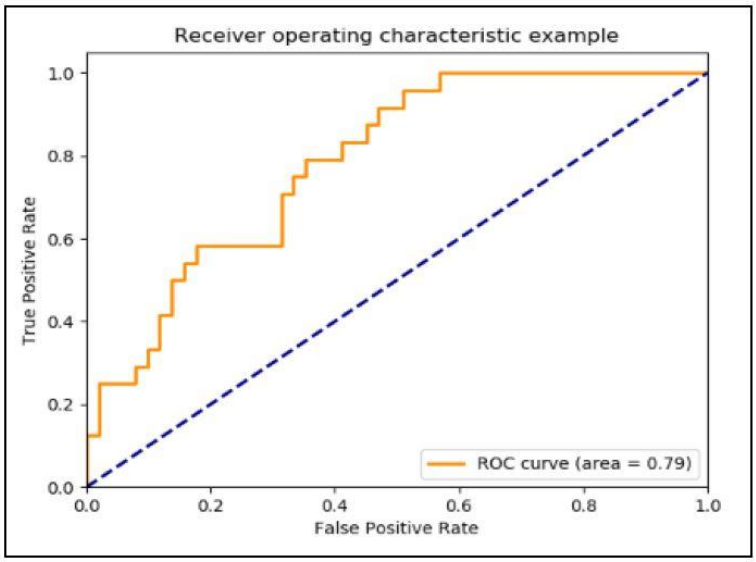

Figure 13: Interpretation of the ROC Curve

6.5 Experiment Setup

A. Hardware

The models built were trained on a Windows 10 workstation and on Google Cloud.

1) Workstation:

a. Processor: Intel Core i5-7200U CPU up to 3.1 GHz

b. RAM: 12.0 GB

c. System Type: 64-bit OS, x-64 based processor

2) Google Cloud:

a. Processor: Python 3 Google Compute Engine Backend

b. RAM: 13.0 GB

B. Software and Libraries

1) Jupyter Notebook: It is an open-source application that facilitates data preprocessing, statistical modeling, data visualization, machine learning and much more.

2) Google Colab: It is a colab notebook hosted on google cloud servers providing access to GPU's and TPU's for tasks that can be done in a Jupyter notebook

3) Kaggle: It allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers.

4) Libraries: Python was used as the scripting language for writing most of the code.

a. Scikit-learn: It is used for predictive analytics tasks like classification, regression, clustering, dimensionality reduction, model selection and preprocessing.

b. Imblearn: This library provides an API for imbalanced learning and wide samples.

c. Xgboost: It is a highly efficient gradient boosting library for xgboost classifiers.

d. Pandas: It is a data analysis and manipulation library implemented in python

e. NumPy: It provides numerical computing capability and high-level mathematical functions for multidimensional arrays and matrices.

f. Matplotlib & seaborn: This package was used to create visualizations.

6.5 Implementation

Initially, the N-BaIoT dataset was loaded from Kaggle public dataset. From the network traffic of nine devices, only the Provision PT-737E Security Camera's data was used for this research. Both the benign and malicious data were combined into a single object and then randomized. The labels for the class of attack were created for validating the model. The data was standardized, where the mean is zero and that the standard deviation is one, before feeding the data through the model. The standardized data is split for training, validation and testing purposes. The 80% of the data is provided for training and 20% for testing the model.

The training data is further split by providing 12.5% for validating the model.

Firstly, the ANN model was created with 5 dense layers which is activated by ReLU activation function. Secondly, the CNN-LSTM model was created with two 1D convolutional layers with 64 and 32 filter values, and two LSTM layer with 32 and 16 filter values. Thirdly, the proposed Deep Residual CNN model was created with five 1D CNN structures with each CNN layer having 32 filter kernel values. Each residual block is implemented with Maximum Pooling at the end.

The model is compiled with three arguments namely loss, optimizer and metrics. Categorical cross-entropy is used as loss function is designed to quantify the difference between two probability distributions. For training the networks, Adam optimization method is used with the learning rate, beta-1, and beta-2 of 0.001, 0.9, and 0.999, respectively. Learning rate is decayed exponentially with the decay factor of 0.75 every 10000 iterations. The callback is also applied in the form of early stopping. Models are trained by NumPy arrays using fit(), which evaluates the models on training. The epochs, which determines the number of times the model needs to be evaluated, is set to 30. Once the model is trained, the accuracy and the loss of the model is plotted against the epochs. In the end, the models were run on the test dataset to report the model performance.

7 RESULTS AND DISCUSSION

7.1 Experimental Results

To evaluate and examine the proposed system, experiment was conducted on Provision PT-737E Security Camera data. The deep learning algorithms were implemented to provide a 11-class classification of botnet attacks. The results of the three deep learning models discussed in this study to detect botnet attacks are presented.

7.1.1 Experiment 1 (ANN Model). The ANN model was applied to detect the anomaly from network data extracted from the Security camera. Figure 14 shows the results of the ANN model. The weighted averages of the performance of the proposed system in detecting attack anomalies are 78, 84, and 80% with respect to precision, recall, and F1-score metrics. The overall accuracy and loss calculated while testing the model is 80% and 0.278 respectively.

```
                precision    recall  f1-score   support

        benign       1.00      1.00      1.00     12463
     mirai_udp       0.81      0.70      0.75     12246
  gafgyt_combo       0.54      0.67      0.60      6258
   gafgyt_junk       1.00      0.99      0.99      6005
   gafgyt_scan       0.50      1.00      0.67     20798
    gafgyt_tcp       0.00      0.00      0.00     20788
    gafgyt_udp       0.99      0.97      0.98     12099
     mirai_ack       1.00      1.00      1.00     19226
    mirai_scan       1.00      1.00      1.00     13187
     mirai_syn       1.00      1.00      1.00     31361
mirai_udpplain       0.97      1.00      0.99     11221

      accuracy                           0.84    165652
     macro avg       0.80      0.85      0.82    165652
  weighted avg       0.78      0.84      0.80    165652

  74/5177 [..............................] - ETA: 7s - loss: 0.2693 - accuracy: 0.8260/usr/
  _warn_prf(average, modifier, msg_start, len(result))
5177/5177 [==============================] - 7s 1ms/step - loss: 0.2788 - accuracy: 0.8366
Test: accuracy = 0.836633  ;  loss = 0.278760
```

Figure 14: Performance of the ANN model in detecting attacks

Utilizing confusion metrics parameters, namely, true positives, false negatives, true negatives, and false positives, to detect the botnet attacks, Figure 15 shows the confusion metrics of the training model for identifying the pattern of botnet attacks. Figure 16 shows accuracy performance of the ANN model for identifying intrusion from Security camera device. The accuracy of the proposed system in detecting ten attacks and benign traffic from IoT devices is presented; it is noted that performance begins at approximately 40% and reaches 86% with 30 epochs. Figure 16 also shows the cross-entropy loss of ANN when training the data. The training loss of the system to detect the attacks has been reduced from 1.4 to 0.29.

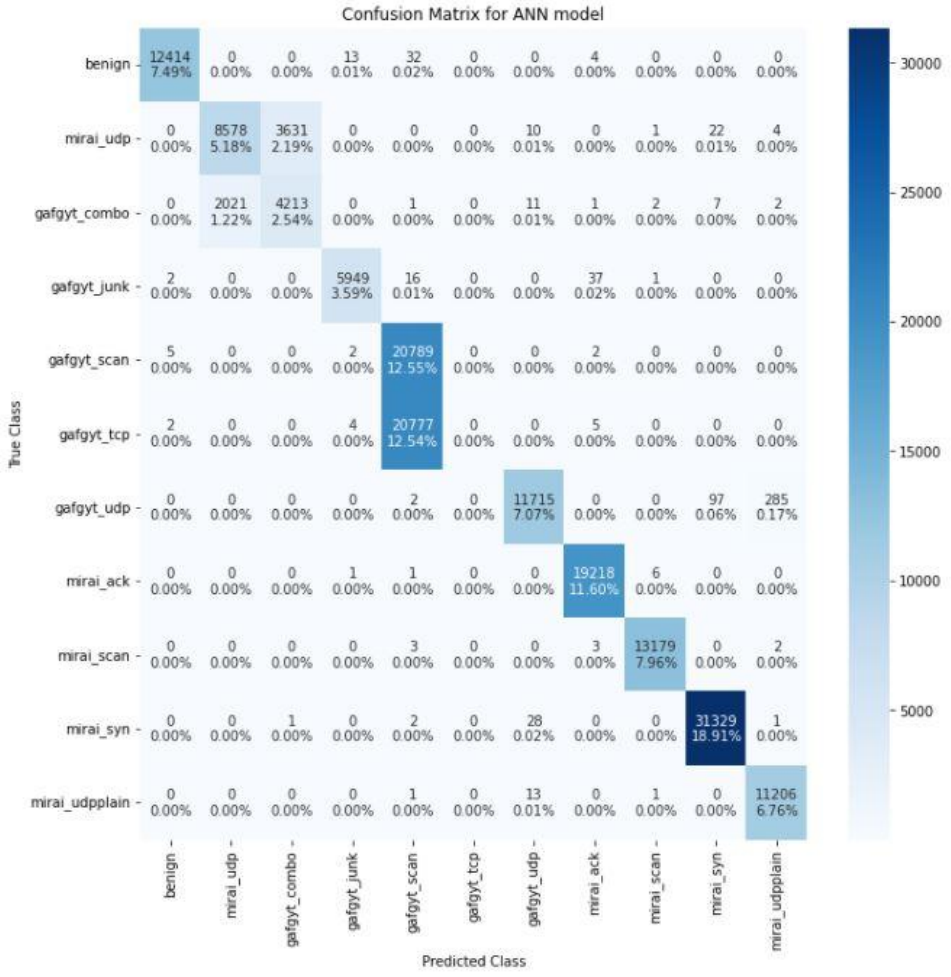

Figure 15: Confusion Matrix for ANN model

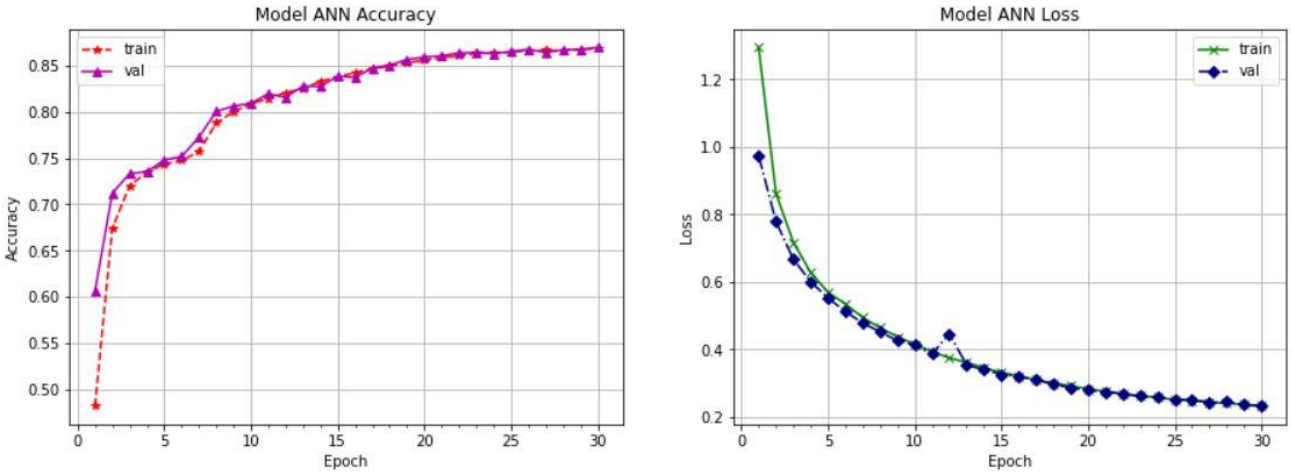

Figure 16: Accuracy and training loss of the ANN model to detect attacks

7.1.2 Experiment 2 (CNN-LSTM Model). In this experiment, the hybrid CNN-LSTM model was implemented to detect intrusion. Figure 17 summarizes the results of the CNN-LSTM for detecting botnet attacks. The weighted averages of evaluation metrics are 72%, 75%, and 70% for precision, recall, and F1- score metrics, respectively. Figure 18 shows the confusion metrics of CNN-LSTM in classifying botnet attacks from the network data that were extracted from the device. It is observed that the system struggled in detecting the botnet attacks compared to the ANN model. Figure 19 shows the performance of the CNN-LSTM model in identifying botnet attacks from thermostat devices that are set up in the IoT environment. Figure 19(a) shows that the accuracy of the CNN-LSTM model increases from 83% to 87% with 30 epochs. The training loss of the system is shown in Figure 19(b); it is noted that training loss is reduced from 0.26 to 0.17. When the model is tested with the testing data, the accuracy and loss were 75% and 0.45 respectively (shown in Figure 17).

	precision	recall	f1-score	support
benign	0.99	0.99	0.99	12463
mirai_udp	0.87	0.66	0.75	12246
gafgyt_combo	0.54	0.80	0.64	6258
gafgyt_junk	0.98	0.78	0.87	6005
gafgyt_scan	0.50	1.00	0.67	20798
gafgyt_tcp	0.12	0.00	0.00	20788
gafgyt_udp	0.85	0.29	0.43	12099
mirai_ack	0.86	0.99	0.92	19226
mirai_scan	0.96	0.84	0.90	13187
mirai_syn	0.79	0.95	0.86	31361
mirai_udpplain	0.82	0.93	0.87	11221
accuracy			0.75	165652
macro avg	0.75	0.75	0.72	165652
weighted avg	0.72	0.75	0.70	165652

```
5177/5177 [==============================] - 118s 23ms/step - loss: 0.4506 - accuracy: 0.7520
Test: accuracy = 0.752028  ;  loss = 0.450599
```

Figure 17: Performance of the CNN-LSTM model in detecting attacks

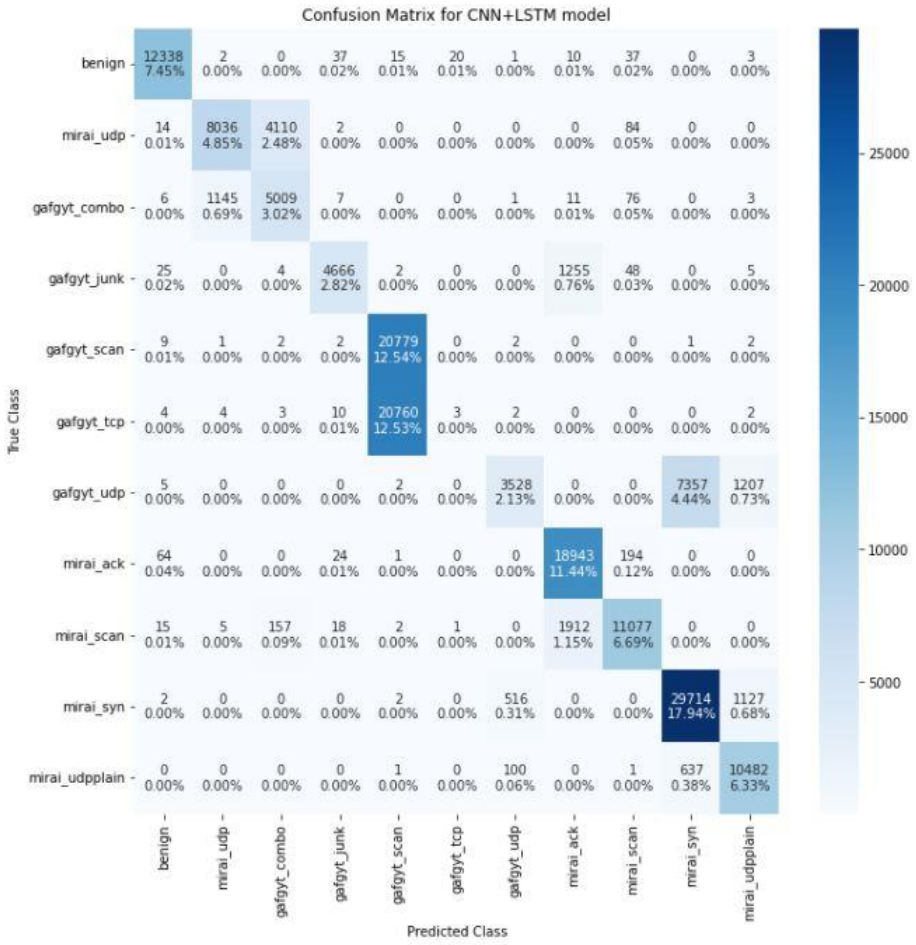

Figure 18: Confusion Matrix for CNN-LSTM model

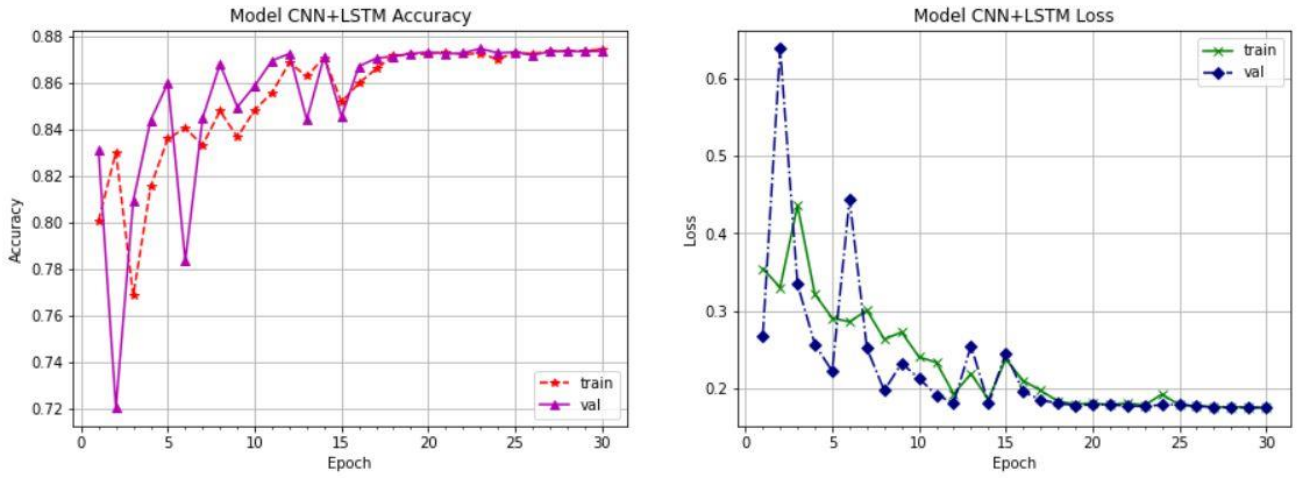

Figure 19: Accuracy and training loss of the CNN-LSTM model to detect attacks

7.1.2 Experiment 3 (Deep Residual CNN Model). In this experiment, the proposed Deep residual CNN model was tested to detect intrusion. The results of the proposed system are expressed in Figure 20. From the optimal results, the system has achieved good accuracy in finding unknown patterns from datasets to handle botnet attacks. The weighted averages of the system are 91%, 87%, and 83% for precision, recall, and F1-score metrics, respectively. The confusion metrics obtained through using this model is presented in Figure 21. It is shown that the system has the ability to train all the botnet attacks. Figure 22 shows the performance of Deep residual CNN model in detecting botnet attacks. The accuracy has been increased from 82% to 87.3%, whereas the training loss decreases from 0.3 to 0.17 with 30 epochs. The Deep residual CNN model produced the highest accuracy (87.2%) and least loss (0.175) among the other models, when tested with the testing data.

Predicted Class

	precision	recall	f1-score	support
benign	1.00	1.00	1.00	12382
mirai_udp	1.00	1.00	1.00	12106
gafgyt_combo	1.00	1.00	1.00	6262
gafgyt_junk	1.00	1.00	1.00	5951
gafgyt_scan	0.78	0.00	0.00	21048
gafgyt_tcp	0.50	1.00	0.66	20662
gafgyt_udp	1.00	1.00	1.00	12141
mirai_ack	1.00	1.00	1.00	19399
mirai_scan	1.00	1.00	1.00	13120
mirai_syn	1.00	1.00	1.00	31283
mirai_udpplain	1.00	1.00	1.00	11298
accuracy			0.87	165652
macro avg	0.93	0.91	0.88	165652
weighted avg	0.91	0.87	0.83	165652

```
5177/5177 [==============================] - 42s 8ms/step - loss: 0.1756 - accuracy: 0.8728
Test: accuracy = 0.872794  ;  loss = 0.175615
```

Figure 20: Performance of the Deep Residual CNN model in detecting attacks

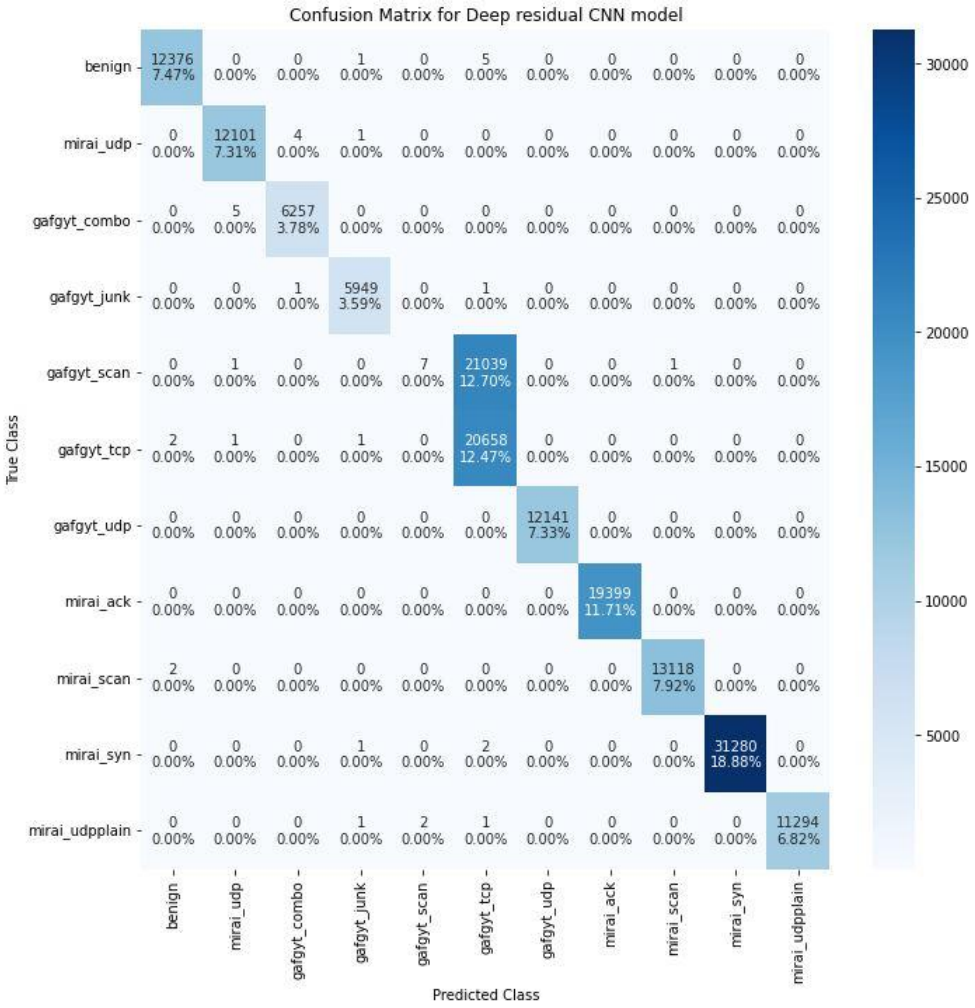

Figure 21: Confusion Matrix for Deep Residual CNN model

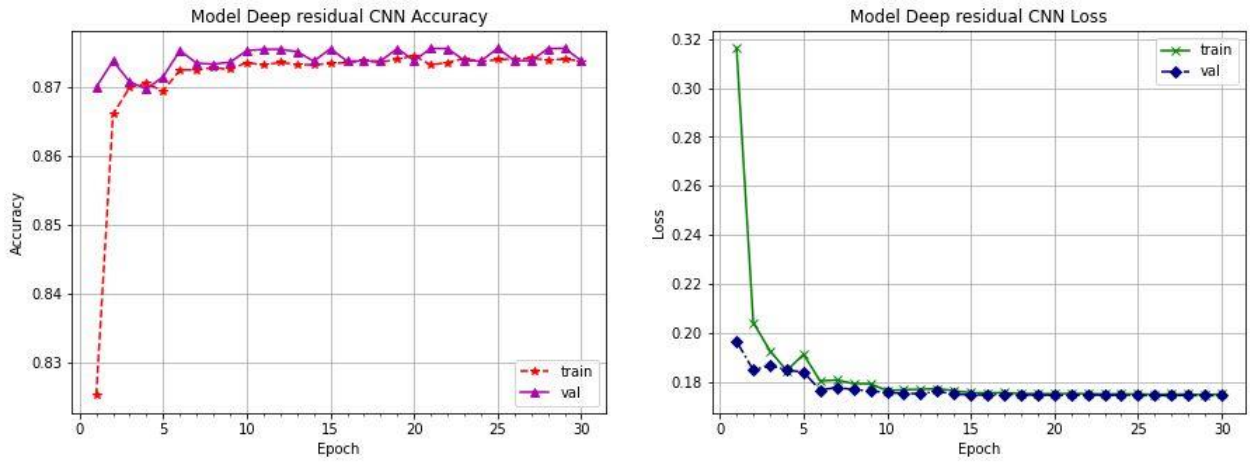

Figure 22: Accuracy and training loss of the Deep residual CNN model to detect attacks

Figure 23-25 displays receiver operating characteristic (ROC) curves for simulation results of the models for detecting botnet attacks. The ROC is used to measure the validation of the proposed system to detect botnets from IoT devices. The graphical representation (y-axis) is the recall metric for detecting ten attacks and benign traffic in nine different commercial devices; x-axis represents the specificity metric for detecting all botnet attacks.

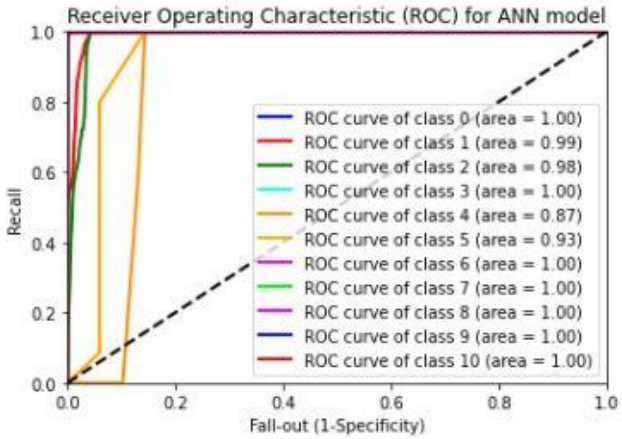

Figure 23: ROC curves of the ANN model for detecting botnet attacks

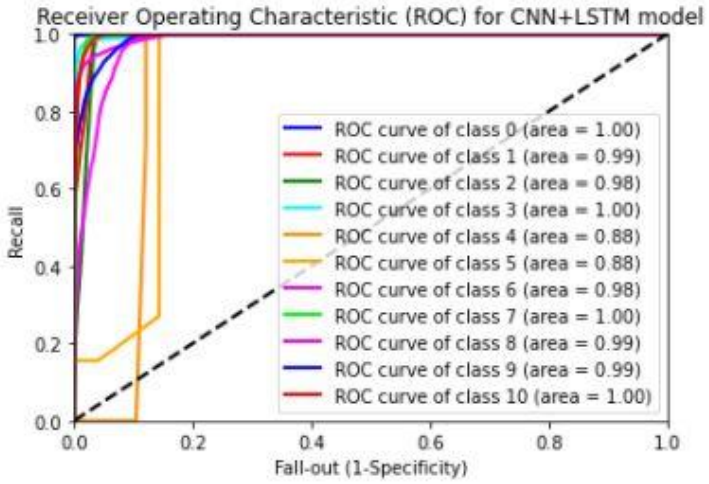

Figure 24: ROC curves of the CNN-LSTM model for detecting botnet attacks

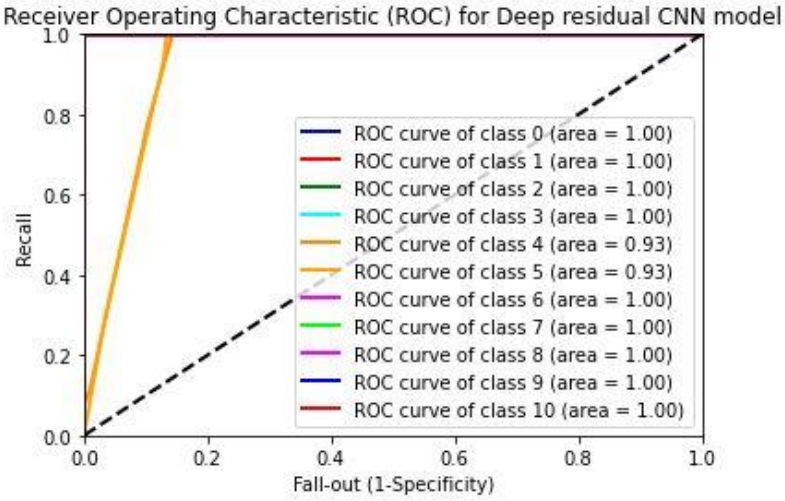

Figure 25: ROC curves of the Deep residual CNN model for detecting botnet attacks

7.2 Discussion

Botnet attacks are one of the serious attacks that threaten IoT devices. As we know, most of our real-life applications are based on IoT technology. The attackers have developed batch files of botnet attacks for preventing security system devices from recognizing these attacks. This makes it difficult for technology companies to design zero-day security system devices to protect the IoT environment. Therefore, using artificial intelligence models to detect botnet attacks, by extracting various unknown patterns that are developed by attackers, can easily help protect an IoT platform. In this research, three deep learning algorithms were applied to detect botnets. This system was tested by a dataset generated from nine commercial device injections from ten attacks.

The results of the first experiment, to identify botnet attacks using ANN model, is shown in Figure 14. The following points can be observed from the results:

- The ANN model achieved 100% with respect to precision, recall, and F1-score in detecting most attacks.
- The ANN system was unable to detect Bashlite TCP attack, where precision, recall, and F1-score are 0.0.
- The ROC curve of the model shows that 100% accuracy is achieved in detecting most attacks. The Bashlite scan and TCP attacks produced most false positives among the other attacks.

In the second experiment, the CNN-LSTM model was used to detect botnet attacks, as shown in Figure 17. The following points were obtained from the results:

- The CNN-LSTM model has achieved low performance in detecting most of the

attacks compared to other models. The performance of the proposed system was highest at 99% with respect to precision, recall, and F1-score metrics in detecting benign traffic.

- The accuracy and training loss of the model varied drastically during the first 15 epochs, after which it saturated.
- The overall accuracy and loss could have been improved by adding more convolutional layers with LSTM and increasing the number of epochs to fit the model.

In the third experiment, the proposed Deep residual CNN model was implemented to detect the botnet attacks, as summarized in Figure 20. The following points were obtained from the results:

- The proposed system attained 100% performance in classifying most botnet attacks in terms of precision, recall, and F1-score metrics.
- The Deep residual CNN model demonstrated low performance in detection of Scan attacks (precision (78%), recall and F1-score (0.00)), and TCP flood attacks (precision (50%) and F1-score (0.66)).

Overall, the Deep Residual model has the ability to detect botnet attacks from IoT devices with optimal performance. The proposed system showed low performance in detecting Scan and TCP flood attacks. Curve and confusion metrics are presented and proved the effectiveness and efficiency of the system to detect botnet attacks from nine commercial IoT devices, including the ten most serious attacks that infect the IoT environment.

Developing security system to detect the intrusion from IoT environment has played a pivotal role in protecting the IoT network. Table 11 summarizes Deep residual CNN model

results against existing systems. There are few studies that have used N-BaIoT datasets to detect the botnet attack from IoT network. (Liu, Liu and Zhang, 2019) applied CNN algorithms to detect botnet attacks from IoT devices. They have used the N-BaIoT dataset for multiple classification of attack types. Table 4 summarizes the results of the proposed Deep residual model with the existing CNN model implemented by (Liu, Liu and Zhang, 2019). In this study, the proposed system was examined with IoT device datasets extracted from nine devices. The accuracy results were compared against existing systems. Overall, we can observe that the proposed system shows better performance.

Table 4: Comparison of the proposed system with existing models (Liu, Liu and Zhang, 2019)

Attacks (Mirai)	**F1-score**		**Precision**		**Recall**	
	CNN	**Deep Residual CNN**	**CNN**	**Deep Residual CNN**	**CNN**	**Deep Residual CNN**
Benign	1.0	1.0	1.0	1.0	1.0	1.0
ACK	0.965	1.0	0.955	1.0	0.977	1.0
Scan	1.0	1.0	1.0	1.0	1.0	1.0
SYN	1.0	1.0	1.0	1.0	1.0	1.0
UDP	0.985	1.0	0.987	1.0	0.98	1.0
UDPPlain	1	1.0	1.0	1.0	1.0	1.0

8 CONCLUSION

For this dissertation, I have developed a system based on a deep learning algorithm to reduce the risks that IoT devices face from DDoS attacks. Identification of DDoS attacks in the early stages can help network security by speeding up operations to disconnect most of the IoT devices from Internet connections for preventing and stopping botnet attacks from accelerating. In this research, the N-BaIoT dataset generated from nine commercial IoT devices was used, which is injected by two major IoT attacks, namely, BASHLITE and Mirai botnet attacks. The BASHLITE attack has sub-attacks that are Junk, TCP flood, UDP flood, Scan, and COMBO, whereas the Mirai attack is categorized into ACK, SYN, Plain UDP, UDP flood, and Scan. The main findings of this research are as follows:

- The Deep residual CNN model has successfully achieved good results compared to the CNN-LSTM model and ANN model.
- The proposed system has achieved low accuracy in detecting Scan and TCP flood attacks in terms of evaluation metrics.
- The Deep residual CNN model has accomplished high results in detecting most botnet attacks. I believe that the proposed system can effectively enhance security in various types of IoT platforms by detecting botnet attacks.
- The main contribution of this study is to develop a framework by using advanced artificial intelligence to identify various unknown patterns from IoT devices to detect botnet attacks effectively and efficiently. In the future, this study can be developed to improve the detection of Scan and TCP flood attacks and also to work with different dataset and IoT devices.

9 REFERENCES

Al-Turjman, F. and Abujubbeh, M. (2019) 'IoT-enabled smart grid via SM: An overview', *Future Generation Computer Systems*, 96, pp. 579–590. doi: https://doi.org/10.1016/j.future.2019.02.012.

Amini, P., Araghizadeh, M. A. and Azmi, R. (2015) 'A survey on Botnet: Classification, detection and defense', in *2015 International Electronics Symposium (IES)*, pp. 233–238. doi: 10.1109/ELECSYM.2015.7380847.

Asthon, K. (2010) 'That ' Internet of Things ' Thing', *RFiD Journal*, p. 4986. Available at: www.itrco.jp/libraries/RFIDjournal-That Internet of Things Thing.pdf.

Atzori, L., Iera, A. and Morabito, G. (2010) 'The Internet of Things: A survey', *Computer Networks*, 54(15), pp. 2787–2805. doi: https://doi.org/10.1016/j.comnet.2010.05.010.

Balaji, S. (2020) *Binary Image classifier CNN using TensorFlow*. Available at: https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697.

Beaulieu, R. *et al.* (2015) 'SIMON and SPECK : Block Ciphers for the Internet of Things'.

Bekerman, D. and Zawoznik, A. (2016) *650Gbps DDoS Attack from the Leet Botnet*. Available at: https://www.imperva.com/blog/650gbps-ddos-attack-leet-botnet/.

De Cannière, C., Dunkelman, O. and Knežević, M. (2009) 'KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5747 LNCS, pp. 272–288. doi: 10.1007/978-3-642-04138-9_20.

Chen, S. *et al.* (2014) 'A Vision of IoT: Applications, Challenges, and Opportunities With China Perspective', *IEEE Internet of Things Journal*, 1(4), pp. 349–359. doi: 10.1109/JIOT.2014.2337336.

Cole, B. A., Mellor, M. and Noyes, D. (2009) 'Botnets : The Rise of the Machines'.

Dinh-Le, C. *et al.* (2019) 'Wearable health technology and electronic health record integration: Scoping

review and future directions', *JMIR mHealth and uHealth*, 7(9). doi: 10.2196/12861.

Diro, A. A. and Chilamkurti, N. (2018) 'Distributed attack detection scheme using deep learning approach for Internet of Things', *Future Generation Computer Systems*, 82, pp. 761–768. doi: https://doi.org/10.1016/j.future.2017.08.043.

Domingo, M. C. (2012) 'An overview of the Internet of Things for people with disabilities', *Journal of Network and Computer Applications*, 35(2), pp. 584–596. doi: https://doi.org/10.1016/j.jnca.2011.10.015.

Doshi, R., Apthorpe, N. and Feamster, N. (2018) 'Machine Learning DDoS Detection for Consumer Internet of Things Devices', in *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 29–35. doi: 10.1109/SPW.2018.00013.

Douligeris, C. and Mitrokotsa, A. (2003) 'DDoS attacks and defense mechanisms: a classification', in *Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology (IEEE Cat. No.03EX795)*, pp. 190–193. doi: 10.1109/ISSPIT.2003.1341092.

Evans, D. (2011) 'The Internet of Things: How the next evolution of the internet is changing everything', *CISCO white papers*, 1, pp. 1--11.

Feily, M., Shahrestani, A. and Ramadass, S. (2009) 'A Survey of Botnet and Botnet Detection', in *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, pp. 268–273. doi: 10.1109/SECURWARE.2009.48.

GeeksforGeeks (2021) *Understanding of LSTM Networks*. Available at: https://www.geeksforgeeks.org/understanding-of-lstm-networks/.

Ghayvat, H. *et al.* (2015) 'WSN- and IOT-Based Smart Homes and Their Extension to Smart Buildings', *Sensors*, 15(5), pp. 10350–10379. doi: 10.3390/s150510350.

Goodfellow *et al.* (2016) *Deep learning*.

He, K. *et al.* (2015) 'Deep Residual Learning for Image Recognition'.

Herzberg, B. (2016) ‘Breaking Down Mirai: An IoT DDoS Botnet Analysis’. Available at: https://www.pentestpartners.com/security-blog/what-is-mirai-the-malware-explained/.

Heslop, B. (2019) ‘By 2030, Each Person Will Own 15 Connected Devices — Here’s What That Means for Your Business and Content’. Available at: https://www.martechadvisor.com/articles/iot/by-2030-each-person-will-own-15-connected-devices-heres-what-that-means-for-your-business-and-content/.

Hoang, X. D. and Nguyen, Q. C. (2018) ‘Botnet Detection Based On Machine Learning Techniques Using DNS Query Data’, *Future Internet*, 10(5). doi: 10.3390/fi10050043.

Hsu, F.-H. *et al.* (2017) ‘Detecting Web-Based Botnets Using Bot Communication Traffic Features’, *Security and Communication Networks*. Edited by S. Wendzel, 2017, p. 5960307. doi: 10.1155/2017/5960307.

Iqbal, W. *et al.* (2020) ‘An In-Depth Analysis of IoT Security Requirements, Challenges, and Their Countermeasures via Software-Defined Security’, *IEEE Internet of Things Journal*, 7(10), pp. 10250–10276. doi: 10.1109/JIOT.2020.2997651.

Joseph, S. and Jasmin, E. A. (2015) ‘Stream computing framework for outage detection in smart grid’, in *2015 International Conference on Power, Instrumentation, Control and Computing (PICC)*, pp. 1–5. doi: 10.1109/PICC.2015.7455744.

Jung, W. *et al.* (2020) ‘IoT botnet detection via power consumption modeling’, *Smart Health*, 15, p. 100103. doi: https://doi.org/10.1016/j.smhl.2019.100103.

Kambourakis, G., Kolias, C. and Stavrou, A. (2017) ‘The Mirai botnet and the IoT Zombie Armies’, in *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*, pp. 267–272. doi: 10.1109/MILCOM.2017.8170867.

Kaspersky (2018) ‘Kaspersky Lab DDoS Intelligence Report: Long-lasting Attacks, Amplification Attacks and Old Botnets Make a Comeback’.

Kouretas, I. and Paliouras, V. (2019) ‘Simplified Hardware Implementation of the Softmax Activation Function’, in *2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pp. 1–4. doi: 10.1109/MOCAST.2019.8741677.

LaptrinhX (2017) *New IoT/Linux Malware Targets DVRs, Forms Botnet*. Available at: https://laptrinhx.com/new-iot-linux-malware-targets-dvrs-forms-botnet-1436581994/.

Leo, M. *et al.* (2014) ‘A federated architecture approach for Internet of Things security’, in *2014 Euro Med Telco Conference (EMTC)*, pp. 1–5. doi: 10.1109/EMTC.2014.6996632.

Liu, J., Liu, S. and Zhang, S. (2019) ‘Detection of IoT Botnet Based on Deep Learning’, in *2019 Chinese Control Conference (CCC)*, pp. 8381–8385. doi: 10.23919/ChiCC.2019.8866088.

Luzuriaga, J. E. *et al.* (2015) ‘Handling mobility in IoT applications using the MQTT protocol’, in *2015 Internet Technologies and Applications (ITA)*, pp. 245–250. doi: 10.1109/ITechA.2015.7317403.

Mahmoud, M., Nir, M. and Matrawy, and A. (2015) ‘A Survey on Botnet Architectures, Detection and Defences’, *International Journal of Network Security*, 17(3), pp. 272–289.

McDermott, C. D., Majdani, F. and Petrovski, A. V. (2018) ‘Botnet Detection in the Internet of Things using Deep Learning Approaches’, *Proceedings of the International Joint Conference on Neural Networks*, 2018-July. doi: 10.1109/IJCNN.2018.8489489.

Meidan, Y. *et al.* (2018) ‘N-BaIoT-Network-based detection of IoT botnet attacks using deep autoencoders’, *IEEE Pervasive Computing*, 17(3), pp. 12–22. doi: 10.1109/MPRV.2018.03367731.

Miorandi, D. *et al.* (2012) ‘Internet of things: Vision, applications and research challenges’, *Ad Hoc Networks*, 10(7), pp. 1497–1516. doi: https://doi.org/10.1016/j.adhoc.2012.02.016.

Mwagwabi, F., McGill, T. and Dixon, M. (2018) ‘Short-term and long-term effects of fear appeals in improving compliance with password guidelines’, *Communications of the Association for Information Systems*, 42(1), pp. 147–182. doi: 10.17705/1CAIS.04207.

Nagamalai, D., Dhinakaran, C. and Lee, J. K. (2007) ‘Multi Layer Approach to Defend DDoS Attacks

Caused by Spam', in *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, pp. 97–102. doi: 10.1109/MUE.2007.157.

Orsi, E. and Nesmachnow, S. (2017) 'Smart home energy planning using IoT and the cloud', in *2017 IEEE URUCON*, pp. 1–4. doi: 10.1109/URUCON.2017.8171843.

Prokofiev, A. O., Smirnova, Y. S. and Silnov, D. S. (2017) 'The Internet of Things cybersecurity examination', in *2017 Siberian Symposium on Data Science and Engineering (SSDSE)*, pp. 44–48. doi: 10.1109/SSDSE.2017.8071962.

Ramachandran, A., Feamster, N. and Dagon, D. (2006) 'Revealing Botnet Membership Using DNSBL Counter-Intelligence'.

Ratasich, D. *et al.* (2019) 'A Roadmap Toward the Resilient Internet of Things for Cyber-Physical Systems', *IEEE Access*, 7, pp. 13260–13283. doi: 10.1109/ACCESS.2019.2891969.

Sagiroglu, S. and Canbek, G. (2009) 'Keyloggers: Increasing threats to computer security and privacy', *IEEE Technology and Society Magazine*, 28(3), pp. 10–17. doi: 10.1109/MTS.2009.934159.

Shah, S. H. and Yaqoob, I. (2016) 'A survey: Internet of Things (IOT) technologies, applications and challenges', in *2016 IEEE Smart Energy Grid Engineering (SEGE)*, pp. 381–385. doi: 10.1109/SEGE.2016.7589556.

Sinche, S. *et al.* (2020) 'A Survey of IoT Management Protocols and Frameworks', *IEEE Communications Surveys Tutorials*, 22(2), pp. 1168–1190. doi: 10.1109/COMST.2019.2943087.

Singh, Manmeet, Singh, Maninder and Kaur, S. (2019) 'Issues and challenges in DNS based botnet detection: A survey', *Computers & Security*, 86, pp. 28–52. doi: https://doi.org/10.1016/j.cose.2019.05.019.

Sinha, K., Viswanathan, A. and Bunn, J. (2019) 'Tracking Temporal Evolution of Network Activity for Botnet Detection'. Available at: http://arxiv.org/abs/1908.03443.

Stevanovic, M. and Pedersen, J. M. (2014) 'An efficient flow-based botnet detection using supervised

machine learning', in *2014 International Conference on Computing, Networking and Communications (ICNC)*, pp. 797–801. doi: 10.1109/ICCNC.2014.6785439.

Su, S.-C. *et al.* (2018) 'Detecting P2P Botnet in Software Defined Networks', *Security and Communication Networks*. Edited by J. Díaz-Verdejo, 2018, p. 4723862. doi: 10.1155/2018/4723862.

Villamarin-Salomon, R. and Brustoloni, J. C. (2008) 'Identifying Botnets Using Anomaly Detection Techniques Applied to DNS Traffic', in *2008 5th IEEE Consumer Communications and Networking Conference*, pp. 476–481. doi: 10.1109/ccnc08.2007.112.

Wang, W. *et al.* (2009) 'A Novel Approach to Detect IRC-Based Botnets', in *2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*, pp. 408–411. doi: 10.1109/NSWCTC.2009.72.

Wu, D. *et al.* (2019) 'Evading Machine Learning Botnet Detection Models via Deep Reinforcement Learning', in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pp. 1–6. doi: 10.1109/ICC.2019.8761337.

Wu, M. *et al.* (2010) 'Research on the architecture of Internet of Things', in *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, pp. V5-484-V5-487. doi: 10.1109/ICACTE.2010.5579493.

Wu, T. *et al.* (2017) 'An Autonomous Wireless Body Area Network Implementation Towards IoT Connected Healthcare Applications', *IEEE Access*, 5, pp. 11413–11422. doi: 10.1109/ACCESS.2017.2716344.

Xu, L. Da, He, W. and Li, S. (2014) 'Internet of Things in Industries: A Survey', *IEEE Transactions on Industrial Informatics*, 10(4), pp. 2233–2243. doi: 10.1109/TII.2014.2300753.

Yao, R. *et al.* (2021) 'Intrusion detection system in the advanced metering infrastructure: A cross-layer feature-fusion CNN-LSTM-based approach', *Sensors (Switzerland)*, 21(2), pp. 1–17. doi: 10.3390/s21020626.

Zeidanloo, H. R. *et al.* (2010) ‘A taxonomy of Botnet detection techniques’, in *2010 3rd International Conference on Computer Science and Information Technology*, pp. 158–162. doi: 10.1109/ICCSIT.2010.5563555.

Zoph, B. and Le, Q. V (2018) ‘Searching for activation functions’, *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*, pp. 1–13.

10 APPENDIX

The links and files related to this project are provided in this section.

- The N-BaIoT dataset used in this study can be accessed at https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT.

- The project is implemented in Jupyter Notebook which was developed in Google Collab. The project can be accessed in Google Collab through: Botnet_Detection.ipynb - Colaboratory (google.com)

- The Github repository for this dissertation is available at KiranKumar4225/Dissertation (github.com)

- This github repository contains all the documentation and necessary file for this project.

- To run the project in Google Collab, the Kaggle API is required. This API is essential for accessing the N-BaIoT dataset from Kaggle to the Notebook. I have provided the Kaggle API (Kaggle.json) file in the github repository under the Project_files folder.